University of the Philippines Manila
College of Arts and Sciences
Department of Physical Sciences and Mathematics

Predicting Website's Visual Appeal using a Hybrid of Convolutional Neural Network and Support Vector Machine

A special problem in partial fulfillment
of the requirements for the degree of
Bachelor of Science in Computer Science

Submitted by:

Reinier T. Maristela

May 2017

ACCEPTANCE SHEET

The Special Problem entitled "Predicting Website's Visual Appeal using a Hybrid of Convolutional Neural Network and Support Vector Machine" prepared and submitted by Reinier T. Maristela in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Marvin John C. Ignacio, M.Sc. (*cand.*)
Adviser

EXAMINERS:

	Approved	Disapproved
1. Gregorio B. Baes, Ph.D. (*cand.*)		
2. Avegail D. Carpio, M.Sc.		
3. Richard Bryann L. Chua, Ph.D. (*cand.*)		
4. Perlita E. Gasmen, M.Sc. (*cand.*)		
5. Ma. Sheila A. Magboo, M.Sc.		
6. Vincent Peter C. Magboo, M.D., M.Sc.		

Accepted and approved as partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science.

Ma. Sheila A. Magboo, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences
and Mathematics

Marcelina B. Lirazan, Ph.D.
Chair
Department of Physical Sciences
and Mathematics

Leonardo R. Estacio Jr., Ph.D.
Dean
College of Arts and Sciences

Abstract

In predicting a website's visual appeal, hand-crafted feature extractors may not be able to determine all the relevant features to extract from a screenshot of a website's homepage. Also, since it is hand-crafted, there is a need to determine what features to extract. This study aims to use a hybrid of convolutional neural network and support vector machine to predict a website's visual appeal. Using the hybrid CNN-SVM model, the AI system would extract features from an image of a website's homepage and determine its visual appeal with respect to the age, gender, country, and educational level of the target users.

Keywords: Website Visual Appeal, Deep Learning, Convolutional Neural Network, Support Vector Machine

Contents

List of Figures

I. Introduction

A. Background of the Study

As of March 2016, there are 1 billion websites online in the Internet [1]. Starting from shopping, reading news, gaming, conducting researches, and down to social interaction, it is evident that majority of our needs are met through the Internet [2]. Millions of websites have been created to fulfill our needs with ease and convenience. Considering the number of websites that users can choose from, it is important for websites to capture the interests of its users.

When users are visiting a website for the first time, a website's visual design is the first thing that affects the users [2]. It is important for websites to have a good first impression because it is a way of capturing the interest of its users [3]. This determines whether a user would continue to use the site or surf elsewhere. Furthermore, a visually appealing website also increases users' satisfaction, minimizes perceived irritation, avoids distrust, and encourages users to return and continue to use its services [4, 5, 6, 7]. However, visual appeal alone does not guarantee success. Factors such as good usability [6], information and navigation designs [5], services, and content [8] also influence users' experience and satisfaction. Still, designing websites to be visually appealing should not be optional but a necessity.

A website's visual appeal is subject to two dimensions: (1) its visual design and (2) the visual preferences of its target users [9]. Some example of a website's visual design can be its colorfulness, visual weight distribution, visual complexity, and figure-background color contrast [2, 3, 7, 10]. Furthermore, a website's visual appeal is also subject to the visual preferences of its target users. Specifically, the age, gender, country of residence, and educational level of a user affects his/her visual preferences [9]. People with these same backgrounds share the same visual preferences. Thus, to maximize a website's visual appeal, users should receive website designs personalized to their visual preferences [9].

One way to determine a website's visual appeal is to conduct surveys. Reinecke et. al. [9] conducted an online survey to quantify the visual preferences of people around the world. The survey shows an image of a website's homepage for 500 milliseconds and asks the participants to rate its perceived visual appeal from a scale of 1 to 9, 1 being the least appealing and 9 the most appealing. With this approach, designers can know the designs that their target users prefer and can make appropriate improvements thereafter.

However, surveys take time to be accomplished. When determining the visual appeal of a design prototype, designers want the result to be immediately available to them [10]. Solving this problem, Reinecke et. al. [10] suggested to create methods that can predict a website's visual appeal from the data gathered from surveys. In this way, there is a rapid evaluation of website's visual appeal.

As an example, Reinecke et. al. [10] made their resources publicly available for anyone who wishes to determine their website's visual appeal. The figure below is an overview of the method's processes provided by Reinecke et. al. [10]. An image of a website's homepage serves as an input to the extraction of information. The extracted information together with the demographic backgrounds of the users will be used as input for the prediction model. The prediction model will output an aesthetic rating.

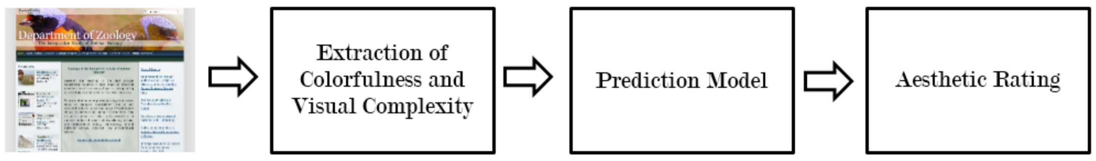


Figure 1: Overview of the predicting method

Their method starts by extracting information from an image of a website. They wanted to measure the colorfulness and visual complexity of the image. To do this, they extracted 12 image metrics from an image of a website. The figure below summarizes their implemented image metrics provided by Reinecke et. al. [10], 2013, p. 2052.

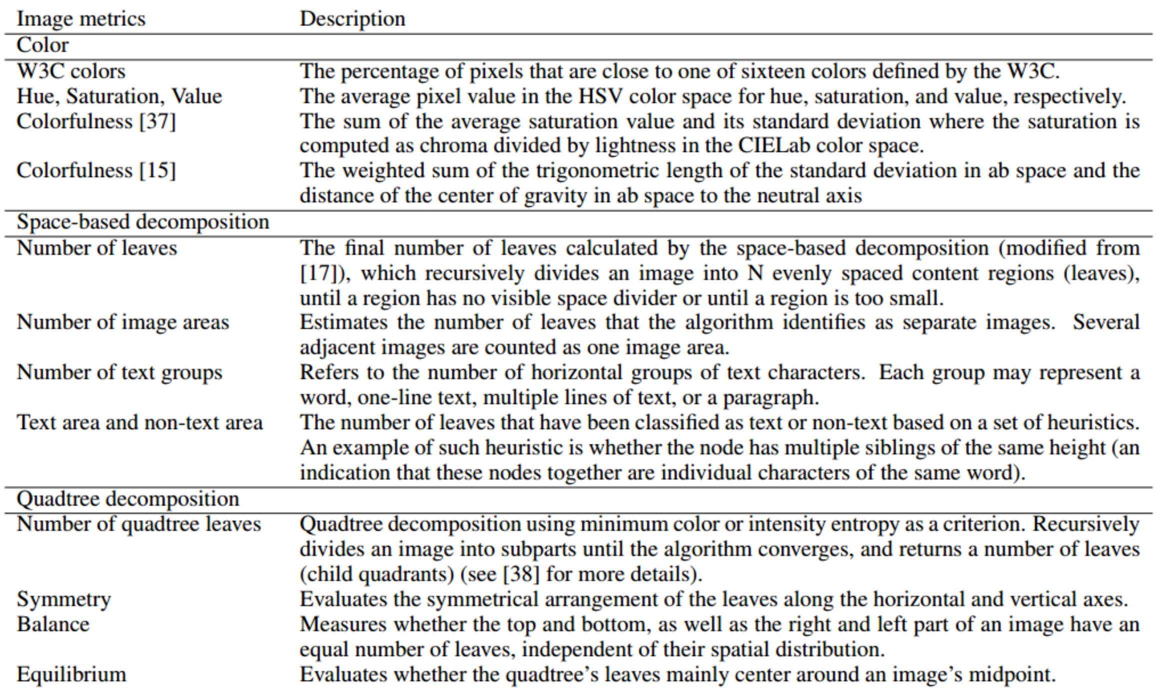


Image metrics	Description
Color	
W3C colors	The percentage of pixels that are close to one of sixteen colors defined by the W3C.
Hue, Saturation, Value	The average pixel value in the HSV color space for hue, saturation, and value, respectively.
Colorfulness [37]	The sum of the average saturation value and its standard deviation where the saturation is computed as chroma divided by lightness in the CIELab color space.
Colorfulness [15]	The weighted sum of the trigonometric length of the standard deviation in ab space and the distance of the center of gravity in ab space to the neutral axis
Space-based decomposition	
Number of leaves	The final number of leaves calculated by the space-based decomposition (modified from [17]), which recursively divides an image into N evenly spaced content regions (leaves), until a region has no visible space divider or until a region is too small.
Number of image areas	Estimates the number of leaves that the algorithm identifies as separate images. Several adjacent images are counted as one image area.
Number of text groups	Refers to the number of horizontal groups of text characters. Each group may represent a word, one-line text, multiple lines of text, or a paragraph.
Text area and non-text area	The number of leaves that have been classified as text or non-text based on a set of heuristics. An example of such heuristic is whether the node has multiple siblings of the same height (an indication that these nodes together are individual characters of the same word).
Quadtree decomposition	
Number of quadtree leaves	Quadtree decomposition using minimum color or intensity entropy as a criterion. Recursively divides an image into subparts until the algorithm converges, and returns a number of leaves (child quadrants) (see [38] for more details).
Symmetry	Evaluates the symmetrical arrangement of the leaves along the horizontal and vertical axes.
Balance	Measures whether the top and bottom, as well as the right and left part of an image have an equal number of leaves, independent of their spatial distribution.
Equilibrium	Evaluates whether the quadtree's leaves mainly center around an image's midpoint.

Figure 2: Overview of implemented image metrics

As we can observe, their method uses hand-crafted feature extractors. This means that a human expert decides what information to extract from the image and how to extract it. In their method, they only considered extracting information about the website's colorfulness and visual complexity. To better improve their method, they suggested to extract more information from the image [10].

Extracting more information from an image using hand-crafted feature extractors results to more algorithms being implemented. Furthermore, it is difficult to know what are the information to extract from an image of a website's homepage that best describe its visual appeal [2]. Solving the problems of hand-crafted feature extractors, we can use the technologies of artificial intelligence.

In computer science, there is a field called artificial intelligence (AI). This field aims to create intelligent machines and softwares [11]. During its early years, AI tackled mathematical problems that are difficult for humans to solve but easy for computers [11]. One approach to artificial intelligence is called the knowledge base where computers can reason using logical inference rules. Still, people struggled to create rules with enough complexity that can accurately describe our world [11].

Solving this problem, a field called machine learning was introduced. In machine learning, AI systems now have the ability to acquire their own knowledge by extracting patterns from data [11]. This means that there is no need for humans to create the formal rules that computers will use. In this field, one of the widely used algorithm in classification and detection tasks is the support vector machine (SVM) because of its high generalization ability [12].

The performance of machine learning algorithms depends on the representation of the data they are given [11]. For example, an AI system that is used to recommend a cesarean delivery does not directly examine the patient. It makes its recommendation using the relevant information that a doctor inputs to the system such as the absence or presence of uterine scar [11]. Relevant information that are used to represent a data are called features. These features should be extracted by humans from the data and input it as something that a computer can understand for the AI system to properly work. However, for many tasks, it is difficult to know what features should be extracted from the data [11].

Representation learning is another approach to AI that aims to use machine learning not only to discover the mapping from representations to output but also the representation itself [11]. With this approach, the problem of choosing the best features to extract from the data will be left for the algorithm to solve. However, for sophisticated tasks such as speech/face recognition, it is not enough to just learn how to represent the data. It is also important to know how to extract high-level, abstract features from the data.

Deep learning aims to solve this problem by introducing representations that are expressed in terms of other, simpler representations [11]. Just like how a human learns, deep learning gives computers the ability to build complex concepts out of simpler concepts. Humans can learn how to solve complex mathematical problems by first learning how to do basic arithmetic operations. Likewise, computers can now perform much more sophisticated tasks because of deep learning. Under this field, convolutional neural networks (CNN) are tagged as one of the best feature extraction methods [2]. The figure below shows the areas under AI and examples of technology provided by Goodfellow et. al [11], 2016, p. 9.

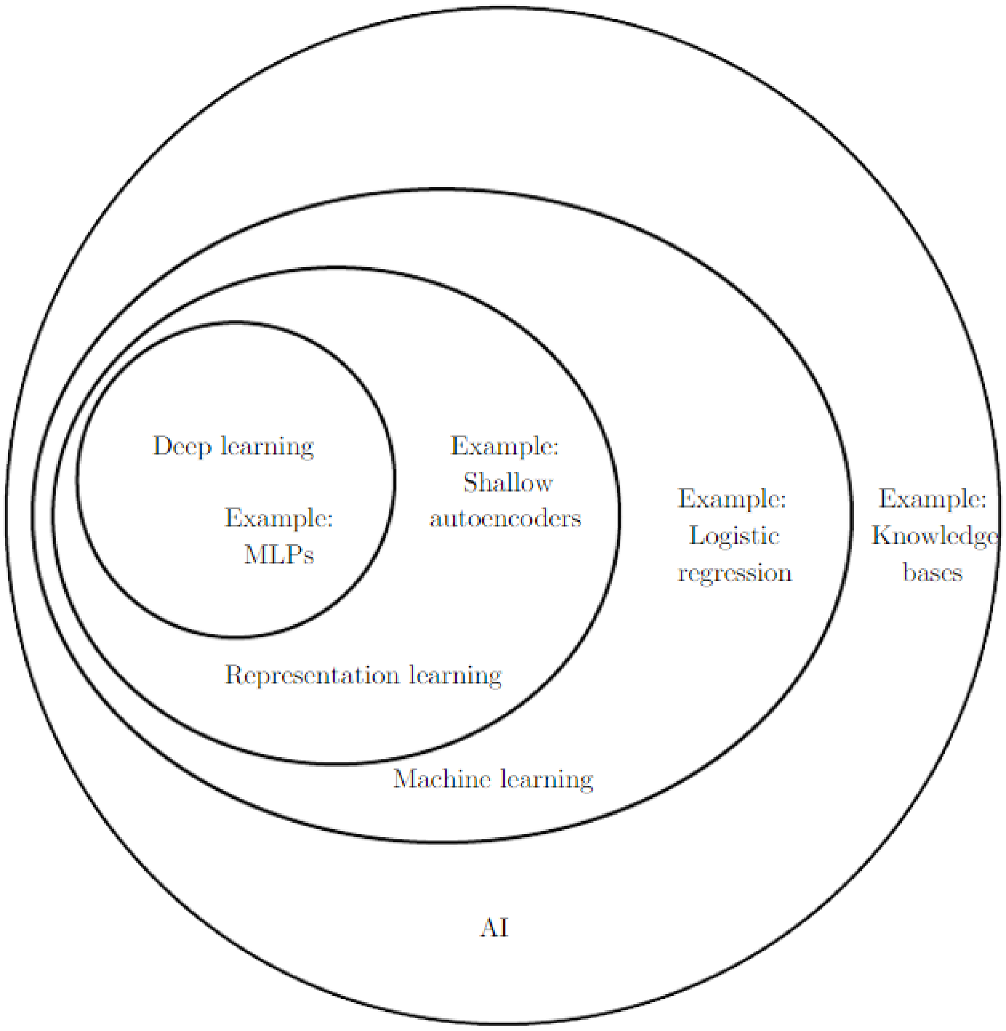


Figure 3: Areas under AI and examples of technology

With deep learning, we can let the computers do all the extraction of features that best represents a website's visual appeal. After that, we can also predict its visual appeal by considering the four demographic backgrounds of its target users. Specifically, we would use a hybrid of CNN and SVM to address the limitation of using hand-crafted feature extractors when predicting a website's visual appeal.

The first step in creating a hybrid CNN-SVM model is to gather a dataset to be used by the model. The dataset is needed so that the hybrid model can successfully perform its task. For predicting website's visual appeal, we would be using the dataset gathered by Reinecke et. al. [9]. The dataset can be accessed and downloaded at http://iis.seas.harvard.edu/resources/ [9]. This dataset consists of images of different websites that were rated by people around the world according to their visual appeal. The dataset also includes the age, gender, country, and educational level of the participants. They rated the visual appeal of websites with a scale of 1-9, 1 being the least visually appealing and 9 the most visually appealing. The next step is to divide the dataset into training set, validation set and test datasets. The purpose of the training and validation datasets is to adjust the configurations of the hybrid model so that it would be able to produce the correct output. The purpose of the test dataset is to make sure that the hybrid model performs equally well even if it is its first time seeing the data.

A hybrid CNN-SVM model is composed of three parts: a convolutional neural network (CNN) model, a principal component analysis (PCA) model, and a support vector machine (SVM) model. For the CNN, we would be using AlexNet. AlexNet is a CNN that was created by Krizhevsky et. al. that showed its strength in classifying images in the ImageNet Large Scale Visual Recognition Challenge in 2012 (ILSVRC 2012) [13]. To link the CNN model to the SVM model, we would be using the principal component analysis model. Principal component analysis is a technique that reduces the dimensionality of large multivariate datasets. AlexNet can output a feature vector of size 4096. This size is too large for the SVM and that is the reason we would be using PCA to reduce the dimension of the feature vector [2]. After reducing the dimension of the output of AlexNet, we would use the reduced feature vector and the demographic backgrounds of the targets users as input to the support vector machine. The SVM will then predict the website's visual appeal. In a hybrid CNN-SVM model, the CNN acts as the trainable feature extractor because it can accept raw image as its input and the SVM acts as the trainable classifier.

The next step is to train the hybrid model using the training and validation datasets and specifying the hyperparameters. Hyperparameters are parameters that control the training of the model. Training means adjusting the configurations of a model to produce the correct outputs with respect to the training and validation datasets. The goal of training is to reduce the incorrect outputs produced by the model. When the model is able to produce the correct outputs with respect to the training and validation datasets, we will evaluate its performance on a test dataset. The test dataset makes sure that the model performs equally well on data it has not yet seen. In other words, the test dataset makes sure that the model did not just memorize the contents of the training and validation datasets to produce the correct outputs. When the hybrid model performs well on the test set, we can now let the web designers to use the hybrid CNN-SVM model to predict a website's visual appeal.

On a similar note, a work performed by Khani et. al. [2] used a hybrid CNN-SVM model to predict a website's visual appeal. However, one limitation of their work is they have used the dataset of Khosla et. al. [14] (prediction of image memorability) in training their CNN model. Normally, when we want a model to do a certain task, we would train the model using a dataset that contains relevant data to accomplish the said task. For example, if we want the model to detect dogs in images we would use a dataset that contains images of dogs. Furthermore, they have only considered the country of the users in predicting a website's visual appeal. They have reported a 34.15% test error on their work.

B. Statement of the Problem

The predicting method provided by Reinecke et. al. [10] uses hand-crafted feature extractors. Using hand-crafted feature extractors leads to the following issues:

1. Since it depends on a human expert to extract the features, the expert may not capture all the relevant information from the image [15].
2. There is a need to decide what features to extract from the image that best represents a website's visual appeal [2].

C. Objectives of the Study

This study aims to develop a system that predicts a website's visual appeal using hybrid CNN-SVM. The system have the following functionalities:

1. Allows the web designers to:
   (a) Upload an image of a website's homepage
   (b) Choose the demographic backgrounds of the website's target users by:
      i. Age
      ii. Gender
      iii. Country
      iv. Educational Level
   (c) Receive a prediction of the website's visual appeal as:
      i. Visually appealing
      ii. Not visually appealing
2. Allows an AI expert to:
   (a) Create a dataset by:
      i. Specifying the name of the dataset
      ii. Selecting data to be grouped as the training dataset

    iii. Selecting data to be grouped as the validation dataset
    iv. Selecting data to be grouped as the test dataset
    v. Grouping the data according to ii, iii, and iv
    vi. Displaying the number of data in the training, validation, and test datasets

(b) Select a dataset to be used for creating and evaluating a hybrid CNN-SVM model

(c) Create a hybrid CNN-SVM model by:

    i. Creating a convolutional neural network model by:
        A. Specifying the name of the hybrid CNN-SVM model
        B. Selecting a convolutional neural network architecture
        C. Selecting hyperparameters (e.g. training epochs, solver type, learning rate, and batch size)
        D. Training the CNN using the training and validation datasets and hyperparameters
        E. Displaying training and validation errors for each epoch
    ii. Creating a principal component analysis model by:
        A. Training a PCA model using the training and validation datasets and trained CNN model
        B. Displaying the reduced dimensions of data
    iii. Creating a support vector machine model by:
        A. Selecting hyperparameters (e.g. type of kernel)
        B. Training the SVM using the training and validation datasets, and trained CNN-PCA model
        C. Displaying a 2-dimensional plot of the data and separating hyperplane

(d) Evaluate the performance of the created hybrid CNN-SVM model by:

    i. Classifying all the data in the test dataset

ii. Receiving a log file of the predictions

iii. Displaying the accuracy of the prediction

(e) Select the hybrid CNN-SVM model to be used by the AI system

D. Significance of the Study

This AI system can assist humans in predicting a website's visual appeal. We will let the computer and the hybrid CNN-SVM to do determine the relevant features to extract from a screenshot of a website's homepage. Thus, all the relevant information can be extracted from the image and we no longer have to determine the features to extract that best represents a website's visual appeal.

Furthermore, this system gives designers an immediate and convenient way of knowing a website's visual appeal. They do not need to create their own predicting methods or learn the technical skills needed to use the available methods. Thus, the determination of a website's visual appeal is immediately available to them. The system also provides them convenience because they will just upload a screen shot of a website and specify the age, gender, country, and educational level of their target users. The AI expert can also make sure that the system is up to date with the visual preferences of the users by conducting surveys in fixed intervals. Thus, this system can help designers in creating visually appealing websites.

E. Scope and Limitations

Since there is a standard dataset that will be used for training and testing the hybrid CNN-SVM model, the system's and dataset's scopes and limitations are the same.

1. The image uploaded has a size of at least 227 (width) x 227 (height).

2. The supported formats are .png and .jpeg.

3. Since an image will be uploaded, only the static elements will be evaluated.

4. The designer can only choose the age, gender, country, and educational level of its target users.

5. The prediction will be a classification of visually appealing or not visually appealing.

6. The system will not give the reasons of the prediction.

7. The supported CNN architecture is AlexNet.

8. The system will be trained and tested using the dataset provided by Reinecke et. al. [9]. The following are the descriptions of the dataset:

   (a) It can be accessed at http://iis.seas.harvard.edu/resources/.

   (b) It consists of 418 screen shots of different homepages of websites. Each having a size of 1024 (width) x 768 (height) pixels.

   (c) It consists of 436,631 visual appeal ratings of the websites using a scale from 1 to 9, 1 being the least appealing and 9 the most appealing.

   (d) It consists of 5 age groups:

      - 12 - 20 years old
      - 21 - 30 years old
      - 31 - 40 years old
      - 41 - 50 years old
      - 51 onwards

   (e) It consists of 2 genders:

      - Male
      - Female

   (f) It consists of 33 countries:

      - Argentina, Australia, Austria, Belgium, Brazil, Bulgaria, Canada, Chile, Denmark, Finland, France, Germany, Greece, Hungary, India, Ireland, Israel, Italy, Lithuania, The Former Yugoslav Republic Of Macedonia, Mexico, Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Serbia, Singapore, Spain, Sweden, United Kingdom, United States

(g) It consists of 7 educational levels:

- Pre-high school
- High school
- College
- Professional School
- Graduate School
- PhD
- Postdoctoral

F. Assumptions

1. The image uploaded to the AI system is a screenshot of a website's homepage.

II. Review of Related Literature

There are good reasons why web designers should aim at making visually appealing websites. According to a study conducted by Lindgaard et. al. [16], it only takes about 50 milliseconds for users to assess a website's visual appeal. This means that users can form a stable first impression within 50 ms of seeing a website. Therefore, designers should aim at making a good first impression within this timeframe. A good first impression is one way of capturing the interest of the users.

In another study by Liu et. al. [4] that used a job-hunting website, they found that the aesthetic design of a homepage can affect users' emotional experience and perceived ease of use that will affect users' satisfaction. They suggested that web designers should pay special attention to their aesthetic design in order to increase users' satisfaction.

In online shopping, Hasan [5] concluded that a visually attractive website can minimize perceptions of irritations among its current and potential customers. Thus, maintaining an attractive and efficient website is key for attracting and retaining customers.

Furthermore, Seckler et. al. [6] conducted a research that aims to study the content of trustful and distrustful user experiences on the web. One of their findings is that bad visual designs are frequently mentioned in distrustful experiences. This suggests that designers should focus on improving the graphic and structure design of websites to avoid distrust among its users.

Thus, a designer should aim for a visually appealing website because it captures the attention of users, increases their satisfaction, minimizes perceived irritation, helps in avoiding distrust, and can attract and retain customers.

However, visual appeal is not the only thing to consider in creating successful websites. Hasan [5] stated that websites should also have effective navigation and information designs. Seckler et. al. [6] also suggested that websites should have good usability so users would trust it more. Conforming to this, Lindgaard et.

al. [8] concluded that users cannot gain a realistic picture of objective usability and trustworthiness from looking briefly at a homepage. Although this study focuses on a website's visual appeal, these are still important to consider in making successful websites.

A website's visual design affects its visual appeal. Khani et. al. [2] reviewed that the colors, visual weight distribution, presence of intelligent titles and images, number of text sections, graphic elements, and links, page dimension, and complexity of reading are some designs that we must consider when creating visually appealing websites. In another study by Reinecke et. al. [10], they found that colorfulness and visual complexity are the most noticeable design at first sight. Furthermore, Tuch et. al. [3] concluded that low visual complexity and high prototypicality were perceived as highly appealing. Lin et. al. [7] stated that a well-designed website with adequate visual complexity and figure-background color contrast can attract and retain customers.

Furthermore, a website's visual appeal is also subject to individual preferences. Reinecke et. al [9] conducted an online survey that aims to quantify the visual preferences of people around the world. In their study, they were able to determine that the age, gender, country of residence, and educational level of users significantly influence their visual preferences. A certain website design may be visually appealing to females but not in the case of males. For example, females prefer more colorful websites than male. Likewise, older and higher educated users may have different visual preferences compared to younger users. People sharing the same demographic backgrounds share the same visual preferences. Thus, users should receive visual designs personalized to their preferences to maximize the website's visual appeal.

With all the different factors to consider when determining a website's visual appeal, Reinecke et. al. [10] suggested to create methods that can predict a website's visual appeal. In this way, designers have a rapid evaluation of their website designs without having to conduct surveys. As an example, they provided a method that lets users determine their website's visual appeal by using statistical models. The method starts by extracting 12 image metrics to quantify a website's colorfulness and visual complexity. These metrics were then used in predicting a user's first impression about a website's visual appeal.

However, since they only considered the colorfulness and visual complexity of the image, they suggested to include more image metrics to improve their method. Furthermore, their method uses hand-crafted feature extractors. This means that they are the ones that determined the relevant information to extract from the image. As extension of their work, they suggest to extract more features from the image.

As stated before, colorfulness and visual complexity are not the only designs that can affect a website's visual appeal. This means that there are more relevant information to extract from a website that can describe its aesthetic. However, as stated by Khani et. al. [2], it is difficult to determine what are the best features to extract from an image that can best describe a website's aesthetic. Solving the limitation of using the hand-crafted feature extractors of Reinecke et. al. [10]'s method, Khani et. al. [2] proposed a methodology that uses the technologies of deep learning.

Convolutional neural networks (CNN) have shown their strength at different tasks without the need of hand-crafted feature extractors. A study by LeCun et. al. [15] showed that CNN has outstanding performance in document recognition. Krizhevsky et. al. [13] used CNN to classify 1.2 million images into 1000 different classes. They also entered a variant of their model in the ImageNet Large-Scale Visual Recognition Challenge (ILSVRC) 2012 and achieved a winning top-5 error rate of 15.3%. Furthermore, Szegedy et. al. [17] also used CNN in the ILSVRC 2014 competition and outperformed the other methods. Also, Zhou et. al. [18] used CNN for the task of scene recognition.

Furthermore, CNN can be also be used for predicting tasks. Khosla et. al. [14] used CNN to predict the memorability of an image. Because of their work, one can now estimate the memorability of images and use the prediction accordingly. For example, learning visual materials should have high memorability and users can use their method to estimate the image's memorability and adjust accordingly in case of a low memorability prediction. Doing so can help the people consume information efficiently.

Another predicting task that uses CNN is the work done by Lu et. al. [19]. In this study, they have used CNN as a pictorial aesthetic evaluation system. They are rating the aesthetic value of images that consists of real world objects such as animals, natural scenes, people, and etc. Whereas in this paper, we aim to evaluate images of websites that consists of links, buttons, texts, and etc. Furthermore, we will not be using their architecture is because of the amount of images in our dataset. Their datasets consists of thousands of images and CNN works well when there are thousands of images.

The work of Khani et. al. [2] is the closest solution for Reinecke et. al. [10]'s problem. Khani et. al. [2] used a hybrid of CNN and SVM that predicts a website's visual appeal. The CNN acts as the trainable feature extractor and SVM acts as the classifier. However, one limitation of their work is they have used the dataset of Khosla et. al. [14] (prediction of image memorability) in training their model. Normally, when we want a model to do a certain task, we would train the model using a dataset that contains relevant data to accomplish the said task. For example, if we want the model to detect dogs in images we would use a dataset that contains images of dogs. Thus, they have concluded that even though their model was trained on another dataset, it performs equally well compared to Reinecke et. al. [10]'s method. Furthermore, they have only considered the country of the users in predicting a website's visual appeal.

III. Theoretical Framework

A. Predicting Website's Visual Appeal

Aesthetics deals with the creation and appreciation of beauty [16]. However, we are unsure about what is being judged when talking about the aesthetic value of something. It may be the properties of the object, subjective experiences, emotional reactions, or cognitive judgments [16]. However, since this study is not about defining aesthetics, we will use the term ”visual appeal” in accordance to [16]. Visual appeal is the measure of how a user likes a website's visual design [9]. To rate a website's visual appeal, users can use a scale from 1 to 9, 1 being the least appealing and 9 the most appealing with accordance to [9]. In this way, we can quantify a website's visual appeal.

A website's visual appeal is subject to its visual design and the visual preferences of its target users. This means that a method aiming to predict a website's visual appeal must be able to extract features from an image of a website that best represents its visual appeal. Also, the method must also consider the age, gender, country, and educational level of the target users when doing the prediction.

Since the aim of this paper is to not use hand-crafted feature extractors, the predicting method would be using the technologies of deep learning to do the extraction of features from an image of a website.

For the visual preferences of the target users, we can use the data from the survey conducted by Reinecke et. al. [9]. In this survey, they have found that the age, gender, country of residence, and educational level of the users influence their visual preferences. For example, females like colorful websites more and colorless websites less than males [9]. For visual complexity, they observed that females disliked simple websites more compared to male. Users that are 41 years old and above prefer websites with higher complexity and colorfulness compared to younger users [9]. Furthermore, they observed that users living in countries in close proximities share similar visual preferences [9]. For example, they reported that Finland and Russia preferred the lowest visual complexity and colorfulness. Likewise, they reported that users from Macedonia, Serbia, and Bosnia prefer highly colorful websites. An interaction of these four demographic variables affect a user's visual preferences. For example, a higher educated female above 40 years old prefer a similar colorfulness preference to the average male [9]. With this data, we can extract them as features and use it for the prediction method.

B. Convolutional Neural Networks

Deep learning is an area under artificial intelligence that can represent features in terms of other simpler features. Also, the technologies under deep learning are able to learn how to extract features from a raw image. With this, we can let the CNN extract the features of an image of a website. Thus, deep learning solves the limitations of using hand-crafted feature extractors.

Before defining what is a convolutional neural network, let us define a deep feedforward neural network first. Deep feedforward networks defines a mapping $y = f(x, \theta)$ and learns the value of the parameters θ that result in the best function approximation [11]. Its goal is to approximate some function f*.

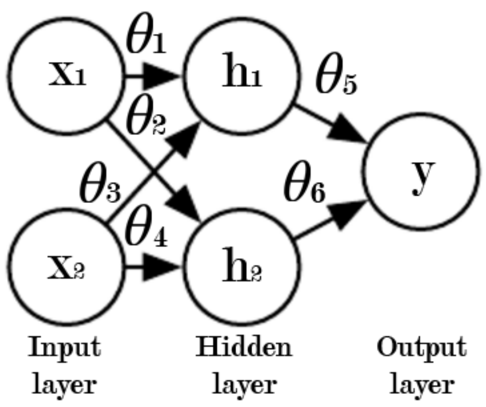


Figure 4: Example of a Feedforward Neural Network

Convolutional neural network (CNN) is a special kind of deep feedforward neural network for processing data with a known, grid-like topology such as images [11]. A neural network becomes a CNN when it used a mathematical operation called convolution in at least one of its layers [11]. A convolution operation can be visualized with the figure below provided by Goodfellow et. al [11], 2016, p. 334.

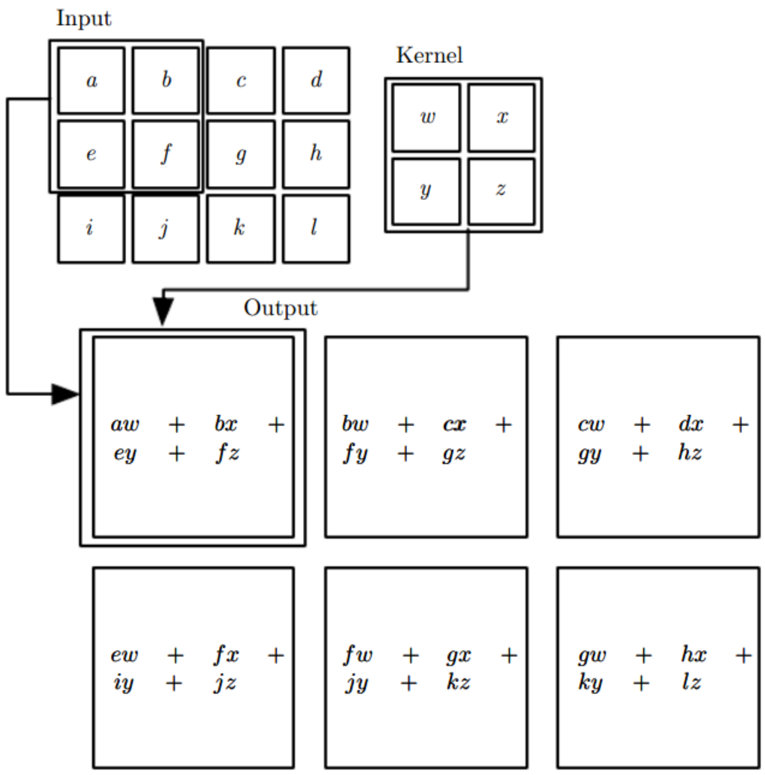


Figure 5: Example of 2-D Convolution

The convolution operation is the reason why we can input the RGB values of an image. We would just input an image and let the CNN do the feature extraction. Thus, the problem of using hand-crafted feature extractors in images is addressed by the CNN.

CNN can be viewed as the composition of two parts: an automatic feature extractor and a trainable classifier [20]. In our case, we will only be using the automatic feature extractor of the CNN. This is because CNN needs thousands or millions of images to work properly. Since our dataset only have a small number of images, we will have to resort to other algorithms for the prediction.

B..1 AlexNet

For this study, we will be using AlexNet in accordance to [2]. AlexNet, created by Krizhevsky et. al. [13], is a convolutional neural network that showed its strength in classifying millions of images into 1000 different classes in the ImageNet Large-Scale Visual Recognition Challenge in 2012. The figure below is the architecture of AlexNet provided by Krizhevsky et. al. [13], 2012, p. 1101.

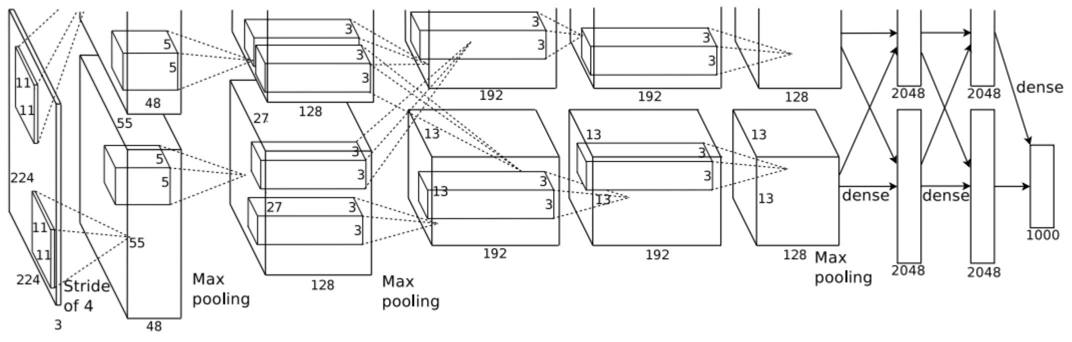


Figure 6: Architecture of AlexNet

C. Support Vector Machines

Support vector machine is a machine learning algorithm used for two-group classification problems [12]. It is one of the widely used machine learning algorithm because of its high generalization ability [12]. SVM has high generalization ability because it works by finding the optimal separating hyperplane. The optimal separating hyperplane is defined by the support vectors [12]. For visualization, the figure below provided by Cortes et. al. [12], 1995, p. 275 shows the optimal hyperplane and the support vectors are marked with grey squares.

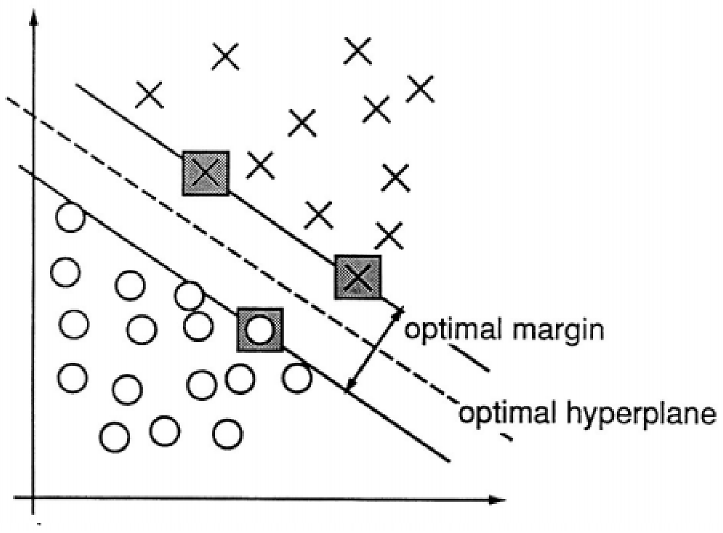


Figure 7: Optimal Separating Hyperplane and Support Vectors

Since the method will only predict a good or bad visual appeal, we will be using SVM. Moreover, SVM also works with a small number of data.

D. Hybrid CNN-SVM

A hybrid CNN-SVM model aims to integrate the strength of CNN and SVM [20]. The CNN can take the image of a website as its input. However, the limited number of images in the dataset does not let us use just the CNN. Thus, the CNN works as the trainable feature extractor for the hybrid model.

SVM can work with small number of data. However, it cannot extract features from images on its own. The AI expert must be the one that inputs the features for the SVM. With CNN acting as the feature extractor, that problem is solved. Thus, the SVM works as the trainable classifier for the hybrid model.

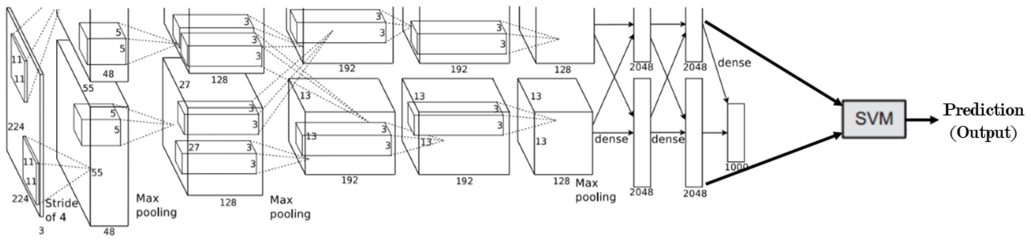


Figure 8: Example of a Hybrid CNN-SVM

E. Principal Component Analysis

Principal component analysis (PCA) is one technique for reducing the dimensionality of large multivariate datasets [21]. It works by replacing the p original variables by a smaller number q. The derived variables, which are the principal components, are linear combinations of the original variables [21]. For its role in the hybrid CNN-SVM model, we can use PCA to reduce the dimensionality of the feature vector that the CNN outputs.

IV. Design and Implementation

A. Use Case Diagram

The figure below shows the use case diagram of the AI system. It has two actors - the Web Designer and the AI Expert. The Web Designer is the one that primarily uses the AI system to predict the visual appeal of their website. The Web Designer can upload a screenshot of a website's homepage, specify the demographic backgrounds of the target users, and receive a prediction of the website's visual appeal. The AI expert is the one in charge of preparing the datasets, creating a hybrid CNN-SVM model, and evaluating the performance of the hybrid model.

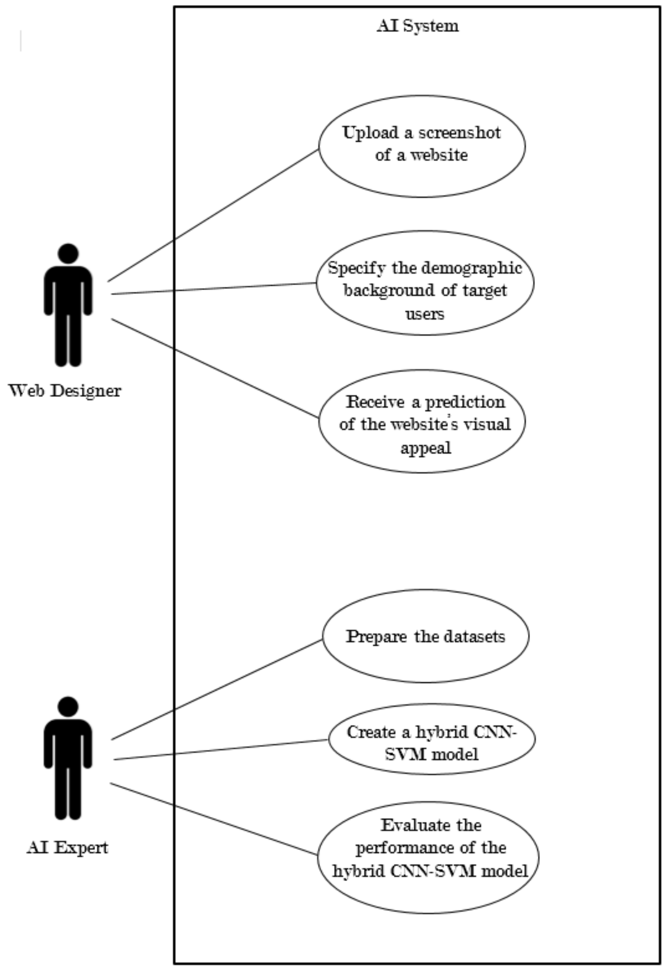

AI System
Upload a screenshot of a website
Specify the demographic background of target users
Receive a prediction of the website's visual appeal
Prepare the datasets
Create a hybrid CNN-SVM model
Evaluate the performance of the hybrid CNN-SVM model
Web Designer
AI Expert

Figure 9: Use Case Diagram

B. Activity Diagram

The figure below shows the activity diagram of the Web Designer. First, the Web Designer uploads an image of a website's homepage. After uploading, he/she selects the age, gender, country, and educational level of the target users. Then, the AI system will output a prediction of good or bad visual appeal. After this, the Web Designer can further select different demographic backgrounds to obtain a new prediction with the existing image. If there is another image available, he/she can also upload a new image and start the process again.

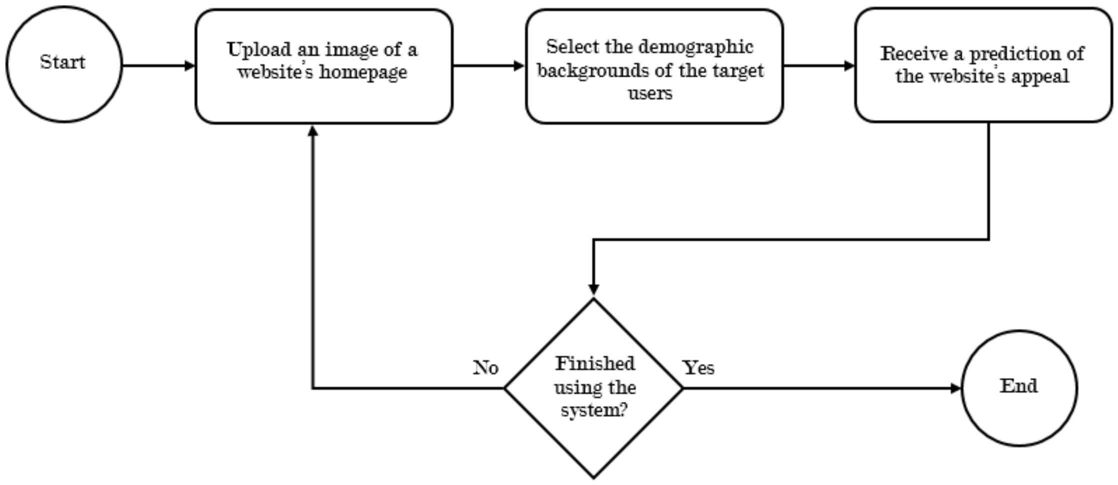

Start
Upload an image of a website's homepage
Select the demographic backgrounds of the target users
Receive a prediction of the website's appeal
Finished using the system?
No
Yes
End

Figure 10: Web Designer Activity Diagram

The figure below shows the activity diagram when preparing the datasets. The AI Expert selects the data to be grouped into training, validation, and test datasets. The AI system will group the data accordingly. The AI system will display the number of data in the training, validation and test datasets.

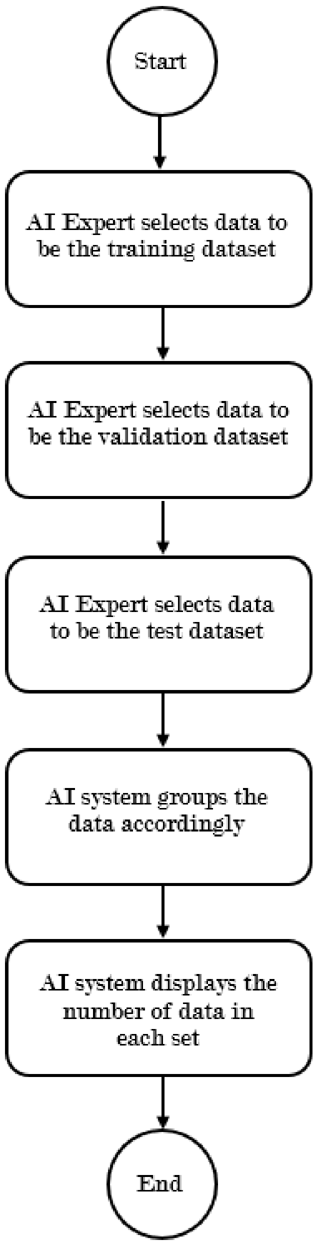

Start
AI Expert selects data to be the training dataset
AI Expert selects data to be the validation dataset
AI Expert selects data to be the test dataset
AI system groups the data accordingly
AI system displays the number of data in each set
End

Figure 11: Activity Diagram for Preparing the Datasets

The figure below shows the activity diagram when creating a hybrid CNN-SVM model. First, the AI Expert selects the dataset to be used for creating and evaluating a hybrid CNN-SVM model. The AI Expert selects a CNN architecture and hyperparameters. After the selection, the AI system trains the CNN with the training and validation datasets while displaying the training and validation error for each epoch. The AI system will output a CNN model. The AI system will create a PCA model by using the training and validation datasets and the CNN model. After creating a PCA model, the AI system displays the reduced dimensions of data. The AI Expert selects the hyperparameters for training the SVM. The AI system will train the SVM using the training and validation datasets, hyperparameters, and the CNN-PCA model. The AI system will display a 2-dimensional plot of the data and separating hyperplane. The AI system will output a hybrid CNN-SVM model.

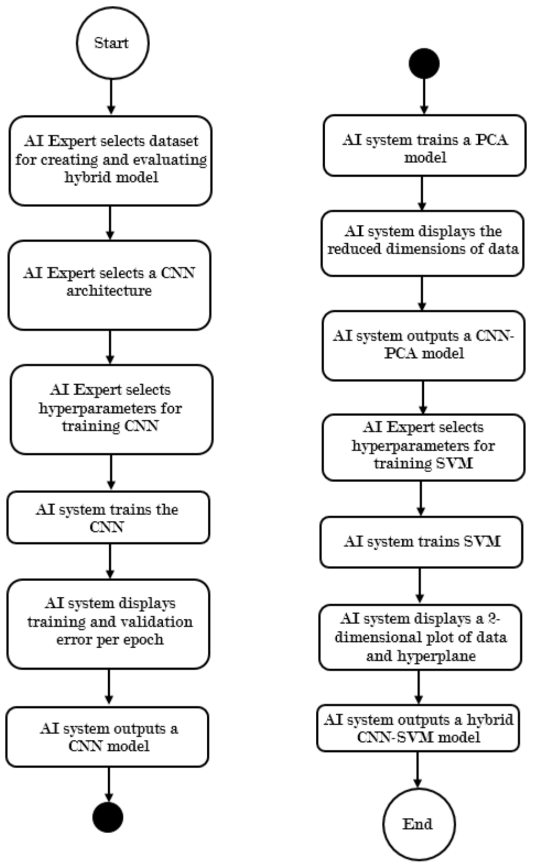

Start
AI Expert selects dataset for creating and evaluating hybrid model
AI Expert selects a CNN architecture
AI Expert selects hyperparameters for training CNN
AI system trains the CNN
AI system displays training and validation error per epoch
AI system outputs a CNN model
AI system trains a PCA model
AI system displays the reduced dimensions of data
AI system outputs a CNN-PCA model
AI Expert selects hyperparameters for training SVM
AI system trains SVM
AI system displays a 2-dimensional plot of data and hyperplane
AI system outputs a hybrid CNN-SVM model
End

Figure 12: Activity Diagram for Creating a Hybrid CNN-SVM Model

The figure below shows the activity diagram when evaluating the performance of the hybrid CNN-SVM model. The AI system will classify the data in the test dataset using the hybrid CNN-SVM model. The AI system will output a log file of the predictions. The AI system will display the accuracy of the prediction.

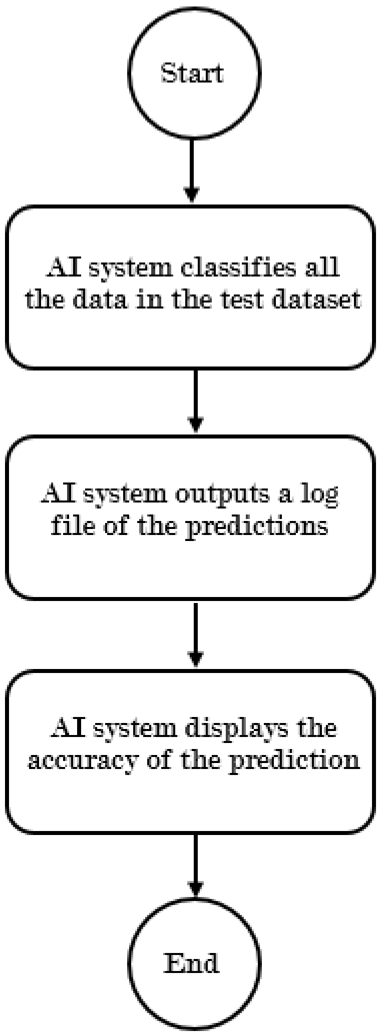

Start
AI system classifies all the data in the test dataset
AI system outputs a log file of the predictions
AI system displays the accuracy of the prediction
End

Figure 13: Activity Diagram for Evaluating the Performance of the Hybrid Model

C. The Dataset

The dataset can be accessed at http://iis.seas.harvard.edu/resources/ [9]. The dataset consists of:

1. 418 screen shots of different homepages of websites. Each having the size of 1024 (width) x 768 (height) pixels.
2. 436,631 visual appeal ratings of the websites. The participants used a scale from 1 to 9, 1 being the least appealing and 9 the most appealing.
3. 5 age groups
    - 12 - 20 years old
    - 21 - 30 years old
    - 31 - 40 years old
    - 41 - 50 years old
    - 51 onwards
4. 2 genders
    - Male
    - Female
5. 33 countries
    - Argentina, Australia, Austria, Belgium, Brazil, Bulgaria, Canada, Chile, Denmark, Finland, France, Germany, Greece, Hungary, India, Ireland, Israel, Italy, Lithuania, The Former Yugoslav Republic Of Macedonia, Mexico, Netherlands, New Zealand, Norway, Poland, Portugal, Romania, Serbia, Singapore, Spain, Sweden, United Kingdom, United States

6. 7 educational levels

   - Pre-high school
   - High school
   - College
   - Professional School
   - Graduate School
   - PhD
   - Postdoctoral

C..1 Data Augmentation

With the small number of images in the dataset, we can artificially enlarge the dataset using label-preserving transformations. Krizhevsky et. al. [13] used two forms of data augmentation in training their AlexNet. The first form of data augmentation is done by extracting random 224 x 224 patches from the images. The second form is altering the intensities of the RGB channels in the training images. With these two forms of data augmentation, we can use them for training our hybrid CNN-SVM.

C..2 Categorical Features

Since the age, gender, country, and educational level are categories, we can use the 1-of-K coding to represent them as features for the SVM [22]. Thus, the categories can be represented as:

- 12 - 20 years old = [1 0 0 0 0]
- 21 - 30 years old = [0 1 0 0 0]
- 31 - 40 years old = [0 0 1 0 0]
- 41 - 50 years old = [0 0 0 1 0]

- 51 onwards = [0 0 0 0 1]
- Male = [1 0]
- Female = [0 1]
- Argentina = [1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]
- United States = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1]
- Pre-high school = [1 0 0 0 0 0 0]
- Postdoctoral = [0 0 0 0 0 0 1]

The same will be applied to the other countries and educational levels.

C..3 Visual Appeal Ratings

Since in the dataset, the visual appeal ratings are in a scale from 1 to 9. We cannot train the model to predict an exact rating because of the limited number of images [2]. To label the images, a rating of 5 and above would mean a good visual appeal. On the other hand, a rating of below 5 would mean a bad visual appeal.

- Visual appeal rating $\geq 5 = 1$
- Visual appeal rating $< 5 = 0$

D. Predicting Method Using Hybrid CNN-SVM

The figure below shows an overview of the predicting method using hybrid CNN-SVM. A screenshot of a website's homepage will be the input for AlexNet. AlexNet would automatically extract the features from the image. Since AlexNet would output a feature vector with a size of 4096, the feature vector would undergo the principal component analysis algorithm. After the PCA, the reduced feature vector along with the age, gender, country, and educational level will be the features that will be used as input for the support vector machine. The SVM would output a prediction of a good or bad visual appeal.

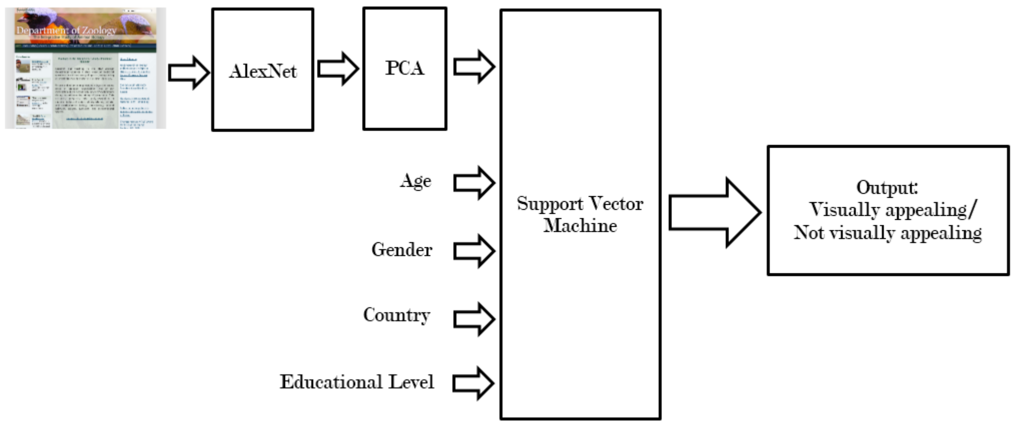


Figure 14: Overview of the Predicting Method using Hybrid CNN-SVM

E. System Architecture

The predicting method that uses a hybrid CNN-SVM model will be implemented in Python using the Keras framework [23] and GPU-accelerated LIBSVM package [24].

1. Software requirements:

    (a) Anaconda with Python 3.5 [25]

    (b) Theano [26]

    (c) Keras [23]

    (d) CUDA [27]

    (e) GPU-accelerated LIBSVM package [24]

    (f) Convnets-keras library [28, 29]

F. Technical Architecture

1. Minimum system requirements:

    (a) Computer with 2 GHz processor or higher

    (b) 4 GB of free disk space

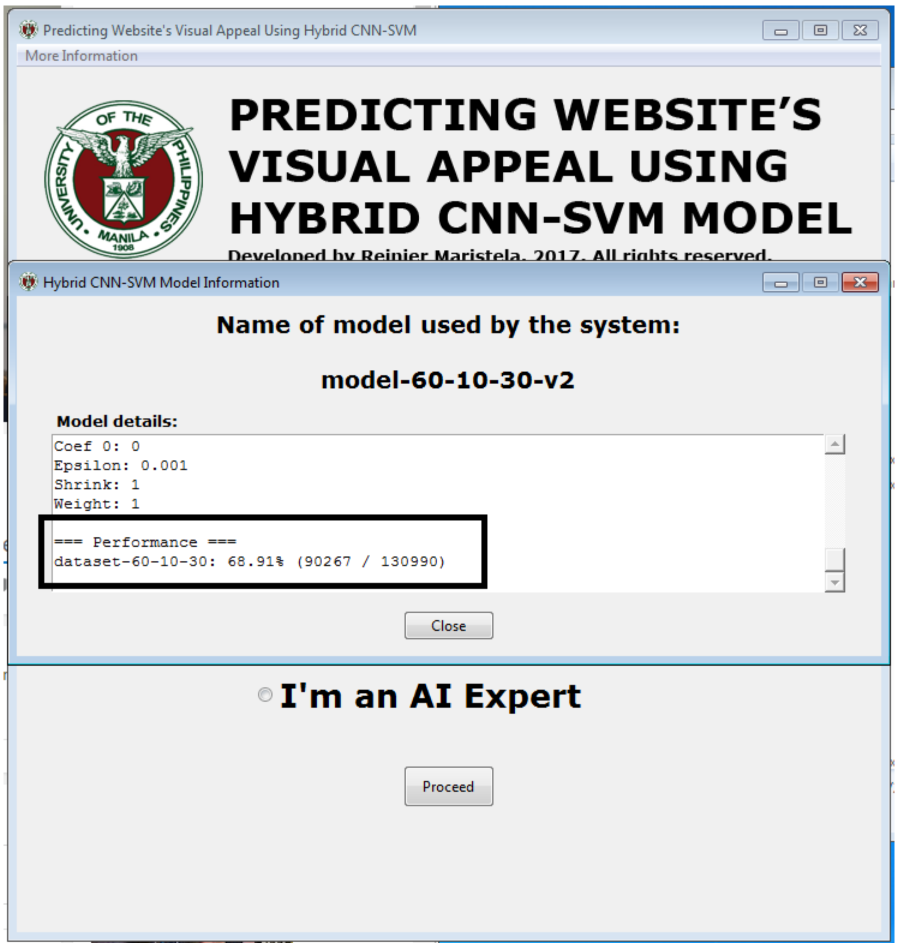
    (c) 4 GB of RAM or higher

    (d) Graphics processing unit is recommended

V. Results

A. Performance of Hybrid Model used by the System

The figure below shows the performance of the current hybrid CNN-SVM model used by the system. As stated earlier, the work of Khani et. al. [2] reported a 34.15% test error. In other words, it has an accuracy of 65.85% with respect to their test set. As we can observe below, the hybrid CNN-SVM model named as model-60-10-30-v2 reported an accuracy of 68.91%. So far, this is the best performing hybrid CNN-SVM model with respect to the different hybrid CNN-SVM models created by the author.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
PREDICTING WEBSITE'S VISUAL APPEAL USING HYBRID CNN-SVM MODEL
Developed by Reinier Maristela, 2017. All rights reserved.
Hybrid CNN-SVM Model Information
Close
I'm an AI Expert
Proceed

Name of model used by the system:

model-60-10-30-v2

Model details:

```
Coef 0: 0
Epsilon: 0.001
Shrink: 1
Weight: 1

=== Performance ===
dataset-60-10-30: 68.91% (90267 / 130990)
```

Figure 15: Performance of the current hybrid CNN-SVM model

B. Welcome Page

This is the first page that any user will see when starting the application. As shown in the figure below, the user is given two choices. If the user is a web designer, he can click the corresponding radio button and click the Proceed button. Likewise, if he is an AI expert, he can click the corresponding radio button and click the Proceed button.

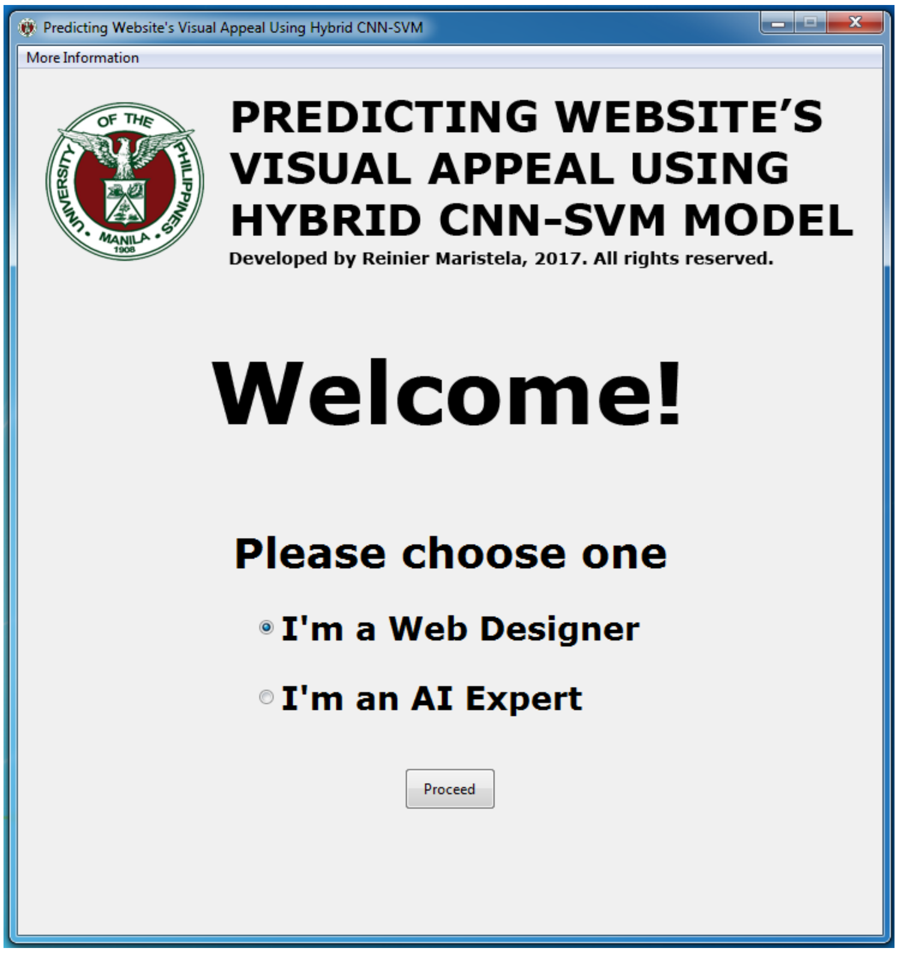
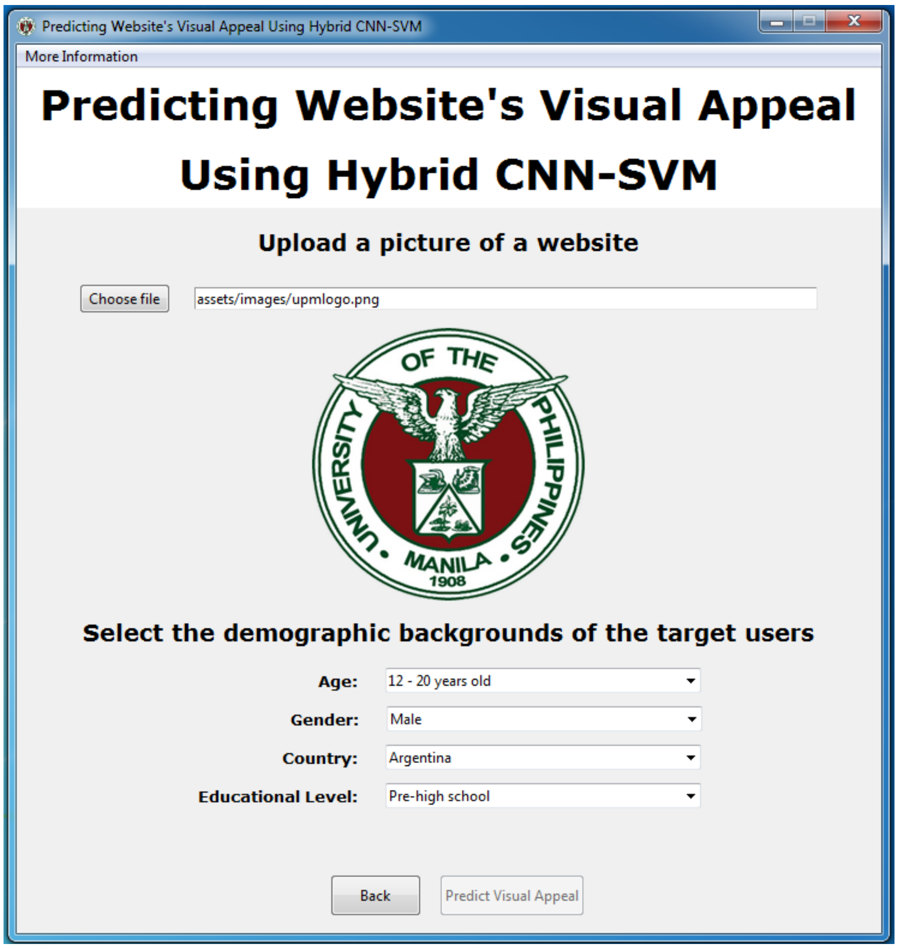

Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
PREDICTING WEBSITE'S VISUAL APPEAL USING HYBRID CNN-SVM MODEL
Developed by Reinier Maristela, 2017. All rights reserved.
Welcome!
Please choose one
I'm a Web Designer
I'm an AI Expert
Proceed

Figure 16: Welcome page of the application

C. Predicting a Website's Visual Appeal

If the user is a web designer, the figure below shows the page where he can predict a website's visual appeal. First, he needs to upload an image to the application. This can be performed by clicking the Choose file button located at the upper left of the page and choosing the appropriate image he wishes to predict. Next, the web designer needs to select the demographic backgrounds of the target users. This can be done by clicking the drop-down lists shown at the lower half of the page. After selecting the desired demographic backgrounds, the web designer can click the Predict Visual Appeal button at the bottom of the page.



Predicting Website's Visual Appeal Using Hybrid CNN-SVM

Upload a picture of a website

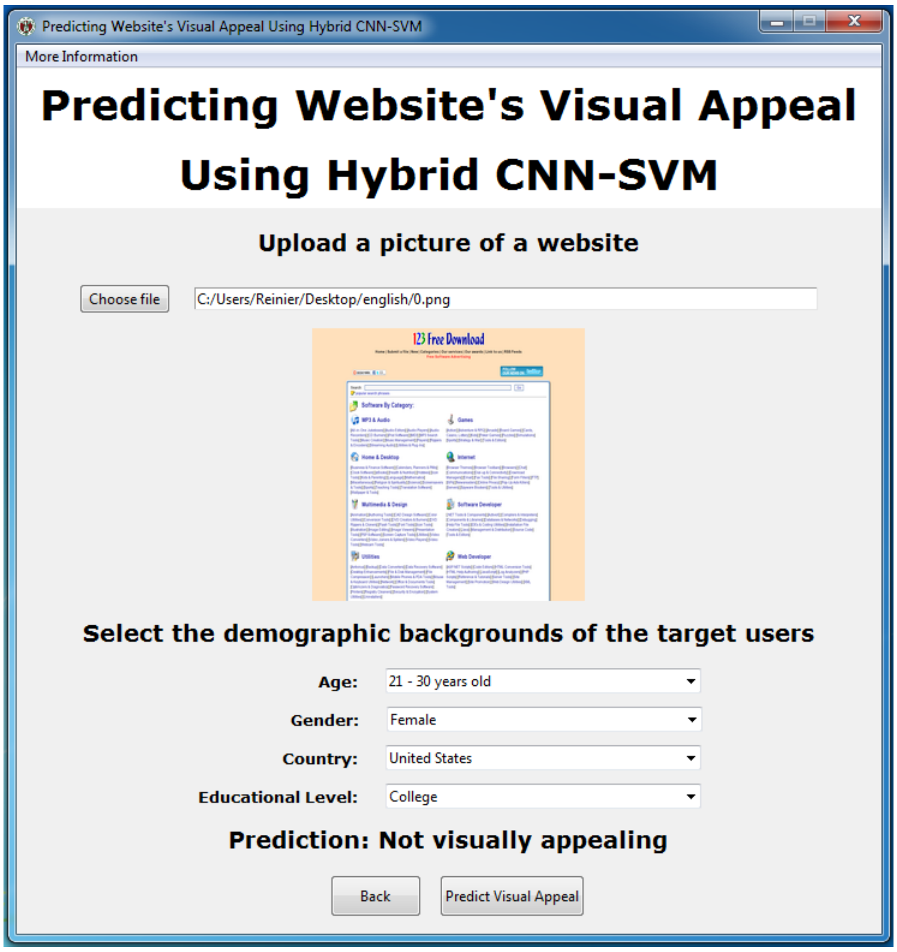
assets/images/upmlogo.png

Select the demographic backgrounds of the target users

Age:	12 - 20 years old
Gender:	Male
Country:	Argentina
Educational Level:	Pre-high school

Figure 17: Page for the Web designer

The figure below is an example on how a web designer will use the application. As we can observe, he uploaded the screenshot of a website that he wishes to predict and supplied the necessary demographic backgrounds. After clicking the Predict Visual Appeal button, the line "Prediction: Not visually appealing" will appear on the page. On the other hand, the line "Prediction: Visually appealing" will appear if that is the prediction of the application.



Predicting Website's Visual Appeal Using Hybrid CNN-SVM

Upload a picture of a website

C:/Users/Reinier/Desktop/english/0.png

Select the demographic backgrounds of the target users

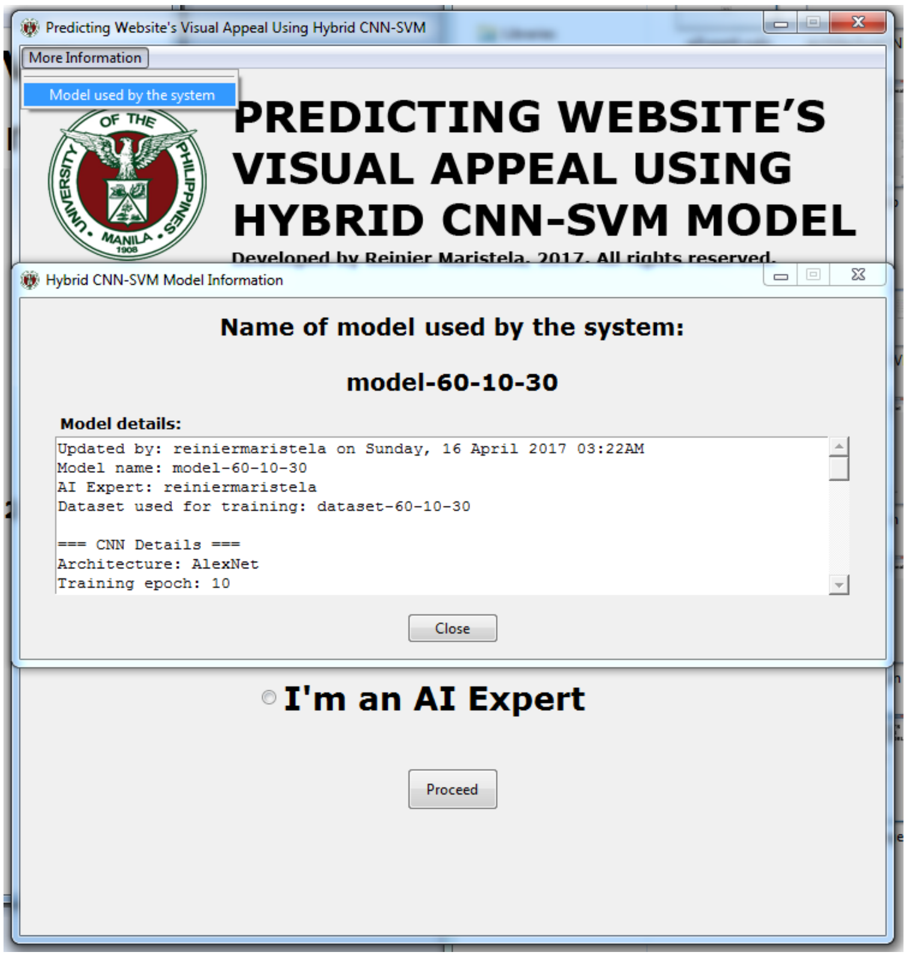
Age:	21 - 30 years old
Gender:	Female
Country:	United States
Educational Level:	College

Prediction: Not visually appealing

Figure 18: Sample prediction

D. Viewing the Hybrid Model that the System uses

Both the web designer and AI expert can view the hybrid CNN-SVM model that the system currently uses. As shown in the figure below, this can be performed by clicking the More Information button at the upper left of the page and clicking the Model used by the system button. A pop-up window will show the name and details of the hybrid CNN-SVM model.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Model used by the system
PREDICTING WEBSITE'S VISUAL APPEAL USING HYBRID CNN-SVM MODEL
Developed by Reinier Maristela, 2017. All rights reserved.
Hybrid CNN-SVM Model Information
Close
I'm an AI Expert
Proceed

Name of model used by the system:

model-60-10-30

Model details:

```
Updated by: reiniermaristela on Sunday, 16 April 2017 03:22AM
Model name: model-60-10-30
AI Expert: reiniermaristela
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
```

Figure 19: Pop-up window showing the details of the hybrid CNN-SVM model

E. AI Expert Log In Page

The remaining sections of this chapter will be for the AI expert. First, the figure below shows the page where the AI expert can input his username and password. There is a need to have a log in page so that not all of the users can use the functionalities exclusively for the AI expert. The functionalities for the AI expert are resource intensive and someone that do not have enough knowledge can disrupt the behavior of the application or the computer itself. If a user wishes to create an account, he can click the Sign Up button located at the bottom part of the page.

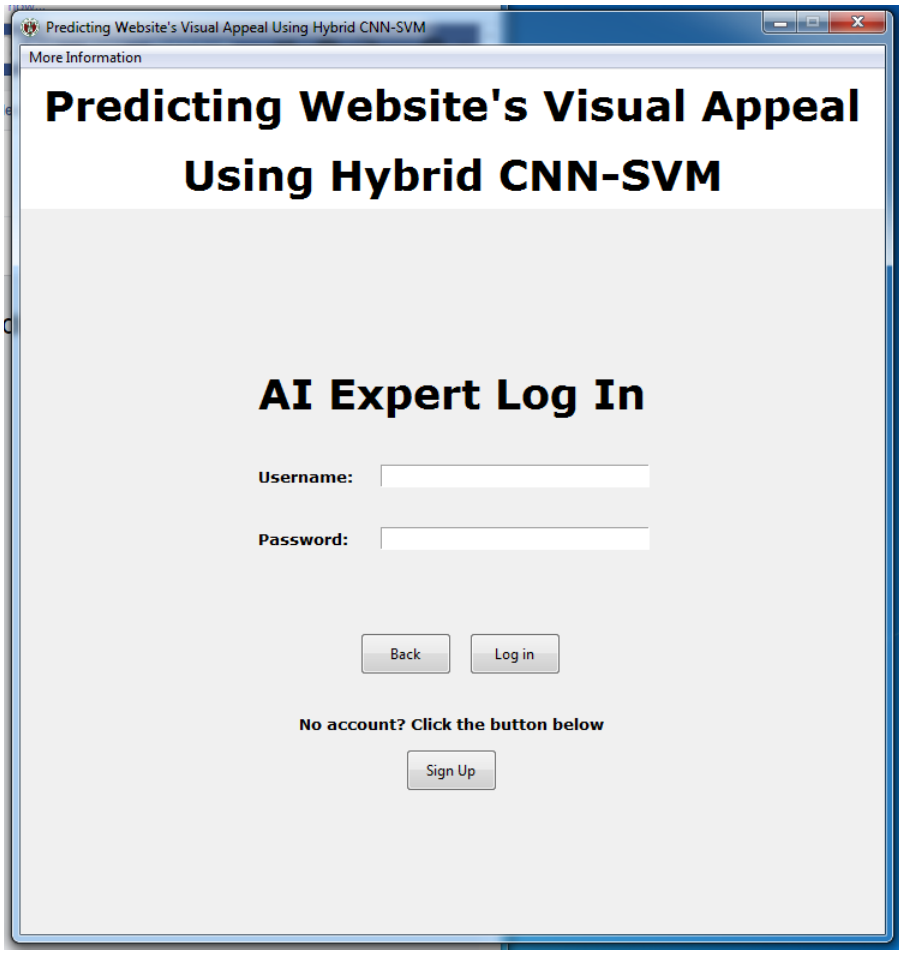
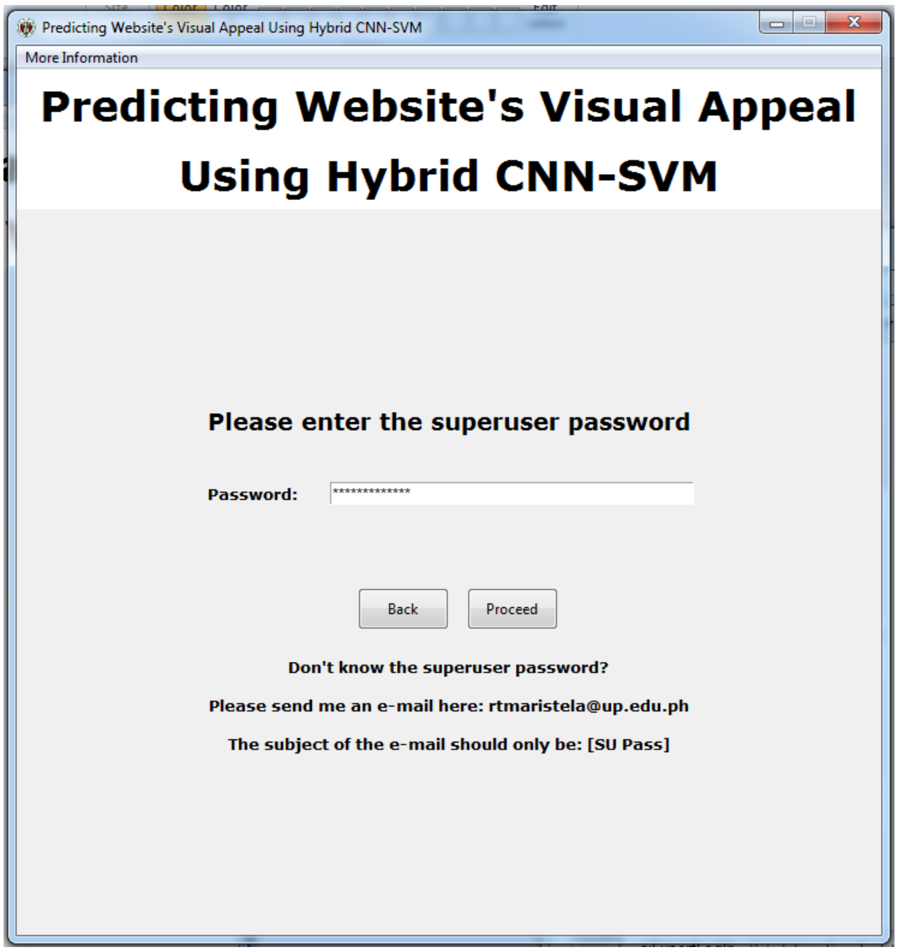

Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Predicting Website's Visual Appeal Using Hybrid CNN-SVM
AI Expert Log In
Username:
Password:
Back
Log in
No account? Click the button below
Sign Up

Figure 20: AI Expert Log In Page

F. Creating an Account

The figure below shows the page where the user needs to input the superuser password. This superuser password can be obtained by sending me an e-mail to the e-mail address listed at the bottom part of the page. After obtaining the superuser password, the user can input the said password to the text box below and click the Proceed button.



Predicting Website's Visual Appeal Using Hybrid CNN-SVM

Please enter the superuser password

Password: *************

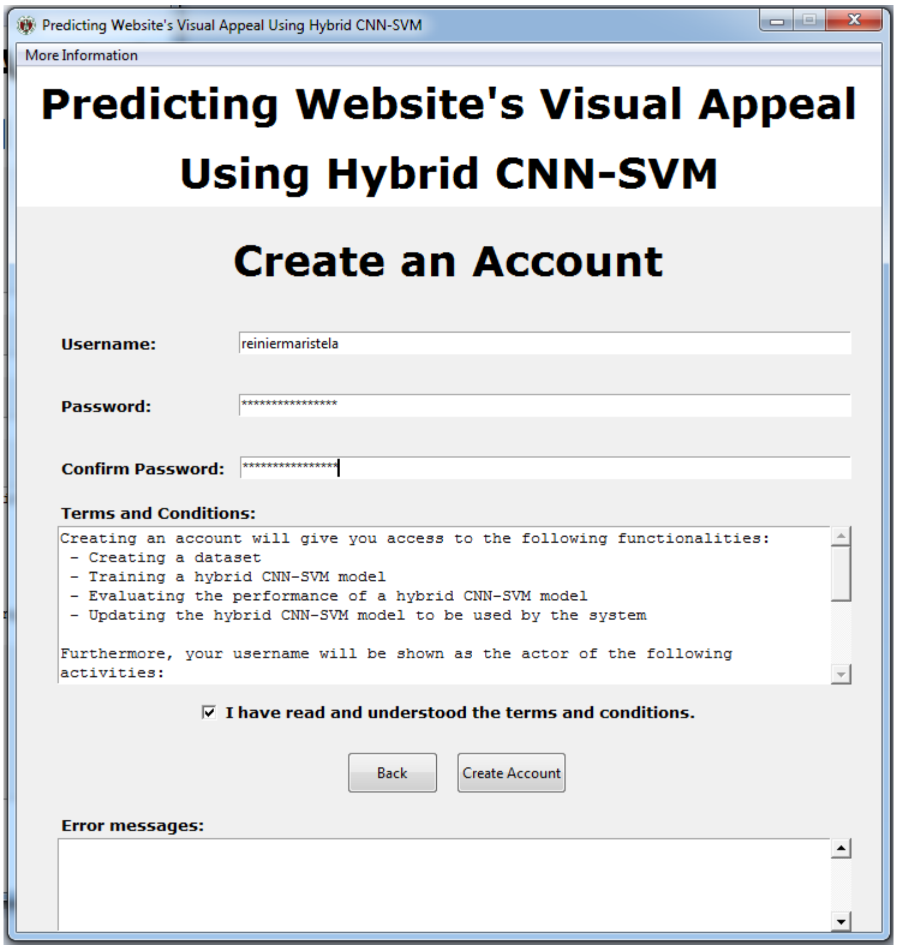
Don't know the superuser password?

Please send me an e-mail here: rtmaristela@up.edu.ph

The subject of the e-mail should only be: [SU Pass]

Figure 21: Typing in the superuser password

The figure below shows the page where the user needs to input all the necessary values to create an account. The user needs to input his username, his password, and agree to the terms and conditions when using the application. Next, the user can click the Create Account button to create his account.



Predicting Website's Visual Appeal Using Hybrid CNN-SVM

Create an Account

Username: reiniermaristela

Password: ****************

Confirm Password: ****************

Terms and Conditions:

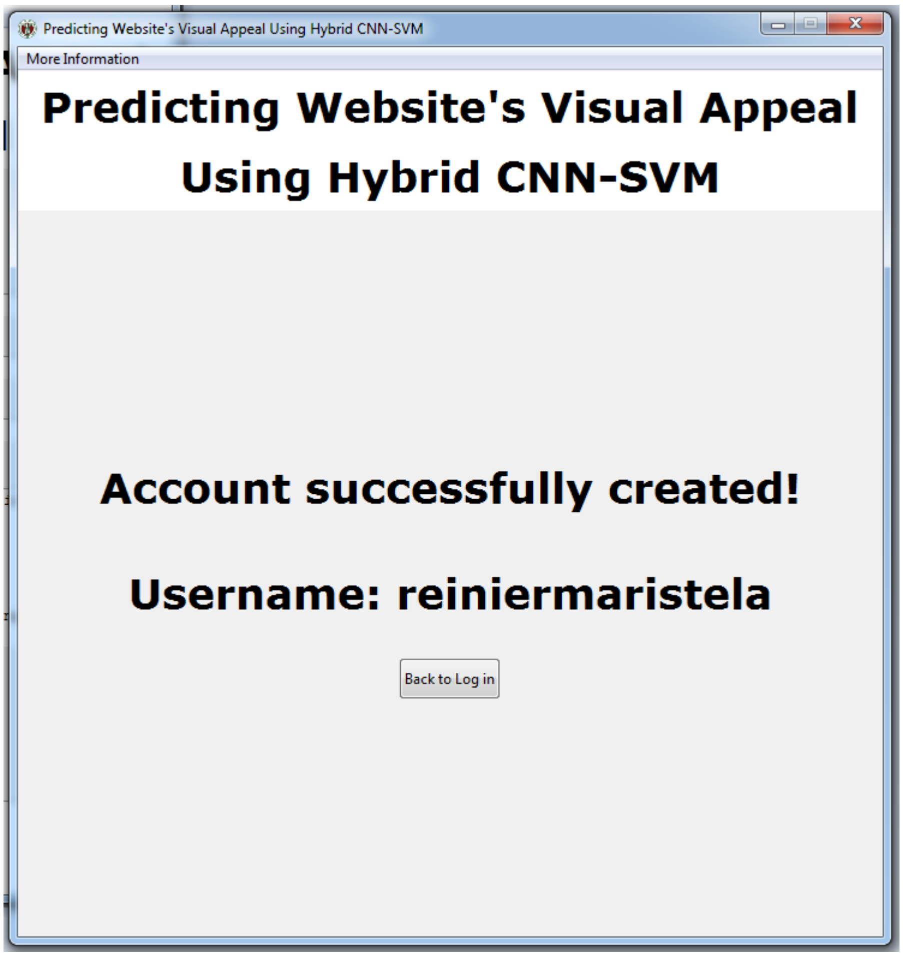
```
Creating an account will give you access to the following functionalities:
 - Creating a dataset
 - Training a hybrid CNN-SVM model
 - Evaluating the performance of a hybrid CNN-SVM model
 - Updating the hybrid CNN-SVM model to be used by the system

Furthermore, your username will be shown as the actor of the following
activities:
```

[✓] I have read and understood the terms and conditions.

Error messages:

Figure 22: Typing in all the necessary values for creating an account

The figure below shows the page where the user can see if his account is created successfully. As we can observe below, the username of the user will be shown to see that it was a success.



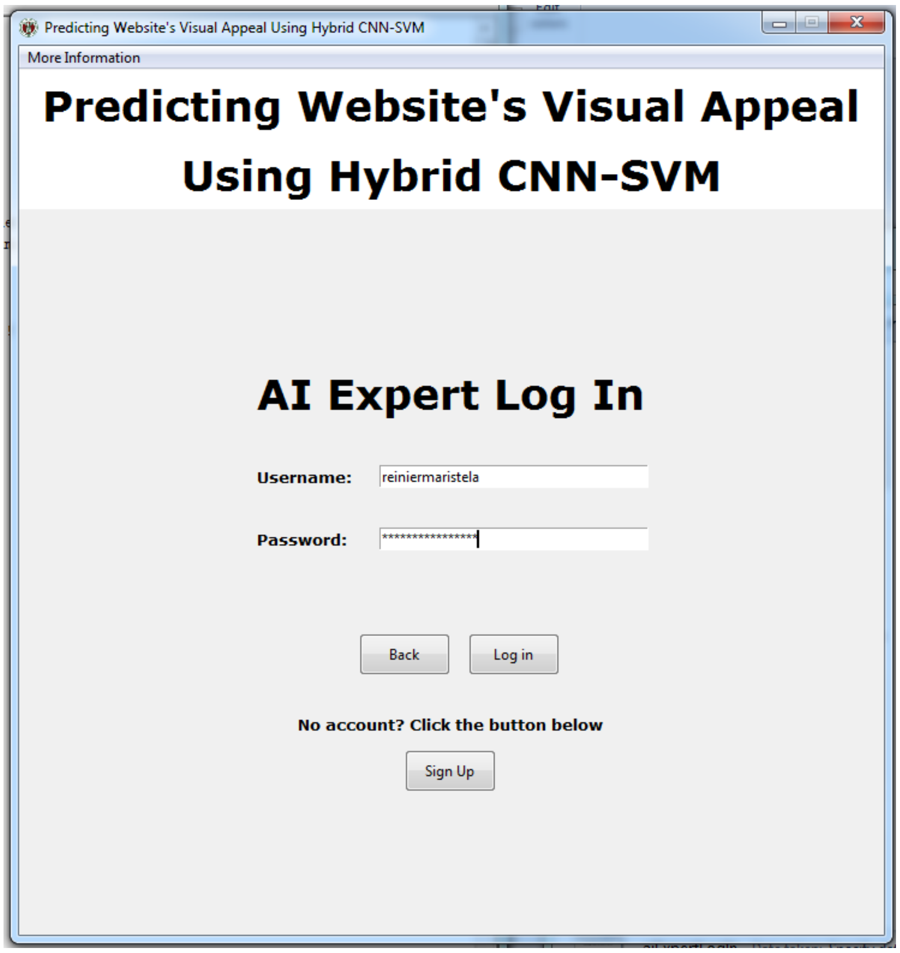
Predicting Website's Visual Appeal Using Hybrid CNN-SVM

Account successfully created!

Username: reiniermaristela

Figure 23: The account was successfully created

Now, the user can log in using his username and password. The figure below shows where to input the username and password. Next, the user can click the Log In button to proceed.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM

More Information

Predicting Website's Visual Appeal Using Hybrid CNN-SVM

AI Expert Log In

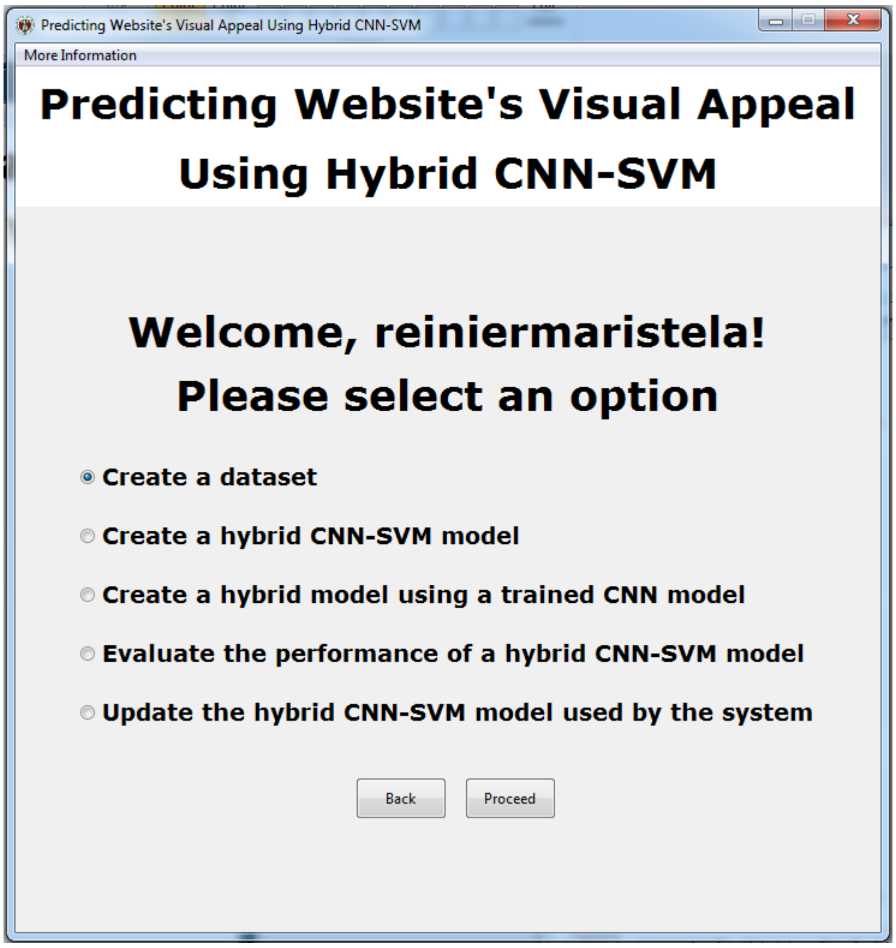
Username: reiniermaristela

Password: ****************

Figure 24: The newly created user can now log in

G. AI Expert Home Page

The figure below shows the home page of the AI expert. The AI expert has five choices to choose from. He can create a dataset, create a hybrid CNN-SVM model, create a hybrid CNN-SVM model from a trained CNN model, evaluate the performane of a hybrid CNN-SVM model, and update the hybrid CNN-SVM model that the system uses. After selecting one of the choices, the AI expert can click the Proceed button to continue.



Predicting Website's Visual Appeal Using Hybrid CNN-SVM

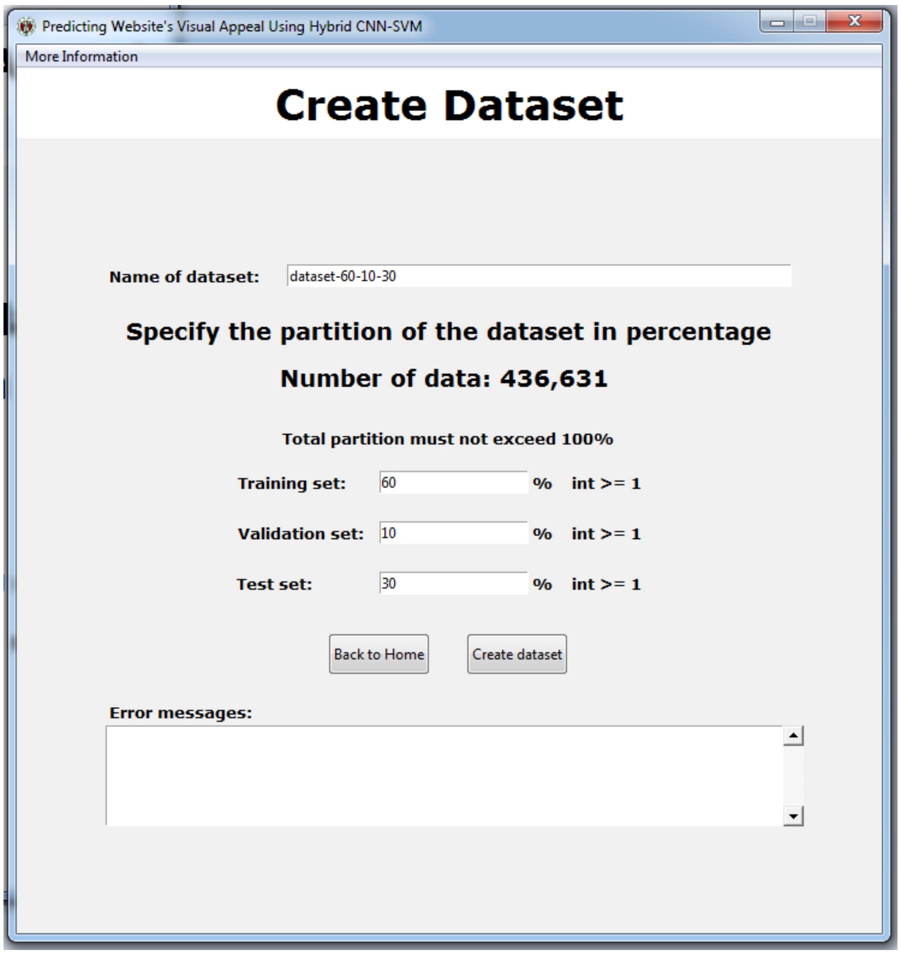
Welcome, reiniermaristela!
Please select an option

[✓ Create a dataset]
Create a hybrid CNN-SVM model
Create a hybrid model using a trained CNN model
Evaluate the performance of a hybrid CNN-SVM model
Update the hybrid CNN-SVM model used by the system

Figure 25: Home page of the AI Expert

H. Creating a dataset

The figure below shows the page when the AI expert chose to create a dataset. As we can observe, the AI expert needs to input the name of the dataset and the partition of the whole dataset into training, validation, and test sets. After typing in the necessary values, the AI expert can click the Create dataset button to proceed.



Create Dataset

Name of dataset: dataset-60-10-30

Specify the partition of the dataset in percentage

Number of data: 436,631

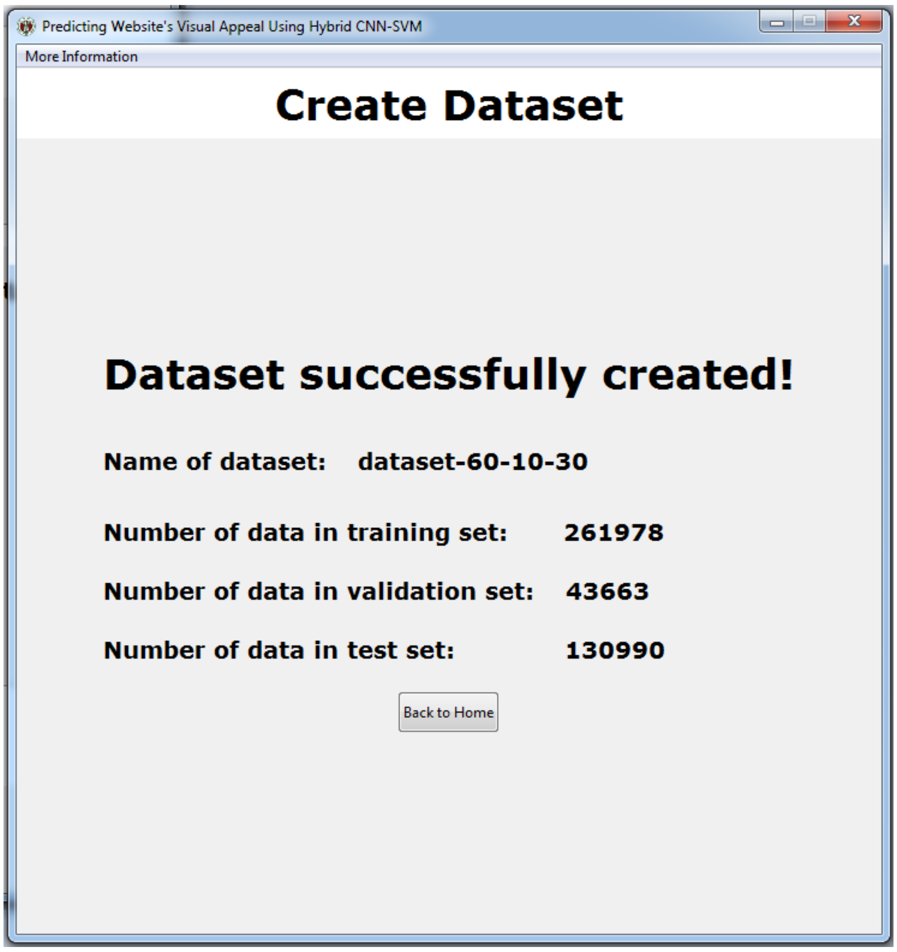
Total partition must not exceed 100%

Training set:	60	%	int >= 1
Validation set:	10	%	int >= 1
Test set:	30	%	int >= 1

Error messages:

Figure 26: Creating a dataset

The figure below shows the page when the dataset was successfully created. As we can observe, the application shows the exact number of data in each of the training, validation, and test sets. Clicking the Back to Home button will bring the AI expert back to the home page.



Dataset successfully created!

Name of dataset: dataset-60-10-30

Number of data in training set: 261978

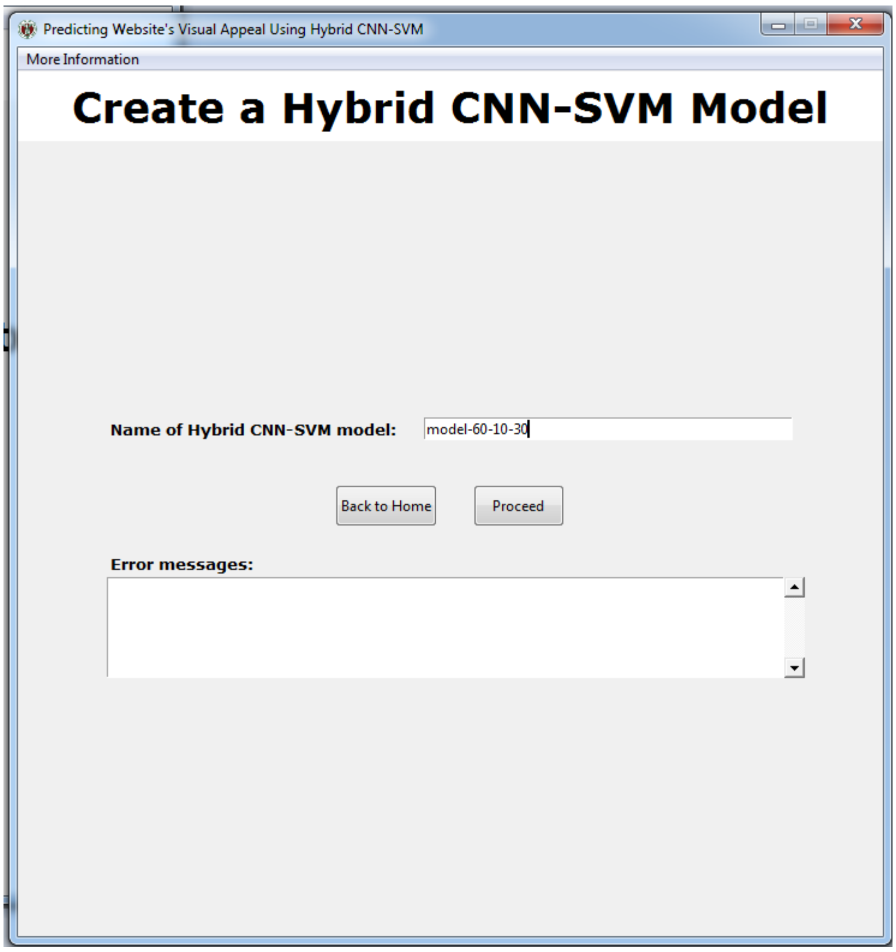
Number of data in validation set: 43663

Number of data in test set: 130990

Figure 27: Dataset successfully created

I. Creating a hybrid CNN-SVM model

The figure below shows the page when the AI expert chose to create hybrid CNN-SVM model. In this page, the AI expert needs to give a name to the hybrid CNN-SVM model he wishes to create. Clicking the Proceed button will bring him to the next step of creating the hybrid CNN-SVM model.



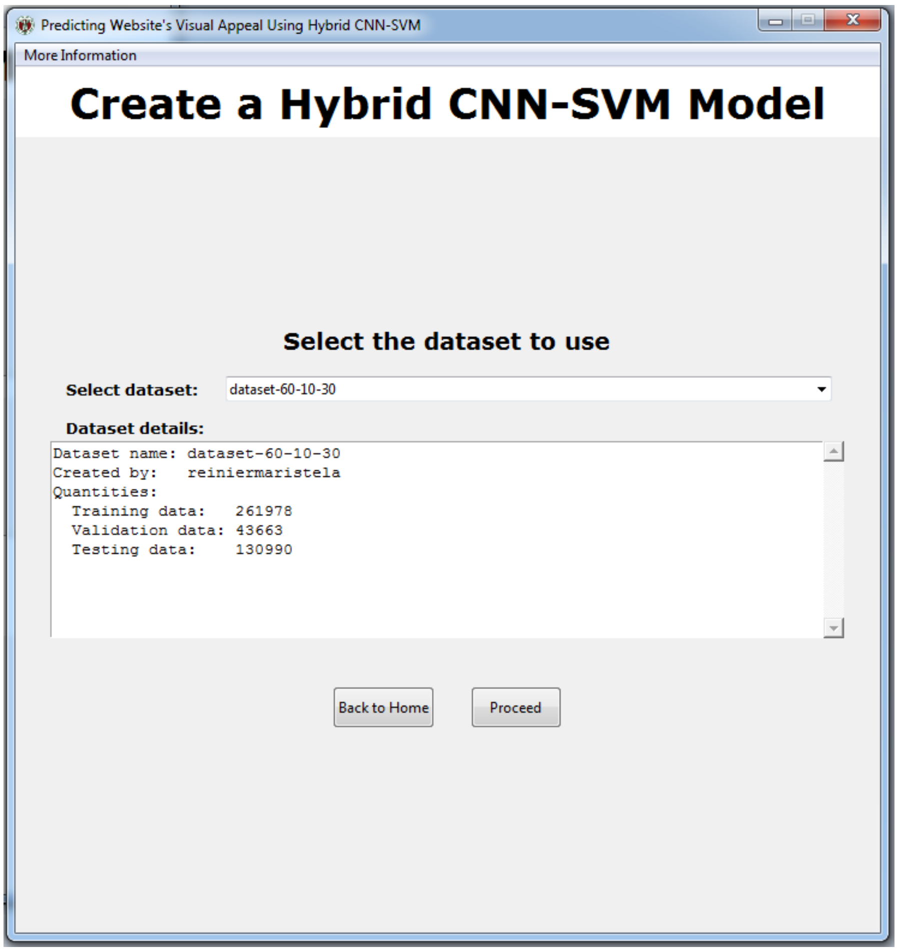
Create a Hybrid CNN-SVM Model

Name of Hybrid CNN-SVM model: model-60-10-30

Error messages:

Figure 28: Naming the hybrid CNN-SVM model

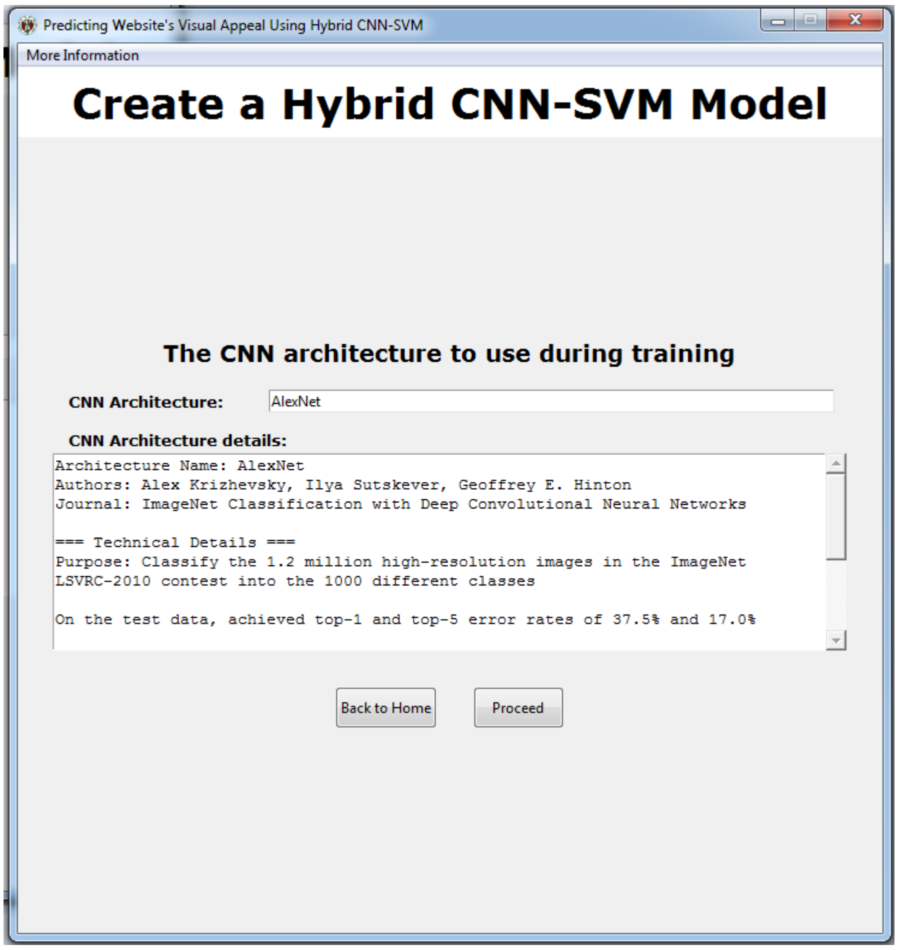
The figure below shows the page where the AI expert will choose what dataset the hybrid CNN-SVM model will be trained. As we can observe, the AI expert chose the dataset we created a while ago. The application will show the details of the selected dataset. Next, the AI expert can click the Proceed button to continue.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
Select the dataset to use
Select dataset:
dataset-60-10-30
Dataset details:
Back to Home
Proceed

```
Dataset name: dataset-60-10-30
Created by:   reiniermaristela
Quantities:
  Training data:   261978
  Validation data: 43663
  Testing data:    130990
```

Figure 29: Selecting a dataset for training

The figure below shows the page where the AI expert can see the details of the architecture of AlexNet. In this page, some of the details of the architecture are shown here. Next, the AI expert can click the Proceed button to continue.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
The CNN architecture to use during training
Back to Home
Proceed

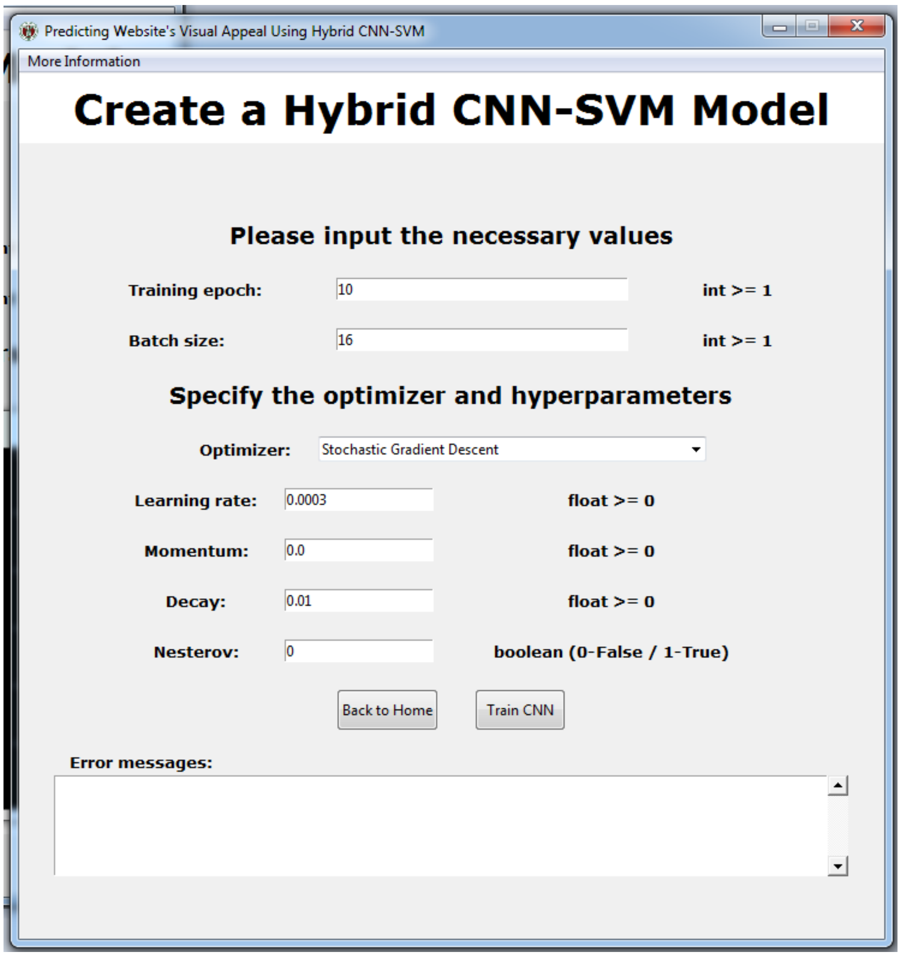
CNN Architecture: AlexNet

CNN Architecture details:

```
Architecture Name: AlexNet
Authors: Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton
Journal: ImageNet Classification with Deep Convolutional Neural Networks

=== Technical Details ===
Purpose: Classify the 1.2 million high-resolution images in the ImageNet
LSVRC-2010 contest into the 1000 different classes

On the test data, achieved top-1 and top-5 error rates of 37.5% and 17.0%
```

Figure 30: AlexNet architecture details

The figure below shows the page where the AI expert can input the hyperparameters for training the CNN model. In this page, the AI expert needs to input the training epoch, batch size, optimizer, and corresponding parameters of the selected optimizer. After supplying the required values, the AI expert can click the Train CNN button to start the training.



Create a Hybrid CNN-SVM Model

Please input the necessary values

Training epoch: 10 int >= 1

Batch size: 16 int >= 1

Specify the optimizer and hyperparameters

Optimizer: Stochastic Gradient Descent

Learning rate: 0.0003 float >= 0

Momentum: 0.0 float >= 0

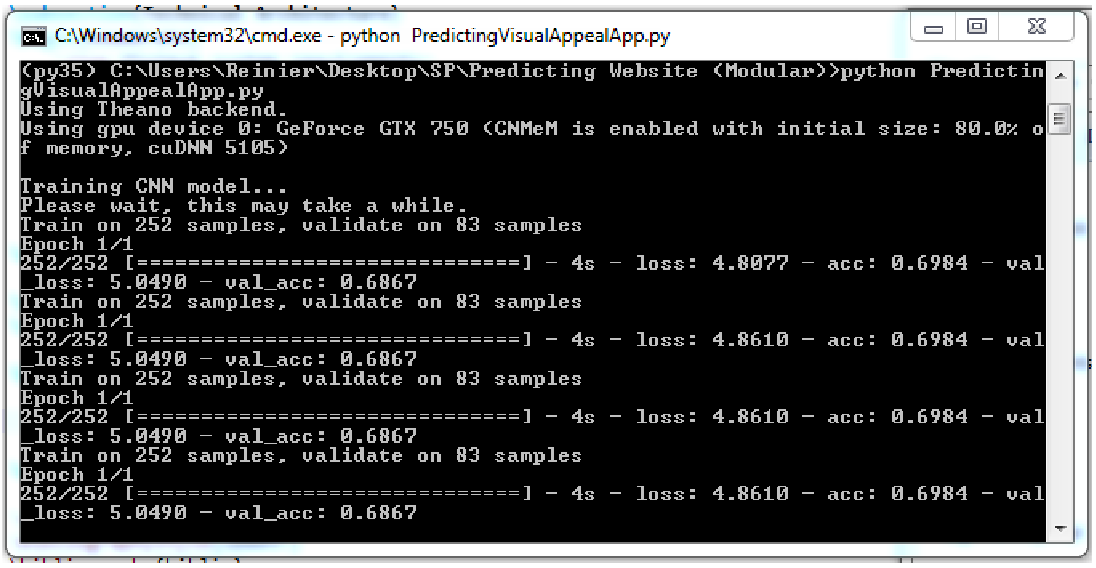
Decay: 0.01 float >= 0

Nesterov: 0 boolean (0-False / 1-True)

Error messages:

Figure 31: Hyperparameters for training the CNN model

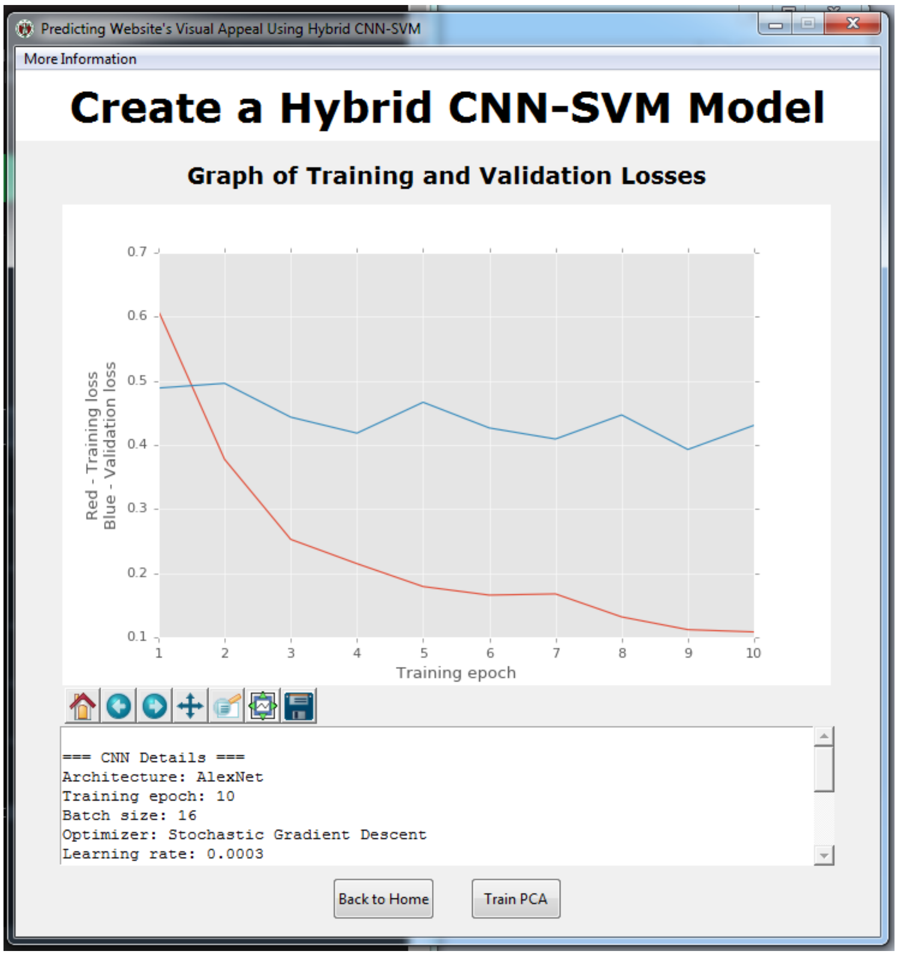
The figure below shows what the console outputs during the training of the CNN model. As we can observe, the console outputs the number of training samples and validate samples. Also, the training losses, validation losses, training accuracy, and validation accuracy are also shown. Please note that this process may take a long time depending on the computational power of your hardware.


C:\Windows\system32\cmd.exe - python PredictingVisualAppealApp.py

```
(py35) C:\Users\Reinier\Desktop\SP\Predicting Website (Modular)>python Predictin
gVisualAppealApp.py
Using Theano backend.
Using gpu device 0: GeForce GTX 750 (CNMeM is enabled with initial size: 80.0% o
f memory, cuDNN 5105)

Training CNN model...
Please wait, this may take a while.
Train on 252 samples, validate on 83 samples
Epoch 1/1
252/252 [==============================] - 4s - loss: 4.8077 - acc: 0.6984 - val
_loss: 5.0490 - val_acc: 0.6867
Train on 252 samples, validate on 83 samples
Epoch 1/1
252/252 [==============================] - 4s - loss: 4.8610 - acc: 0.6984 - val
_loss: 5.0490 - val_acc: 0.6867
Train on 252 samples, validate on 83 samples
Epoch 1/1
252/252 [==============================] - 4s - loss: 4.8610 - acc: 0.6984 - val
_loss: 5.0490 - val_acc: 0.6867
Train on 252 samples, validate on 83 samples
Epoch 1/1
252/252 [==============================] - 4s - loss: 4.8610 - acc: 0.6984 - val
_loss: 5.0490 - val_acc: 0.6867
```

Figure 32: Console output during training

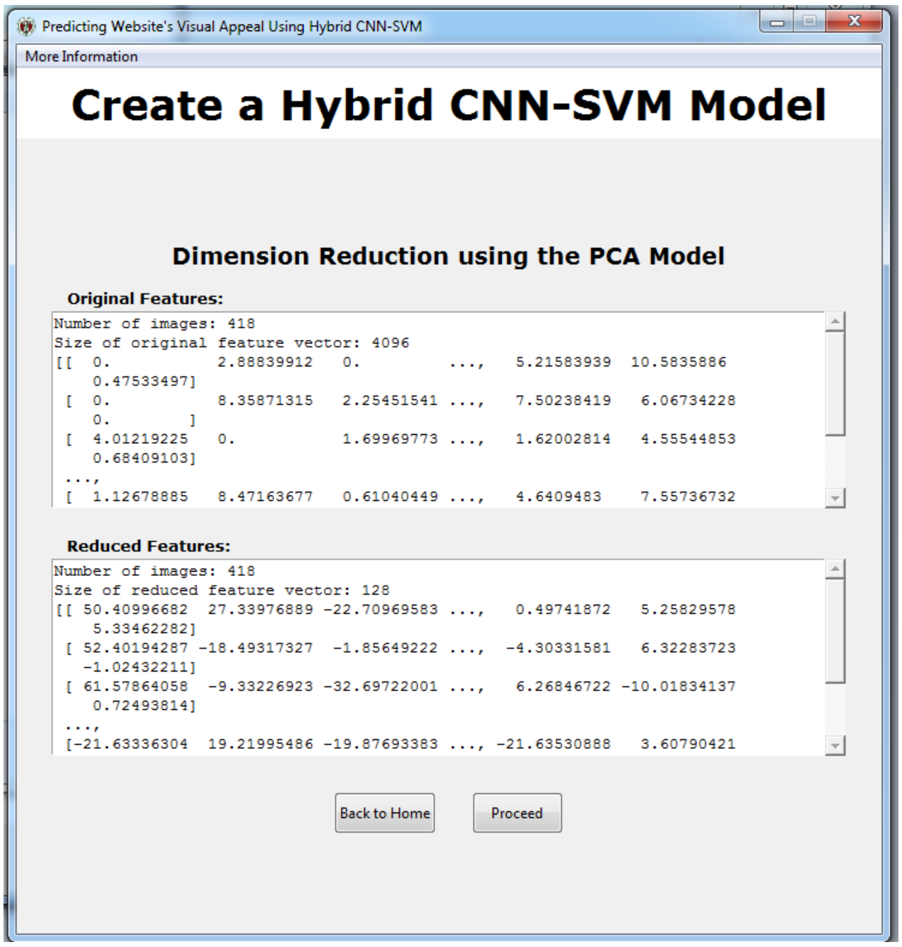
After training the CNN model, the figure below shows what the application will output to the page. As we can observe, the training and validation losses are represented as a graph for a better understanding of the training process. Also, other important details are shown in the box located at the lower half of the page. Next, the AI expert can click the Train PCA to continue.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
Graph of Training and Validation Losses
Red - Training loss
Blue - Validation loss
Training epoch
Back to Home
Train PCA

```
=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
Batch size: 16
Optimizer: Stochastic Gradient Descent
Learning rate: 0.0003
```

Figure 33: CNN model trained successfully

The figure below shows the dimension reduction performed by the PCA model. In this page, there are two boxes showing different data. The first box located at the upper half of the page are the 4096 feature vectors of the website images. In the second box located at the lower half of the page are the reduced 128 feature vectors of the website images. Next, the AI expert can click the Proceed button to continue.

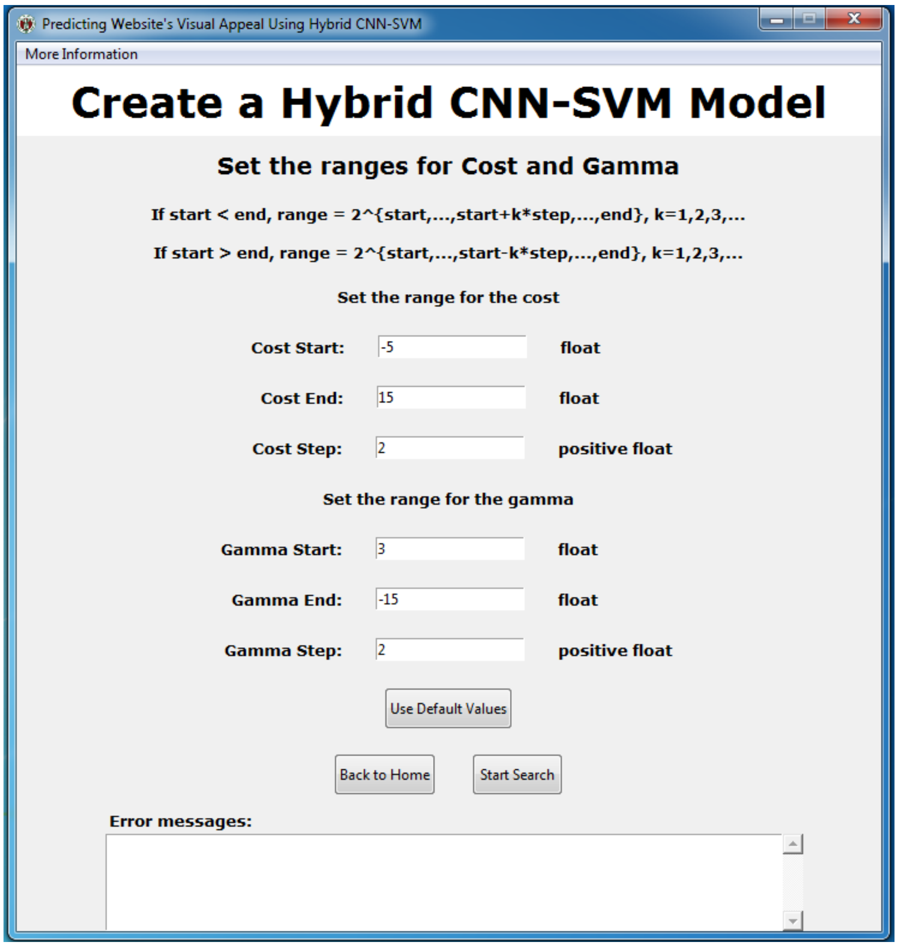

Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
Dimension Reduction using the PCA Model
Original Features:
Reduced Features:
Back to Home
Proceed

```
Number of images: 418
Size of original feature vector: 4096
[[  0.          2.88839912   0.         ...,   5.21583939  10.5835886
    0.47533497]
 [  0.          8.35871315   2.25451541 ...,   7.50238419   6.06734228
    0.        ]
 [  4.01219225   0.          1.69969773 ...,   1.62002814   4.55544853
    0.68409103]
 ...,
 [  1.12678885   8.47163677   0.61040449 ...,   4.6409483    7.55736732
```

```
Number of images: 418
Size of reduced feature vector: 128
[[ 50.40996682  27.33976889 -22.70969583 ...,   0.49741872   5.25829578
    5.33462282]
 [ 52.40194287 -18.49317327  -1.85649222 ...,  -4.30331581   6.32283723
   -1.02432211]
 [ 61.57864058  -9.33226923 -32.69722001 ...,   6.26846722 -10.01834137
    0.72493814]
 ...,
 [-21.63336304  19.21995486 -19.87693383 ..., -21.63530888   3.60790421
```

Figure 34: Dimension reduction by PCA model

The figure below shows the page where the AI expert can input the hyper-parameters for training the SVM. In this page, the AI expert needs to input the start, end, and step of the ranges of the cost and gamma. These two ranges are needed for the cross validation of the SVM model. After clicking the Start Search button, the application will now find the best cost and gamma pair for training the SVM model.



Create a Hybrid CNN-SVM Model

Set the ranges for Cost and Gamma

If start < end, range = 2^{start,...,start+k*step,...,end}, k=1,2,3,...

If start > end, range = 2^{start,...,start-k*step,...,end}, k=1,2,3,...

Set the range for the cost

Cost Start:	-5	float
Cost End:	15	float
Cost Step:	2	positive float

Set the range for the gamma

Gamma Start:	3	float
Gamma End:	-15	float
Gamma Step:	2	positive float

Error messages:

Figure 35: Setting the ranges for cost and gamma

The figure below shows the graph where the application tries to find the best cost and gamma pair for training the SVM model. In this graph, the most important values are labeled. The best cost and gamma pair are located above the graph alongside the accuracy. These values are updated everytime there is a new best cost and gamma pair.

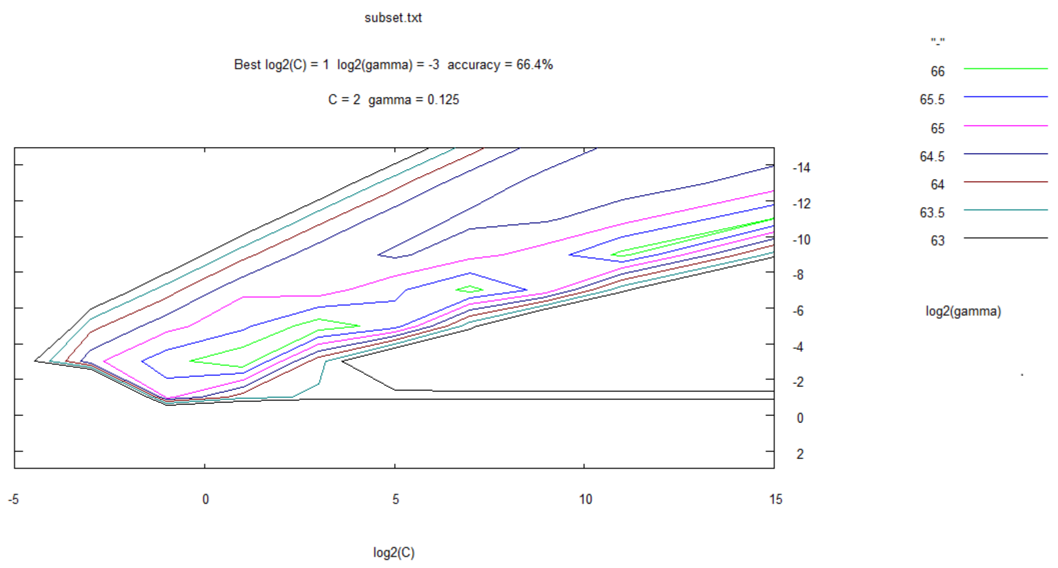
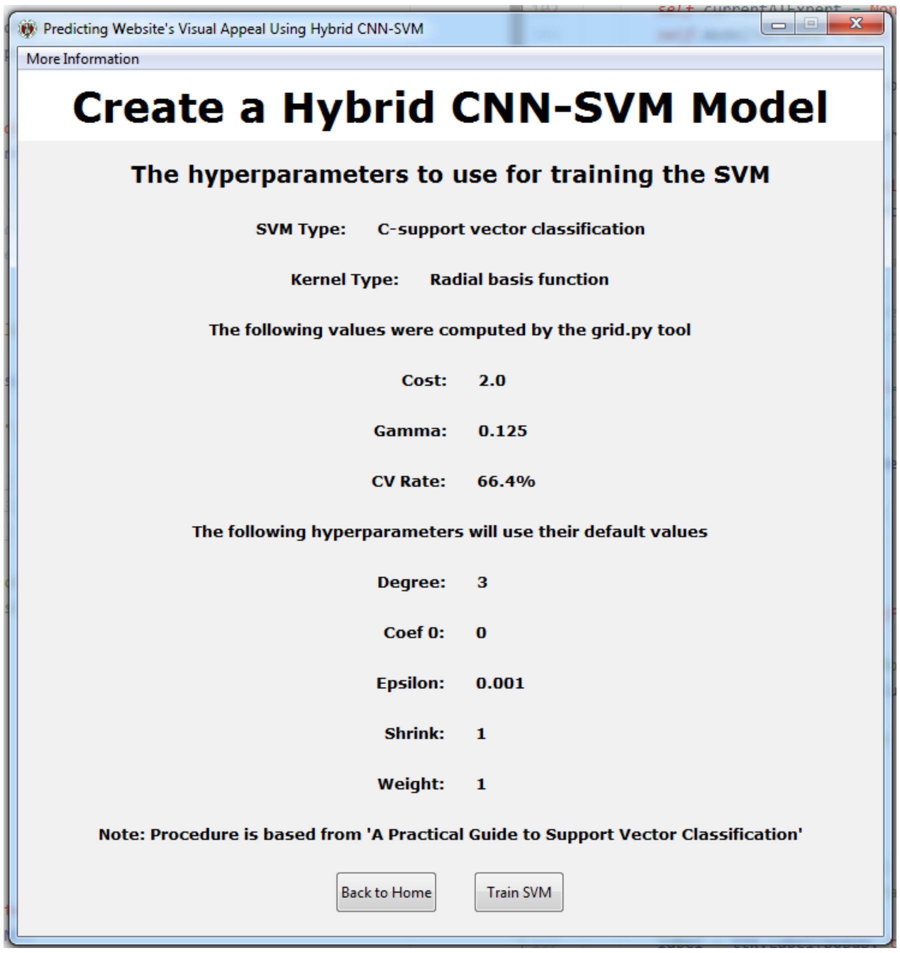


Figure 36: Finding the best cost and gamma pair

The figure below shows the page where the AI expert can see the hyperparameters to use for training the SVM model. As we can observe, the best cost and gamma pair are the ones to be used for training the SVM model. All other hyperparameters will be in their default values. Next, the AI expert can click the Train SVM model to continue.



Create a Hybrid CNN-SVM Model

The hyperparameters to use for training the SVM

SVM Type: C-support vector classification

Kernel Type: Radial basis function

The following values were computed by the grid.py tool

Cost:	2.0
Gamma:	0.125
CV Rate:	66.4%

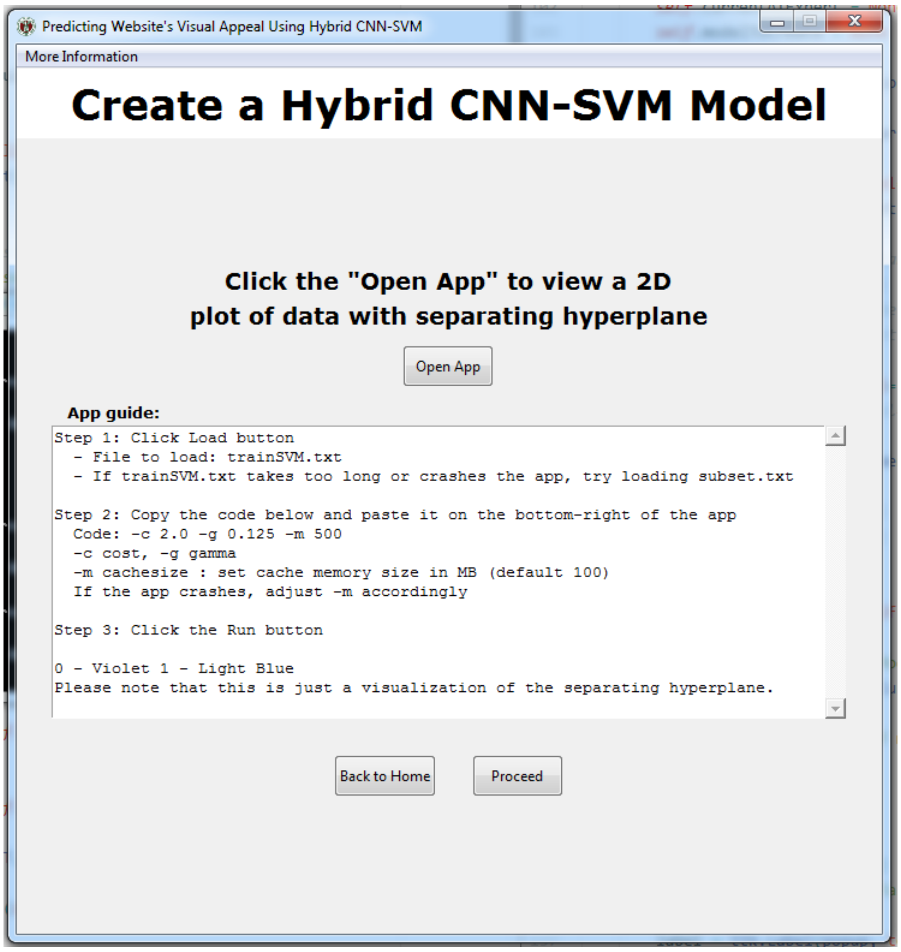
The following hyperparameters will use their default values

Degree:	3
Coef 0:	0
Epsilon:	0.001
Shrink:	1
Weight:	1

Note: Procedure is based from 'A Practical Guide to Support Vector Classification'

Figure 37: Hyperparameters for training SVM model

The figure below shows the page where the AI expert can use the SVM Toy tool provided by Athanasopoulos et. al. [24]. The SVM Toy tool is a simple application to visualize the separating hyperplane of the SVM model in 2-dimensions. The AI expert can click the Open App button to open SVM Toy.



Create a Hybrid CNN-SVM Model

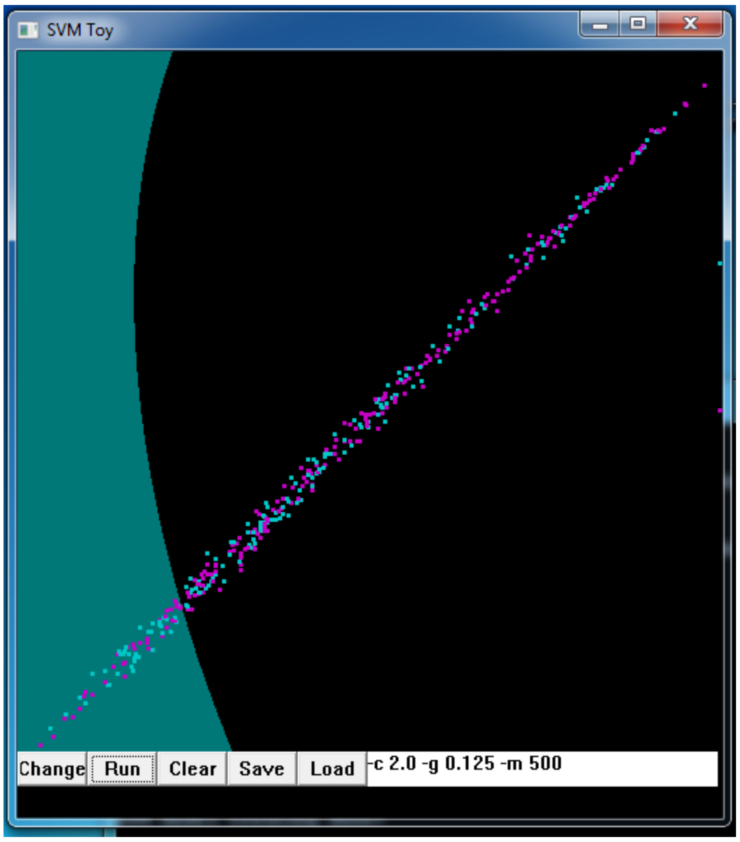
Click the "Open App" to view a 2D plot of data with separating hyperplane

App guide:

```
Step 1: Click Load button
  - File to load: trainSVM.txt
  - If trainSVM.txt takes too long or crashes the app, try loading subset.txt

Step 2: Copy the code below and paste it on the bottom-right of the app
  Code: -c 2.0 -g 0.125 -m 500
  -c cost, -g gamma
  -m cachesize : set cache memory size in MB (default 100)
  If the app crashes, adjust -m accordingly

Step 3: Click the Run button

0 - Violet 1 - Light Blue
Please note that this is just a visualization of the separating hyperplane.
```

Figure 38: SVM Toy guide

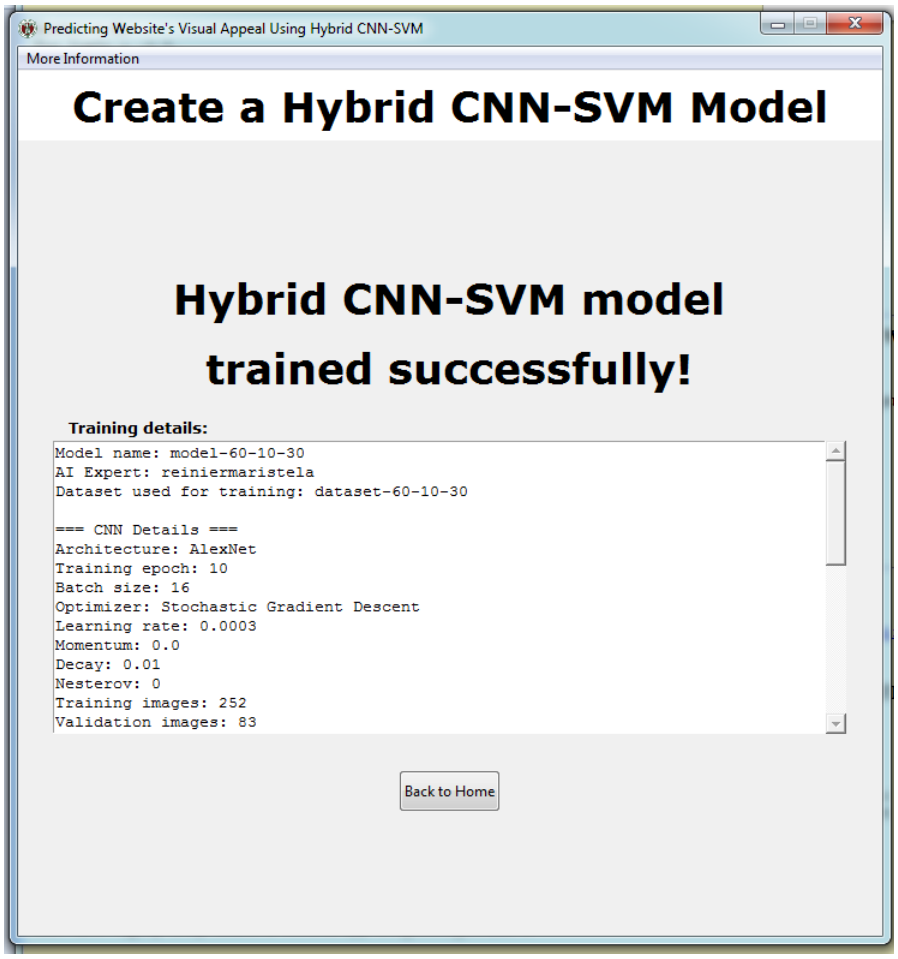
By following the guides written at the center of the page, the AI expert can now see the separating hyperplane. The figure below shows an example of the separating hyperplane in 2-dimensions. The AI expert can click the Proceed button to continue.



-c 2.0 -g 0.125 -m 500

Figure 39: Separating hyperplane visualization

The figure below shows the page where the details of the training are shown. The name of the model, AI expert performing the training, dataset used for training, CNN model training, PCA model training, and SVM model training are shown in the center of the page. This signifies that the training was a success. The AI expert can click the Back to Home button to return to the home page.



Create a Hybrid CNN-SVM Model

Hybrid CNN-SVM model trained successfully!

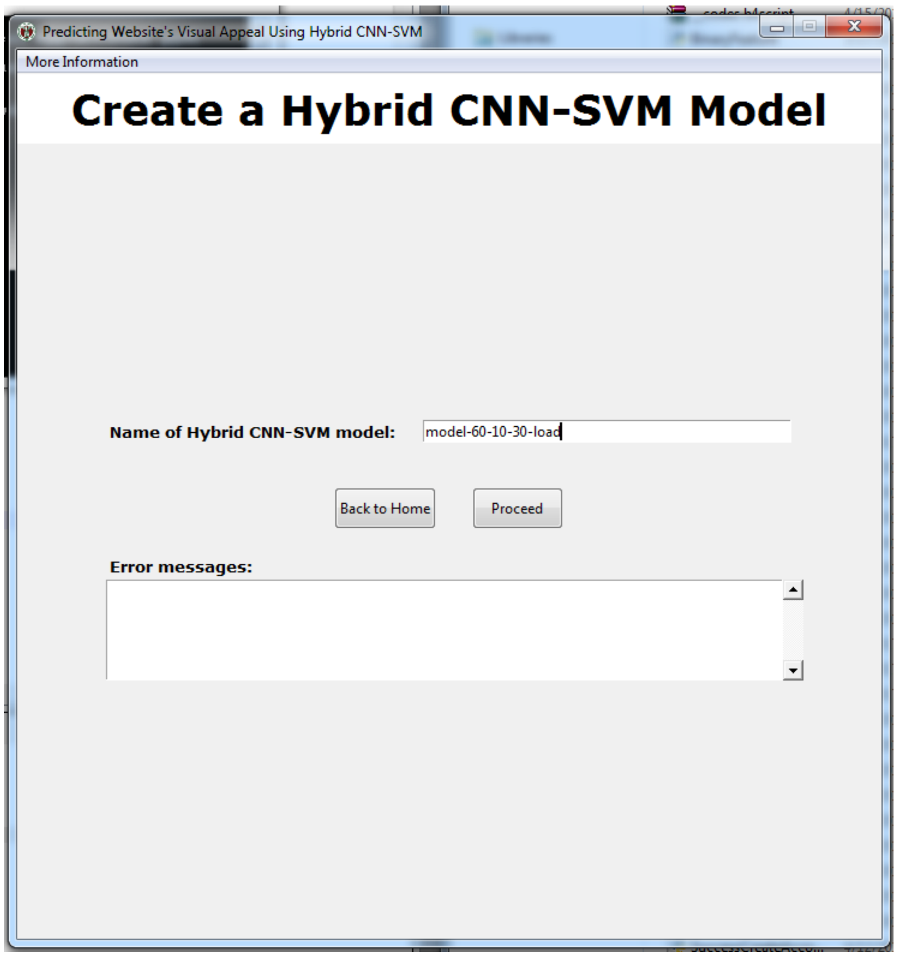
Training details:

```
Model name: model-60-10-30
AI Expert: reiniermaristela
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
Batch size: 16
Optimizer: Stochastic Gradient Descent
Learning rate: 0.0003
Momentum: 0.0
Decay: 0.01
Nesterov: 0
Training images: 252
Validation images: 83
```

Figure 40: Hybrid CNN-SVM model trained successfully

J. Creating a hybrid model using a trained CNN model

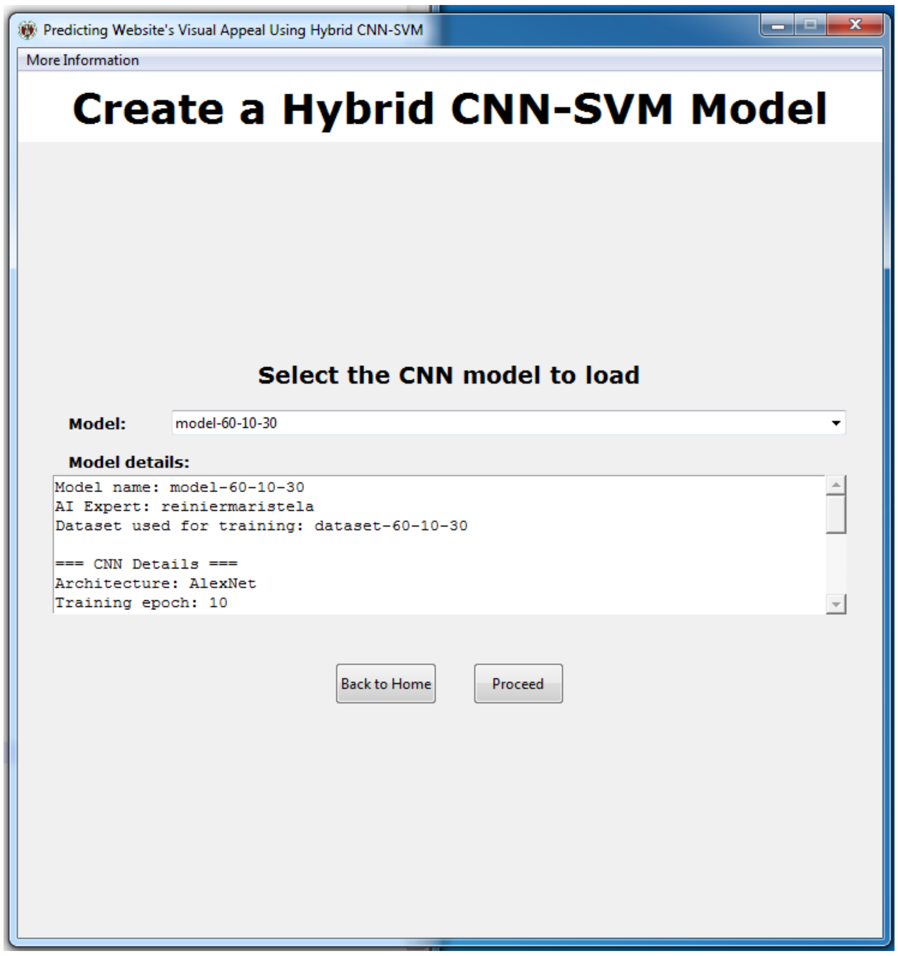
The figure below shows the page when the AI expert chose to create hybrid CNN-SVM model using a trained CNN model. In this page, the AI expert needs to give a name to the hybrid CNN-SVM model he wishes to create. Clicking the Proceed button will bring him to the next step of creating the hybrid CNN-SVM model.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
Back to Home
Proceed
Error messages:

Name of Hybrid CNN-SVM model: model-60-10-30-load

Figure 41: Naming the hybrid CNN-SVM model

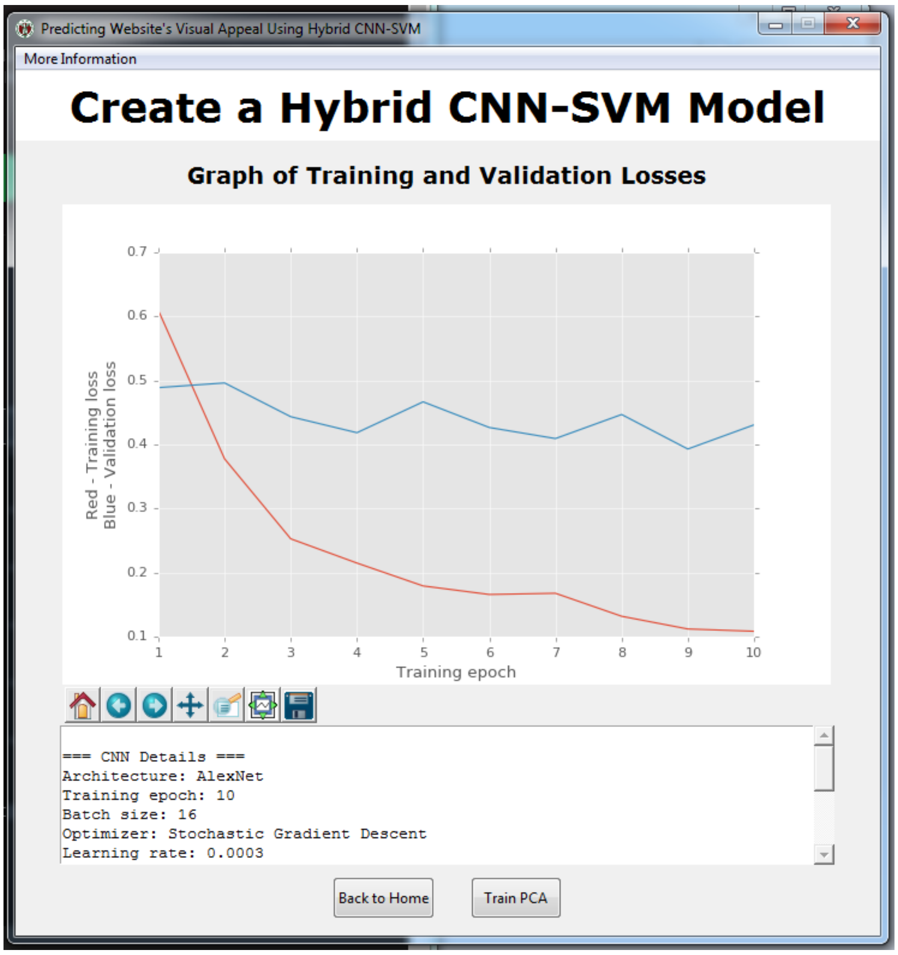
The figure below shows the page where the AI expert can select a trained CNN model to load. This can be performed by clicking the drop-down list located at the center of the page. The details of the trained CNN model will be shown at the box below. Next, the AI expert can click the Proceed button to load the selected model.


Predicting Website's Visual Appeal Using Hybrid CNN-SVM
More Information
Create a Hybrid CNN-SVM Model
Select the CNN model to load
Model:
model-60-10-30
Model details:
Back to Home
Proceed

```
Model name: model-60-10-30
AI Expert: reiniermaristela
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
```

Figure 42: Selecting a trained CNN model to load

The figure below shows the page where the details of the CNN training are shown. As we can observe, this is the same page as the page in the section above. From this page onwards, please refer to the section above for the complete details of creating a hybrid CNN-SVM model.

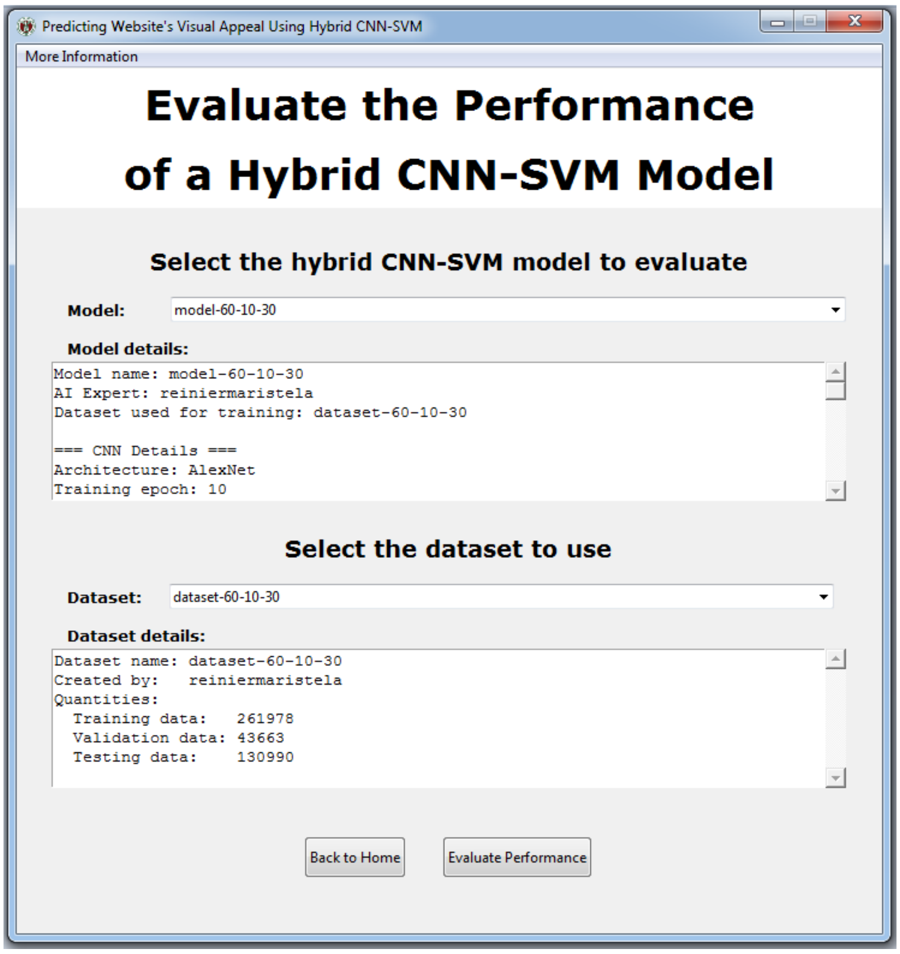


```
=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
Batch size: 16
Optimizer: Stochastic Gradient Descent
Learning rate: 0.0003
```

Figure 43: Trained CNN model loaded successfully

K. Evaluating the performance of the hybrid CNN-SVM model

The figure below shows the page when the AI expert chose to evaluate the performance of the hybrid CNN-SVM model. The AI expert selects the model to evaluate by clicking the drop-down list. The details of the model will be shown at the box below it. Next, the AI expert selects the dataset to use for testing the hybrid CNN-SVM model. This can be performed by clicking the drop-down list and the details will be shown below. Next, the AI expert can click the Evaluate Performance button to continue.



Evaluate the Performance of a Hybrid CNN-SVM Model

Select the hybrid CNN-SVM model to evaluate

Model: model-60-10-30

Model details:

```
Model name: model-60-10-30
AI Expert: reiniermaristela
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
```

Select the dataset to use

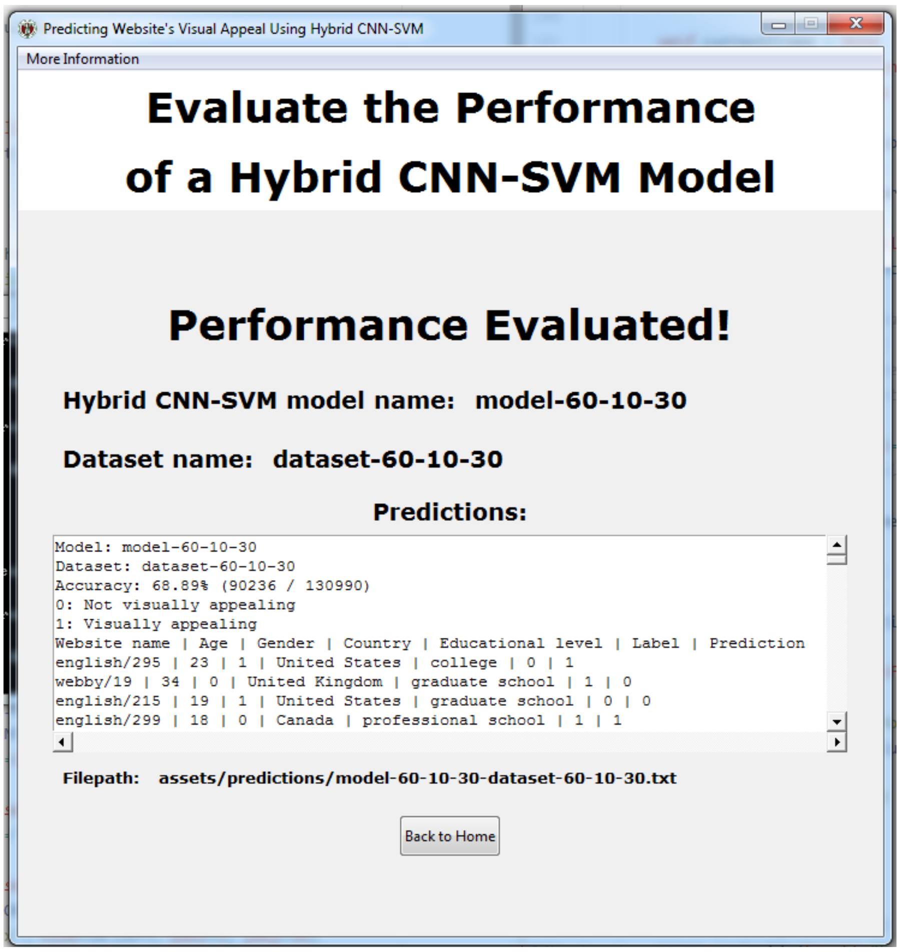
Dataset: dataset-60-10-30

Dataset details:

```
Dataset name: dataset-60-10-30
Created by:   reiniermaristela
Quantities:
  Training data:   261978
  Validation data: 43663
  Testing data:    130990
```

Figure 44: Evaluating the performance of the hybrid CNN-SVM model

The figure below shows the page where the details of the evaluation are shown. The name of the hybrid CNN-SVM model, the name of the dataset, and all the predictions of the hybrid model to the test set. Also, the accuracy is shown to better understand the performance of the hybrid CNN-SVM model. The AI expert can click the Back to Home button to return to the home page.



Evaluate the Performance of a Hybrid CNN-SVM Model

Performance Evaluated!

Hybrid CNN-SVM model name: model-60-10-30

Dataset name: dataset-60-10-30

Predictions:

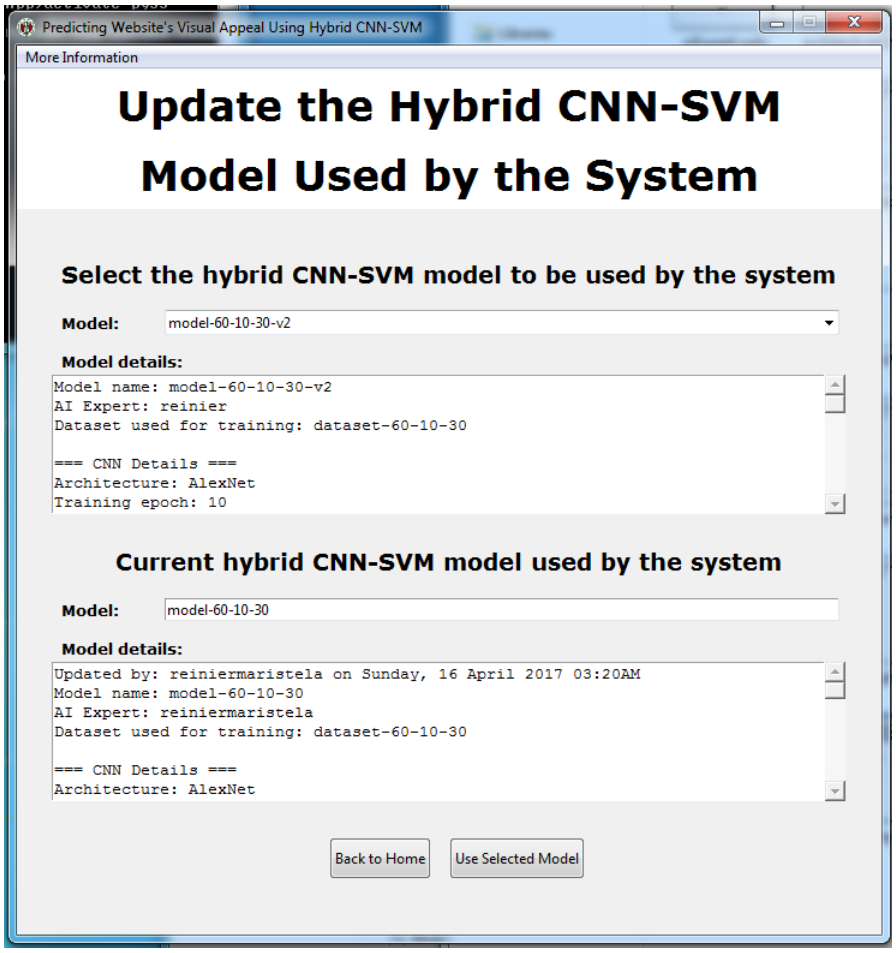
```
Model: model-60-10-30
Dataset: dataset-60-10-30
Accuracy: 68.89% (90236 / 130990)
0: Not visually appealing
1: Visually appealing
Website name | Age | Gender | Country | Educational level | Label | Prediction
english/295 | 23 | 1 | United States | college | 0 | 1
webby/19 | 34 | 0 | United Kingdom | graduate school | 1 | 0
english/215 | 19 | 1 | United States | graduate school | 0 | 0
english/299 | 18 | 0 | Canada | professional school | 1 | 1
```

Filepath: assets/predictions/model-60-10-30-dataset-60-10-30.txt

Figure 45: Evaluation successful

L. Updating the hybrid model used by the system

The figure below shows the page when the AI expert chose to update the hybrid CNN-SVM model to be used by the system. This can be performed by clicking the drop-down list at the upper half of the page and select the new hybrid CNN-SVM model he wishes to use. The details of the new and current models are shown accordingly. Next, the AI expert can click the Use Selected Model to continue.



Update the Hybrid CNN-SVM Model Used by the System

Select the hybrid CNN-SVM model to be used by the system

Model: model-60-10-30-v2

Model details:

```
Model name: model-60-10-30-v2
AI Expert: reinier
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
```

Current hybrid CNN-SVM model used by the system

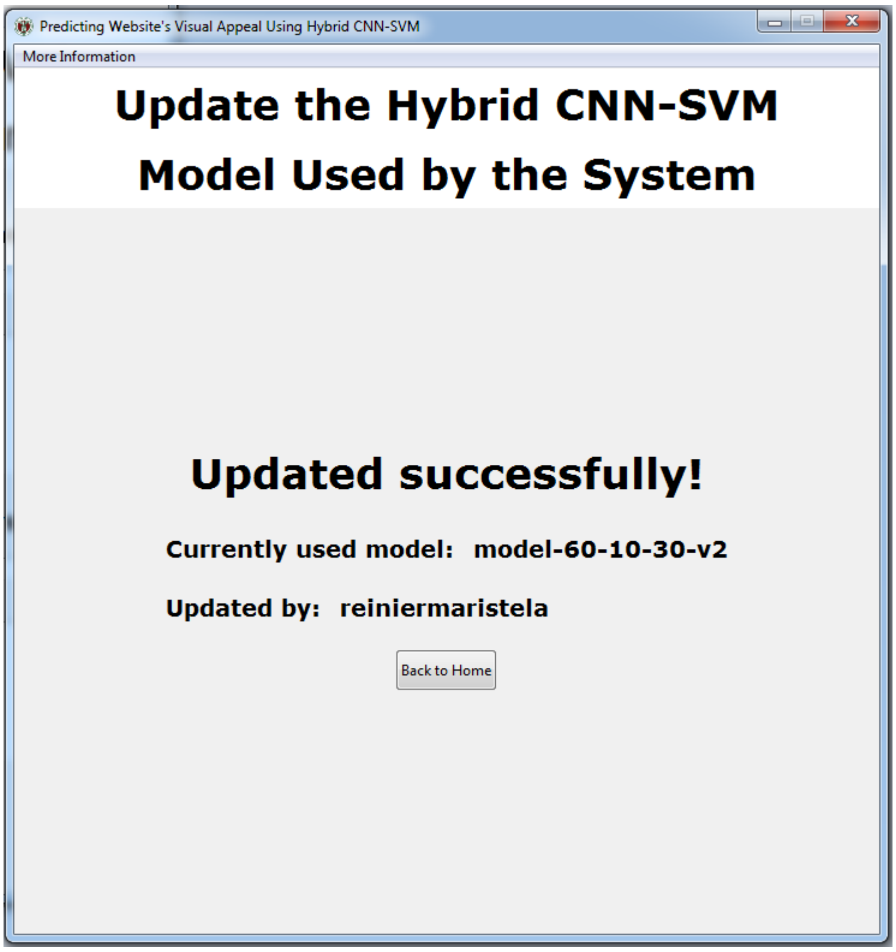
Model: model-60-10-30

Model details:

```
Updated by: reiniermaristela on Sunday, 16 April 2017 03:20AM
Model name: model-60-10-30
AI Expert: reiniermaristela
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
```

Figure 46: Selecting a new hybrid model for the system to use

The figure below shows that the selected hybrid CNN-SVM model is now the model that the system uses. The AI expert can click the Back to Home button to return to the home page.



Update the Hybrid CNN-SVM Model Used by the System

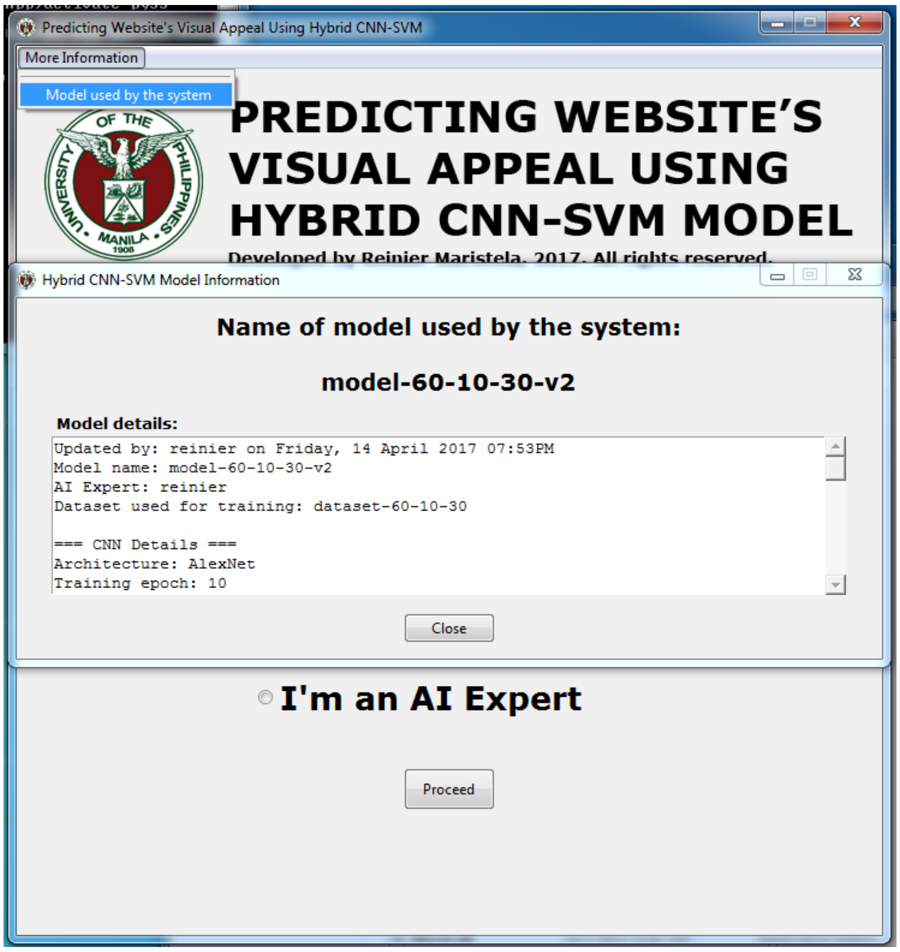
Updated successfully!

Currently used model: model-60-10-30-v2

Updated by: reiniermaristela

Figure 47: Update successful

Following the steps of viewing the current hybrid CNN-SVM model that the system uses, we can now observe that the model selected a while ago is the one that the system uses. As shown in the figure below, the current hybrid CNN-SVM model that the system uses is now updated.



PREDICTING WEBSITE'S VISUAL APPEAL USING HYBRID CNN-SVM MODEL

Developed by Reinier Maristela. 2017. All rights reserved.

Hybrid CNN-SVM Model Information

Name of model used by the system:

model-60-10-30-v2

Model details:

```
Updated by: reinier on Friday, 14 April 2017 07:53PM
Model name: model-60-10-30-v2
AI Expert: reinier
Dataset used for training: dataset-60-10-30

=== CNN Details ===
Architecture: AlexNet
Training epoch: 10
```

Figure 48: The new hybrid model used by the system

VI. Discussions

This AI system uses a hybrid of convolutional neural network and support vector machine that aims to predict a website's visual appeal with respect to the website's target users. The AI system has two users: the web designer and the AI expert. Web designers can use the AI system to predict the visual appeal of their websites by uploading a screenshot of the homepage and specifying the demographic backgrounds of the target users. The AI expert can use the AI system to create a hybrid CNN-SVM model, evaluate the performance of a hybrid model, and update the hybrid model to be used by the AI system.

The predicting method provided by Reinecke et. al. [10] uses hand-crafted feature extractors. As stated earlier, using hand-crafted feature extractors could lead to the following issues: (1) a human expert may not capture all the relevant information from the image [15] and (2) there is a need to decide what features to extract from the image that best represents a website's visual appeal [2]. Using this AI system, the AI expert can create a hybrid CNN-SVM model that can perform the extraction of features from a screenshot of a website's homepage and predict the visual appeal of the website with respect to the demographic backgrounds of its target users. Since the hybrid CNN-SVM model will do those tasks, there is no need to use hand-crafted feature extractors to predict a website's visual appeal. Thus, the issues mentioned earlier are addressed.

Since this AI system does not use hand-crafted feature extractors, the AI system can provide assistance to humans when predicting a website's visual appeal. Furthermore, the AI system provides rapid visual appeal predictions to web designers. They no longer need to conduct surveys or create their own predicting methods. In this way, the AI system can help web designers in creating visually appealing websites.

Khani et. al. [2] showed that a hybrid CNN-SVM model can be used to predict a website's visual appeal. However, one of the limitations of their implementation is that they used the CNN of Khosla et. al. [14]. This CNN was trained using a dataset that was created to predict the memorability of an image. Normally, we would want to train a CNN to a dataset that contains relevant data to perform a specific task. As we can observe, their CNN was trained to predict an image's memorability and not to predict a website's visual appeal. For the training of the SVM, they have used the dataset provided by Reinecke et. al. [9]. This dataset contains information about the visual appeal of different websites which was rated by participants around the world. They reported a 34.15% test error or 65.85% accuracy on their work.

For this implementation, the author trained both the CNN and SVM using the dataset provided by Reinecke et. al. [9]. The best perfoming hybrid CNN-SVM model created by the author reported a test error of 31.09% or 68.91% accuracy. Although we can observe that the obtained accuracy is higher, it is not conclusive to say that this hybrid CNN-SVM model is better than the hybrid CNN-SVM model of Khani et. al. [2]. The reason is that both hybrid CNN-SVM models are not trained on the same training and validations set and, also, not evaluated on the same test set. If there is a way to train and test both hybrid CNN-SVM models on the same training, validation, and test sets, then we can determine which one is better. Still, we can say that using the dataset provided by Reinecke et. al. [9] is enough to predict a website's visual appeal and using the trained CNN of Khosla et. al. [14] is not mandatory to perform the said task.

One of the main issues when creating and evaluating a hybrid CNN-SVM model is the length of time for these tasks to be accomplished. Creating a hybrid CNN-SVM model using 261,978 data for training and 43,663 data for validation takes approximately 24 hours to be accomplished. Evaluating the performance of the hybrid CNN-SVM model using 130,990 data takes approximately 12 hours to be accomplished. This means that, for the entire process to be efficient, it is not recommended to use trial-and-error method when determining what values to use for training the hybrid CNN-SVM model. For training the CNN model, the author consulted the work of Krizhevsky et. al. [13] since they are the creators of

AlexNet. Also, the author consulted the work of Hsu et. al. [30] as they provide practical guides for training support vector machines.

VII. Conclusions

An AI system that uses a hybrid of convolutional neural network and support vector machine is capable of predicting a website's visual appeal with respect to the age, gender, country, and educational level of the target users. Web designers can use this AI system to help them predict the visual appeal of their websites. AI experts can use this system to create a hybrid CNN-SVM model, evaluate a hybrid model, and update the hybrid model to be used by the AI system.

Using the technologies of deep learning, namely a hybrid of CNN and SVM, we have shown that the need for using hand-crafted feature extractors and the issues it brings when predicting a website's visual appeal can be eliminated.

This AI system is developed to assist humans in predicting a website's visual appeal. Since web designers have a rapid way of determining the visual appeal of their websites, the AI system can help web designers to create visually appealing websites.

VIII. Recommendations

It is suggested by Khani et. al. [2] that in order to improve the performance of hybrid CNN-SVM models when predicting a website's visual appeal, a better dataset with more samples could be collected. If that dataset could be collected, then we can now predict an exact visual appeal rating of 1 to 9, 1 being the least visually appealing and 9 the most visually appealing, instead of just visually appealing or not visually appealing.

Also, we can replace the CNN architecture of the hybrid CNN-SVM model from AlexNet to other CNN architecture that is better than AlexNet. Another area of improvement would be the fine-tuning of the hybrid CNN-SVM model. Since training and testing a hybrid model is a time-consuming task, the author may not be able to find the best hyperparameters. It would be a good performance boost if the best hyperparameters could be determined.

Lastly, since this is an AI system that is installed on a local machine, we can make this AI system to be a web service wherein a web designer could just go to a specific link to upload the screenshot and select the demographic backgrounds of the target users. In this way, it would be more convenient to the web designer since they do not need to install any dependencies and would only need to have an internet connection.

IX. Bibliography

[1] W. W. W. Consortium *et al.*, "Internet live stats." http://www.internetlivestats.com/total-number-of-websites/. Accessed: 2016-10-23.

[2] M. G. Khani, M. R. Mazinani, M. Fayyaz, and M. Hoseini, "A novel approach for website aesthetic evaluation based on convolutional neural networks," in *Web Research (ICWR), 2016 Second International Conference on*, pp. 48–53, IEEE, 2016.

[3] A. N. Tuch, E. E. Presslaber, M. StöCklin, K. Opwis, and J. A. Bargas-Avila, "The role of visual complexity and prototypicality regarding first impression of websites: Working towards understanding aesthetic judgments," *International Journal of Human-Computer Studies*, vol. 70, no. 11, pp. 794–811, 2012.

[4] W. Liu, F. Guo, G. Ye, and X. Liang, "How homepage aesthetic design influences users satisfaction: Evidence from china," *Displays*, vol. 42, pp. 25–35, 2016.

[5] B. Hasan, "Perceived irritation in online shopping: The impact of website design characteristics," *Computers in Human Behavior*, vol. 54, pp. 224–230, 2016.

[6] M. Seckler, S. Heinz, S. Forde, A. N. Tuch, and K. Opwis, "Trust and distrust on the web: User experiences and website characteristics," *Computers in Human Behavior*, vol. 45, pp. 39–50, 2015.

[7] S.-W. Lin, L. Y.-S. Lo, and T. K. Huang, "Visual complexity and figure-background color contrast of e-commerce websites: Effects on consumers' emotional responses," in *2016 49th Hawaii International Conference on System Sciences (HICSS)*, pp. 3594–3603, IEEE, 2016.

[8] G. Lindgaard, C. Dudek, D. Sen, L. Sumegi, and P. Noonan, "An exploration of relations between visual appeal, trustworthiness and perceived usability of homepages," *ACM Transactions on Computer-Human Interaction (TOCHI)*, vol. 18, no. 1, p. 1, 2011.

[9] K. Reinecke and K. Z. Gajos, "Quantifying visual preferences around the world," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 11–20, ACM, 2014.

[10] K. Reinecke, T. Yeh, L. Miratrix, R. Mardiko, Y. Zhao, J. Liu, and K. Z. Gajos, "Predicting users' first impressions of website aesthetics with a quantification of perceived visual complexity and colorfulness," in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, pp. 2049–2058, ACM, 2013.

[11] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." Book in preparation for MIT Press, 2016.

[12] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[13] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in neural information processing systems*, pp. 1097–1105, 2012.

[14] A. Khosla, A. S. Raju, A. Torralba, and A. Oliva, "Understanding and predicting image memorability at a large scale," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 2390–2398, 2015.

[15] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[16] G. Lindgaard, G. Fernandes, C. Dudek, and J. Brown, "Attention web designers: You have 50 milliseconds to make a good first impression!," *Behaviour & information technology*, vol. 25, no. 2, pp. 115–126, 2006.

[17] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1–9, 2015.

[18] B. Zhou, A. Lapedriza, J. Xiao, A. Torralba, and A. Oliva, "Learning deep features for scene recognition using places database," in *Advances in neural information processing systems*, pp. 487–495, 2014.

[19] X. Lu, Z. Lin, H. Jin, J. Yang, and J. Z. Wang, "Rapid: rating pictorial aesthetics using deep learning," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 457–466, ACM, 2014.

[20] X.-X. Niu and C. Y. Suen, "A novel hybrid cnn–svm classifier for recognizing handwritten digits," *Pattern Recognition*, vol. 45, no. 4, pp. 1318–1325, 2012.

[21] I. Jolliffe, *Principal component analysis.* Wiley Online Library, 2002.

[22] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011. Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm.

[23] F. Chollet, "Keras." https://github.com/fchollet/keras, 2015.

[24] A. Athanasopoulos, A. Dimou, V. Mezaris, and I. Kompatsiaris, "Gpu acceleration for support vector machines," in *WIAMIS 2011: 12th International Workshop on Image Analysis for Multimedia Interactive Services, Delft, The Netherlands, April 13-15, 2011*, TU Delft; EWI; MM; PRB, 2011.

[25] C. Analytics, “Anaconda software distribution. computer software. vers. 2-2.4.0.” https://continuum.io, 2016-11.

[26] R. Al-Rfou, G. Alain, A. Almahairi, C. Angermueller, D. Bahdanau, N. Ballas, F. Bastien, J. Bayer, A. Belikov, A. Belopolsky, Y. Bengio, A. Bergeron, J. Bergstra, V. Bisson, J. Bleecher Snyder, N. Bouchard, N. Boulanger-Lewandowski, X. Bouthillier, A. de Brébisson, O. Breuleux, P.-L. Carrier, K. Cho, J. Chorowski, P. Christiano, T. Cooijmans, M.-A. Côté, M. Côté, A. Courville, Y. N. Dauphin, O. Delalleau, J. Demouth, G. Desjardins, S. Dieleman, L. Dinh, M. Ducoffe, V. Dumoulin, S. Ebrahimi Kahou, D. Erhan, Z. Fan, O. Firat, M. Germain, X. Glorot, I. Goodfellow, M. Graham, C. Gulcehre, P. Hamel, I. Harlouchet, J.-P. Heng, B. Hidasi, S. Honari, A. Jain, S. Jean, K. Jia, M. Korobov, V. Kulkarni, A. Lamb, P. Lamblin, E. Larsen, C. Laurent, S. Lee, S. Lefrancois, S. Lemieux, N. Léonard, Z. Lin, J. A. Livezey, C. Lorenz, J. Lowin, Q. Ma, P.-A. Manzagol, O. Mastropietro, R. T. McGibbon, R. Memisevic, B. van Merriënboer, V. Michalski, M. Mirza, A. Orlandi, C. Pal, R. Pascanu, M. Pezeshki, C. Raffel, D. Renshaw, M. Rocklin, A. Romero, M. Roth, P. Sadowski, J. Salvatier, F. Savard, J. Schlüter, J. Schulman, G. Schwartz, I. V. Serban, D. Serdyuk, S. Shabanian, E. Simon, S. Spieckermann, S. R. Subramanyam, J. Sygnowski, J. Tanguay, G. van Tulder, J. Turian, S. Urban, P. Vincent, F. Visin, H. de Vries, D. Warde-Farley, D. J. Webb, M. Willson, K. Xu, L. Xue, L. Yao, S. Zhang, and Y. Zhang, “Theano: A Python framework for fast computation of mathematical expressions,” *arXiv e-prints*, vol. abs/1605.02688, May 2016.

[27] J. Nickolls, I. Buck, M. Garland, and K. Skadron, “Scalable parallel programming with cuda,” *Queue*, vol. 6, no. 2, pp. 40–53, 2008.

[28] Heuritech, “Convnets-keras library.” https://github.com/heuritech/convnets-keras, 2016.

[29] G. Taylor and W. Ding, "Theano-based large-scale visual recognition with multiple gpus." hdl:10864/10911, 2015-03.

[30] C.-W. Hsu, C.-C. Chang, C.-J. Lin, *et al.*, "A practical guide to support vector classification," 2003.

X. Appendix

A. Source Code

A..1 BinaryFeature.py

```
class BinaryFeature():
    def __init__(self, age, gender, country, educLevel):
        self.age = age
        self.gender = gender
        self.country = country
        self.educLevel = educLevel

        self.numOfAge = 5
        self.numOfGender = 2
        self.numOfCountry = 33
        self.numOfEducLevel = 7

    def getFeatures(self):
        binFeatures = []
        binFeatures += self.transform("age")
        binFeatures += self.transform("gender")
        binFeatures += self.transform("country")
        binFeatures += self.transform("educLevel")

        return binFeatures

    def transform(self, background):
        feature = []
        id = 0
        max = 0
        if background == "age":
            id = self.age
            max = self.numOfAge
        elif background == "gender":
            id = self.gender
            max = self.numOfGender
        elif background == "country":
            id = self.country
            max = self.numOfCountry
        elif background == "educLevel":
            id = self.educLevel
            max = self.numOfEducLevel

        for i in range(0, max):
            if id == i:
                feature += [1]
            else:
                feature += [0]

        return feature
```

A..2 CBoxHelper.py

```
import os
from FileManipulator import FileManipulator

class CBoxHelper():
    def __init__(self, baseString, path):
        self.baseString = baseString
        self.path = path

    def getValues(self):
        values = [self.baseString]
        try:
            values = values + os.listdir(self.path)
        except OSError:
            pass
        return values

    def getValuesDict(self):
        valuesDict = {}

        i = 0
        for values in self.getValues():
            valuesDict[values] = i
            i = i + 1

        return valuesDict

    def getValuesDetails(self):
        detailsDict = {self.baseString : ""}

        for values in self.getValues():
            fileMan = FileManipulator(self.path + "/" + values + "/details.txt")
            detailsDict[values] = fileMan.readContents()

        return detailsDict
    def getModelsDetails(self):
        detailsDict = {self.baseString : ""}

        for values in self.getValues():
            strToWrite = ""

            fileMan = FileManipulator(self.path + "/" + values + "/details.txt")
            strToWrite += fileMan.readContents()
            fileMan = FileManipulator(self.path + "/" + values + "/cnn.txt")
            strToWrite += fileMan.readContents()
            fileMan = FileManipulator(self.path + "/" + values + "/pca.txt")
            strToWrite += fileMan.readContents()
            fileMan = FileManipulator(self.path + "/" + values + "/svm.txt")
            strToWrite += fileMan.readContents()
            fileMan = FileManipulator(self.path + "/" + values + "/performance.txt")
            strToWrite += fileMan.readContents()

            detailsDict[values] = strToWrite

        return detailsDict

    def getBaseString(self):
        return self.baseString

    #returns {"SGD" : [ [label, default, guide], [label, default, guide] ] }
    def getOptHyperDict(self):
        optHyperDict = {self.baseString : [[]]}

        for values in self.getValues():
            fileMan = FileManipulator(self.path + "/" + values
                                      + "/hyperparameters.txt")
            optHyperDict[values] = []
            contents = fileMan.readContents()

            for line in contents.split('\n'):
                strArray = line.split(',')
                if (len(strArray) == 3):
                    label = strArray[0]
                    default = strArray[1]
                    guide = strArray[2]
                    hyperList = [label, default, guide]
                    optHyperDict[values].append(hyperList)

        return optHyperDict
```

A..3 CNNTrain.py

```
import matplotlib
matplotlib.use("TkAgg")

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
import matplotlib.animation as animation
from matplotlib import style

import tkinter as tk
from tkinter import ttk
import tkinter.scrolledtext as tkst
import Font as ft

from HybridModelTrainer import HybridModelTrainer
from FileManipulator import FileManipulator

style.use("ggplot")


class CNNTrain(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller

        fileMan = FileManipulator("assets/models/" + self.controller.getModelToCreate() + "/cnn.txt")
        self.trainMsg = fileMan.readContents()

        self.label = tk.Label(self,
                              text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.figure = Figure(figsize=(8,5), dpi=80)
        self.subPlot = self.figure.add_subplot(111) #111 1x1 1 chart
        self.subPlot.set_xlabel('Training epoch')
        self.subPlot.set_ylabel('Red - Training loss\nBlue - Validation loss')

        self.modelName = self.controller.getModelToCreate()
        self.basePath = "assets/models/" + self.modelName
        fileObj = FileManipulator(self.basePath + "/trainValLosses.txt")
        losses = fileObj.readContents().split("\n")

        trainEpoch = []
        trainLoss = []
        validateEpoch = []
        validateLoss = []
        for line in losses:
            values = line.split(",")
            if(len(values) == 4):
                trainEpoch.append(int(values[0]))
                trainLoss.append(float(values[1]))
                validateEpoch.append(int(values[2]))
                validateLoss.append(float(values[3]))

        self.subPlot.plot(trainEpoch, trainLoss, validateEpoch, validateLoss)

        self.graphLabel = tk.Label(self.parentFrame,
                                   text="Graph of Training and Validation Losses",
                                   font=ft.LARGE_FONT)
        self.graphLabel.pack(padx=10, pady=10)

        self.graphFrame = tk.Frame(self.parentFrame)
        self.graphFrame.pack()

        self.canvas = FigureCanvasTkAgg(self.figure, self.graphFrame)
        self.canvas.show()
        self.canvas.get_tk_widget().pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        self.toolbar = NavigationToolbar2TkAgg(self.canvas, self.graphFrame)
        self.toolbar.update()
        self.canvas._tkcanvas.pack(side=tk.TOP, fill=tk.BOTH, expand=True)

        self.trainFrame = tk.Frame(self.parentFrame)
        self.trainFrame.pack()

        self.trainEntry = tkst.ScrolledText(self.trainFrame,
                                            height=7, width=78, wrap="word")

        self.trainEntry.insert(tk.INSERT, self.trainMsg)
        self.trainEntry.config(state="disabled")
        self.trainEntry.pack(fill="x")

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Train PCA",
                                     command= self.proceedToPCATrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToPCATrain(self):
        modelName = self.controller.getModelToCreate()
        trainer = HybridModelTrainer(modelName)
        trainer.startTrainPCA()

        fileObj = FileManipulator("assets/models/" + modelName
                                  + "/origFeatures.txt")
        origFeatCont = fileObj.readContents()

        fileObj = FileManipulator("assets/models/" + modelName
                                  + "/reducFeatures.txt")
        reducFeatCont = fileObj.readContents()

        self.controller.createPCATrain(origFeatCont, reducFeatCont)
```

A..4 CreateDataset.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

class CreateDataset(tk.Frame):
    def __init__(self, parent, controller, nameStr,
                 trainingInt, validationInt, testInt, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.nameStr = nameStr
        self.trainingInt = trainingInt
        self.validationInt = validationInt
        self.testInt = testInt
        self.errorMsg = errorMsg

        self.label1 = tk.Label(self, text="Create Dataset",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.nameFrame = tk.Frame(self.parentFrame)
        self.nameFrame.pack(fill="x")

        self.nameLabel = tk.Label(self.nameFrame, text="Name of dataset:",
                                  font=ft.NORMAL_FONT)
        self.nameLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=15)
        self.nameEntry = tk.Entry(self.nameFrame)
        self.nameEntry.insert(0, self.nameStr)
        self.nameEntry.pack(fill="x", padx=10, expand="True")

        self.partitionLabel = tk.Label(self.parentFrame,
                    text="Specify the partition of the dataset in percentage",
                    font=ft.LARGE_FONT)
        self.partitionLabel.pack(padx=10, pady=5)

        self.partitionLabel = tk.Label(self.parentFrame,
                    text="Number of data: 436,631 ", font=ft.LARGE_FONT)
        self.partitionLabel.pack(padx=10, pady=5)

        self.partitionLabel = tk.Label(self.parentFrame,
                    text="Total partition must not exceed 100%",
                    font=ft.NORMAL_FONT)
        self.partitionLabel.pack(padx=10, pady=(20,5))

        self.trainingFrame = tk.Frame(self.parentFrame)
        self.trainingFrame.pack()

        self.trainingLabel = tk.Label(self.trainingFrame,
                                      text="Training set:", font=ft.NORMAL_FONT)
        self.trainingLabel.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.trainingEntry = tk.Entry(self.trainingFrame)
        self.trainingEntry.insert(0, self.trainingInt)
        self.trainingEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.trainingPerc = tk.Label(self.trainingFrame,
                                     text="%", font=ft.NORMAL_FONT)
        self.trainingPerc.pack(fill="x", side=tk.LEFT)

        self.trainingGuide = tk.Label(self.trainingFrame,
                                      text="int >= 1", font=ft.NORMAL_FONT)
        self.trainingGuide.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.validationFrame = tk.Frame(self.parentFrame)
        self.validationFrame.pack()

        self.validationLabel = tk.Label(self.validationFrame,
                                        text="Validation set:",
                                        font=ft.NORMAL_FONT)
        self.validationLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.validationEntry = tk.Entry(self.validationFrame)
        self.validationEntry.insert(0, self.validationInt)
        self.validationEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.validationPerc = tk.Label(self.validationFrame,
                                       text="%", font=ft.NORMAL_FONT)
        self.validationPerc.pack(fill="x", side=tk.LEFT)

        self.validationGuide = tk.Label(self.validationFrame,
                                        text="int >= 1", font=ft.NORMAL_FONT)
        self.validationGuide.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.testFrame = tk.Frame(self.parentFrame)
        self.testFrame.pack()

        self.testLabel = tk.Label(self.testFrame,
                                  text="Test set:", font=ft.NORMAL_FONT)
        self.testLabel.pack(fill="x", side=tk.LEFT, padx=(10, 53), pady=10)

        self.testEntry = tk.Entry(self.testFrame)
        self.testEntry.insert(0, self.testInt)
        self.testEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.testPerc = tk.Label(self.testFrame, text="%", font=ft.NORMAL_FONT)
        self.testPerc.pack(fill="x", side=tk.LEFT)

        self.testGuide = tk.Label(self.testFrame,
                                  text="int >= 1", font=ft.NORMAL_FONT)
        self.testGuide.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.createBtn = ttk.Button(self.buttonFrame, text="Create dataset",
                                    command= self.updateSuccessDataset )
        self.createBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

        self.errorLabel = tk.Label(self.parentFrame, text="Error messages:",
                                   font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame, height=5,
                                            width=70, wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)
```

```
        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x", padx=(10,0) )

    def backToHome(self):
        self.controller.createHomePage()

    def updateSuccessDataset(self):
        nameStr = self.nameEntry.get()
        trainingInt = self.trainingEntry.get()
        validationInt = self.validationEntry.get()
        testInt = self.testEntry.get()

        self.controller.updateSuccessDataset(nameStr.strip(),
                                             trainingInt.strip(),
                                             validationInt.strip(),
                                             testInt.strip())
```

A..5 CreateModel.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

class CreateModel(tk.Frame):
    def __init__(self, parent, controller, nameStr, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.nameStr = nameStr
        self.errorMsg = errorMsg

        self.label1 = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.nameFrame = tk.Frame(self.parentFrame)
        self.nameFrame.pack(fill="x")

        self.nameLabel = tk.Label(self.nameFrame,
                                  text="Name of Hybrid CNN-SVM model:",
                                  font=ft.NORMAL_FONT)
        self.nameLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=15)

        self.nameEntry = tk.Entry(self.nameFrame)
        self.nameEntry.insert(0, self.nameStr)
        self.nameEntry.pack(fill="x", padx=10, expand="True")

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.createBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                    command= self.updateCreateModel )
        self.createBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

        self.errorLabel = tk.Label(self.parentFrame, text="Error messages:",
                                   font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame, height=5,
                                            width=70, wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)

        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x", padx=(10,0) )

    def backToHome(self):
        self.controller.createHomePage()

    def updateCreateModel(self):
        nameStr = self.nameEntry.get().strip()
        self.controller.updateCreateModel(nameStr)
```

A..6 CreateModelLoad.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

class CreateModelLoad(tk.Frame):
    def __init__(self, parent, controller, nameStr, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.nameStr = nameStr
        self.errorMsg = errorMsg

        self.label1 = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.nameFrame = tk.Frame(self.parentFrame)
        self.nameFrame.pack(fill="x")

        self.nameLabel = tk.Label(self.nameFrame,
                                  text="Name of Hybrid CNN-SVM model:",
                                  font=ft.NORMAL_FONT)
        self.nameLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=15)

        self.nameEntry = tk.Entry(self.nameFrame)
        self.nameEntry.insert(0, self.nameStr)
        self.nameEntry.pack(fill="x", padx=10, expand="True")

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command=self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.createBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                    command=self.updateCreateModel )
        self.createBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

        self.errorLabel = tk.Label(self.parentFrame, text="Error messages:",
                                   font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame, height=5,
                                            width=70, wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)

        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x", padx=(10,0) )

    def backToHome(self):
        self.controller.createHomePage()

    def updateCreateModel(self):
        nameStr = self.nameEntry.get().strip()
        self.controller.updateCreateModelLoad(nameStr)
```

A..7 DatasetPartitioner.py

```
import os
import random
import csv
from FileManipulator import FileManipulator

class DatasetPartitioner():
    def __init__(self, name, train, validate, test, AIExpert):
        self.name = name
        self.train = train
        self.validate = validate
        self.test = test
        self.AIExpert = AIExpert

        self.basePath = "assets/"
        self.datasetDir = self.basePath + "datasets/" + self.name

        self.trainPath = self.datasetDir + "/train.txt"
        self.validatePath = self.datasetDir + "/validate.txt"
        self.testPath = self.datasetDir + "/test.txt"
        self.detailsPath = self.datasetDir + "/details.txt"

        self.baseDataset = self.basePath + "reinecke_datasets/base_dataset.csv"

    def partitionTheDataset(self):
        try:
            os.mkdir(self.datasetDir)

            trainFile = open(self.trainPath, "a+", encoding='utf-8')
            validateFile = open(self.validatePath, "a+", encoding='utf-8')
            testFile = open(self.testPath, "a+", encoding='utf-8')

            total = self.train + self.validate + self.test

            trainCtr = 0
            validateCtr = 0
            testCtr = 0

            with open(self.baseDataset, 'rt', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        dataStr = line[0]
                        choice = self.getChoices(trainCtr, validateCtr, testCtr)
                        if choice == 'train':
                            trainFile.write(dataStr + "\n")
                            trainCtr = trainCtr + 1
                        elif choice == 'validate':
                            validateFile.write(dataStr + "\n")
                            validateCtr = validateCtr + 1
                        elif choice == 'test':
                            testFile.write(dataStr + "\n")
                            testCtr = testCtr + 1

            trainFile.close()
            validateFile.close()
            testFile.close()

            detailsFile = open(self.detailsPath, "a+", encoding='utf-8')
            detailsFile.write(self.getDetails())
            detailsFile.close()

            return True
        except (IOError, OSError) as e:
            pass
        return False

    def getDetails(self):
        string = "Dataset name: " + self.name + "\n"
        string += "Created by:    " + self.AIExpert + "\n"
        string += "Quantities:\n"
        string += "   Training data:    " + str(self.train) + "\n"
        string += "   Validation data: " + str(self.validate) + "\n"
        string += "   Testing data:     " + str(self.test) + "\n"
        return string

    def getChoices(self, trainCtr, validateCtr, testCtr):
        choices = []
        if trainCtr != self.train:
            choices.append('train')
        if validateCtr != self.validate:
            choices.append('validate')
        if testCtr != self.test:
            choices.append('test')

        if len(choices) != 0:
            return random.choice(choices)

        return ""
```

A..8 FileManipulator.py

```
import csv
import hashlib, uuid

class FileManipulator():

    def __init__(self, filename):
        self.filename = filename

    def appendToAccounts(self, username, password):
        try:
            file = open(self.filename, "a+", encoding='utf-8')
            salt = uuid.uuid4().hex
            hashedPass = hashlib.sha512(salt.encode('utf-8')
                         + password.encode('utf-8')).hexdigest()
            file.write(username + "," + salt + "," + hashedPass + "\n")
            file.close()
        except IOError:
            pass

    def usernameExists(self, username):
        filepath = self.filename
        try:
            with open(filepath, 'rt', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        arrStr = line[0].split(",")
                        if len(arrStr) == 3:
                            if arrStr[0] == username:
                                return True
        except IOError:
            pass
        return False

    def checkUserPass(self, username, password):
        filepath = self.filename
        try:
            with open(filepath, 'rt', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        arrStr = line[0].split(",")
                        if len(arrStr) == 3:
                            existingUser = arrStr[0]
                            salt = arrStr[1]
                            existingPass = arrStr[2]
                            hashedPass = hashlib.sha512(salt.encode('utf-8')
                                        + password.encode('utf-8')).hexdigest()
                        if existingUser == username and existingPass == hashedPass:
                            return True
        except IOError:
            pass
        return False

    def checkSUPass(self, password):
        salt = "d80e85efb1b048fb95439d2c532b2b21"
        hashedPass = "e0a0c66bf41b58e296a0645f5b09ff7d54bd636f499351535652d7fc825a606b0ebdba2ff8001a367b5e3d32
        saltPassHash = hashlib.sha512(salt.encode('utf-8')
                       + password.encode('utf-8')).hexdigest()
        if (saltPassHash == hashedPass):
            return True

        return False

    def appendToDatasetFile(self, string):
        try:
            file = open(self.filename, "a+", encoding='utf-8')
            file.write(string + "\n")
            file.close()
        except IOError:
            pass

    def readContents(self):
        content = ""
        try:
            with open(self.filename, 'r') as content_file:
                content = content_file.read()
        except IOError:
            pass
        return content

    def writeContents(self, string):
        try:
            file = open(self.filename, "w+", encoding='utf-8')
            file.write(string)
            file.close()
        except IOError:
            pass

    def appendToFile(self, string):
        try:
            file = open(self.filename, "a+", encoding='utf-8')
            file.write(string)
            file.close()
        except IOError:
            pass
```

A..9 Font.py

```
FONT = "Verdana"
TITLE_FONT = (FONT, 26, "bold")
LARGE_FONT = (FONT, 14, "bold")
NORMAL_FONT = (FONT, 10, "bold")
SMALL_FONT = (FONT, 8, "bold")
BOLD_ITALIC = (FONT, 12, "bold italic")
BOLD_FONT = (FONT, 12, "bold")
```

A..10 FormValidator.py

```
from FileManipulator import FileManipulator
import os

class FormValidator():

    def checkLength(self, label, input):
        if len(input) >= 6:
            return ""

        return "- " + label + " must be at least 6 characters.\n"

    def checkAgreeTerms(self, agreedToTerms):
        if agreedToTerms == 1:
            return ""

        return "- Please accept the Terms and Conditions.\n"

    def checkPasswordsEquality(self, password, confPass):
        if password == confPass:
            return ""

        return "- Both passwords must be identical.\n"

    def checkNullInput(self, arrayOfInput):
        for input in arrayOfInput:
            if len(input) == 0:
                return "- Please input all the necessary values.\n"
        return ""

    def checkNullInputDict(self, list, dict):
        for param in list:
            if len(dict[param[0]]) == 0:
                return "- Please input all the necessary values.\n"
        return ""
```

```
def checkExceedPartition(self, trainingInt, validationInt, testInt):
    totalPartition = int(trainingInt) + int(validationInt) + int(testInt)

    if totalPartition <= 100:
        return ""
    return "- Total partition must not exceed 100%\n"

def checkExistenceOfZero(self, trainingInt, validationInt, testInt):
    if int(trainingInt) > 0 and int(validationInt) > 0 and int(testInt) > 0:
        return ""
    return "- A value of 0 or less than 0 is detected in the partition.\n"

def checkNumericInput(self, trainingInt, validationInt, testInt):
    try:
        train = int(trainingInt)
        validate = int(validationInt)
        test = int(testInt)
        errorMsg = ""
        errorMsg += self.checkExceedPartition(train, validate, test)
        errorMsg += self.checkExistenceOfZero(train, validate, test)

        return errorMsg

    except ValueError:
        return "- Please input an integer from 1 - 100.\n"

def checkUniqueUsername(self, username):
    fileObj = FileManipulator("assets/accounts.txt")
    usernameExists = fileObj.usernameExists(username)
    if not usernameExists:
        return ""
    return "- Username is already taken.\n"

def checkUniqueDataset(self, name):
    basePath = "assets/datasets"
    arrDir = os.listdir(basePath)
    for dir in arrDir:
        if dir == name:
            return "- Dataset name already exists.\n"
    return ""

def checkUniqueModel(self, name):
    basePath = "assets/models"
    arrDir = os.listdir(basePath)
    for dir in arrDir:
        if dir == name:
            return "- Model name already exists.\n"

    basePath = "assets/cnn"
    arrDir = os.listdir(basePath)
    for dir in arrDir:
        if dir == name:
            return "- Model name already exists.\n"

    return ""

def checkEpochBatch(self, trainEpoch, batchSize):
    if self.isIntInput(trainEpoch) and self.isIntInput(batchSize):
        if int(trainEpoch) >= 1 and int(batchSize) >= 1:
            return ""
    return "- Please enter a train epoch/batch size >= 1.\n"

def checkHyperInput(self, list, dict):
    for param in list:
        label = param[0]
        value = dict.get(label)
        type, lowLimit, highLimit = self.getRules(label)
        if type == "float1":
            if not self.isFloatInput(value) or float(value) < lowLimit:
                return "- Please enter a float >= 0.\n"

        elif type == "float2":
            if not self.isFloatInput(value) or float(value) <= lowLimit
                    or float(value) >= highLimit:
                return "- Please enter a float value 0 < beta < 1.\n"
        elif type == "boolean":
            if not self.isBooleanInput(value):
                return "- Please enter 0 - false or 1 - true.\n"

    return ""

def isValidCandG(self, list):
    for param in list:
        if not self.isFloatInput(param):
            return "- Please enter a float value for the Cost and Gamma.\n"

    return ""

def isValidStep(self, list):
    for param in list:
        if not self.isPosFloatInput(param):
            return "- Please enter a positive float value for the steps.\n"
    return ""

def isEqualC(self, cStart, cEnd):
    if(cStart != cEnd):
        return ""
    return "- Please enter a different values for Cost Start and Cost End.\n"

def isEqualG(self, gStart, gEnd):
    if(gStart != gEnd):
        return ""

    return "- Please enter a different values for Gamma Start and Gamma End.\n"


def isIntInput(self, input):
    try:
        int(input)
        return True
    except:
        pass
    return False

def isFloatInput(self, input):
    try:
        float(input)
        return True
    except:
        pass
    return False

def isPosFloatInput(self, input):
    try:
        value = float(input)
        if value > 0:
            return True
    except:
        pass
    return False

def isBooleanInput(self, input):
    if(input == "1" or input == "0"):
        return True
    return False

def getRules(self, label):
    if label == "Learning rate" or label == "Rho" or label == "Epsilon"
    or label == "Decay" or label == "Schedule decay" or label == "Momentum":
        return "float1", 0, -1
    elif label == "Beta 1" or label == "Beta 2":
        return "float2", 0, 1
    elif label == "Nesterov":
        return "boolean", -1, -1
    return "error", -1, -1
```

A..11 HomePage.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class HomePage(tk.Frame):

    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.welcomeLabel = tk.Label(self.parentFrame,
                                     text="Welcome, "
                                     + self.controller.getCurrentAIExpert()
                                     + "!", font=ft.TITLE_FONT)
        self.welcomeLabel.pack(padx=10, pady=5)

        self.selectLabel = tk.Label(self.parentFrame,
                                    text="Please select an option",
                                    font=ft.TITLE_FONT)
        self.selectLabel.pack(padx=10, pady=(0,20))

        self.frameRadio = tk.Frame(self.parentFrame)
        self.frameRadio.pack(fill="x", padx=50)

        self.radioVar = tk.IntVar()
        self.radioStyle = ttk.Style()
        self.radioStyle.configure('homepage.TRadiobutton',
                                  font=('Verdana', 14, "bold"))

        self.datasetRadio = ttk.Radiobutton(self.frameRadio,
                                            text="Create a dataset",
                                            variable=self.radioVar,
                                            value=0,
                                            style="homepage.TRadiobutton")
        self.datasetRadio.pack(anchor="w", pady=10)
        self.createRadio = ttk.Radiobutton(self.frameRadio,
                                           text="Create a hybrid CNN-SVM model",
                                           variable=self.radioVar,
                                           value=1,
                                           style="homepage.TRadiobutton")
        self.createRadio.pack(anchor="w", pady=10)

        #
        self.loadRadio = ttk.Radiobutton(self.frameRadio,
                        text="Create a hybrid model using a trained CNN model",
                        variable=self.radioVar,
                        value=2,
                        style="homepage.TRadiobutton")
        self.loadRadio.pack(anchor="w", pady=10)
        #

        self.evaluateRadio = ttk.Radiobutton(self.frameRadio,
                    text="Evaluate the performance of a hybrid CNN-SVM model",
                    variable=self.radioVar,
                    value=3,
                    style="homepage.TRadiobutton")
        self.evaluateRadio.pack(anchor="w", pady=10)

        self.updateRadio = ttk.Radiobutton(self.frameRadio,
                    text="Update the hybrid CNN-SVM model used by the system",
                    variable=self.radioVar,
                    value=4,
                    style="homepage.TRadiobutton")
        self.updateRadio.pack(anchor="w", pady=(10,20))

        self.radioVar.set(0)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.backBtn = ttk.Button(self.buttonFrame, text="Back",
                                  command= self.backToWelcome )
        self.backBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                     command= self.proceedWithChoice )
        self.proceedBtn.pack(side=tk.LEFT, ipady=5,  pady=20)

    def backToWelcome(self):
        self.controller.createWelcomePage()

    def proceedWithChoice(self):
        choice = self.radioVar.get()

        if choice == 0:
            self.controller.createCreateDataset("", "", "", "", "")
        elif choice == 1:
            self.controller.createCreateModel("", "")
        elif choice == 2:
            self.controller.createCreateModelLoad("", "")
        elif choice == 3:
            self.controller.createSelectCNNTest(0, "", 0, "")
        elif choice == 4:
            self.controller.createUpdateModel(0, "")
```

A..12 HybridModel.py

```
from convnetskeras.convnets import convnet, preprocess_image_batch
from keras.models import Model
from keras.layers import Dense, Dropout, Activation

from sklearn.decomposition import PCA
from sklearn.externals import joblib

from FileManipulator import FileManipulator

import numpy as np
import subprocess
import os
from subprocess import *

class HybridModel():
    def __init__(self, name, websiteFeatureDict):
        self.name = name
        self.websiteFeatureDict = websiteFeatureDict

    def getFeatures4096(self, imagesPaths):
        weightsPath = self.name + ".h5"

        self.imagesPaths = imagesPaths[0]

        model = self.modifiedAlexNet4096(weights=weightsPath)
        imageToPredict = preprocess_image_batch(imagesPaths,
                                                img_size=(256,256),
                                                crop_size=(227,227),
                                                color_mode="rgb")

        return model.predict(imageToPredict)[0].reshape(1, -1)


    def getFeatures128(self, features4096):
        pcaPath = self.name + ".pkl"
        pca = joblib.load(pcaPath)

        return pca.transform(features4096)[0]

    def getPrediction(self, features128, features47):

        fileMan = FileManipulator("assets/datasets_webdesigner/features128.txt")
        string = "1 "  #dummy label
        for i in range(0, len(features128)):
            string += str(i + 1) + ":" + str(features128[i]) + " "
        fileMan.writeContents(string)

        svmScalePath = "assets/gpu_libsvm/svm-scale.exe"
        inputFilePath = "assets/datasets_webdesigner/features128.txt"
        outputFilePath = "assets/datasets_webdesigner/scaledFeatures128.txt"
        rangeFilePath = self.name + ".range"

        cmd = '"{0}" -r "{1}" "{2}" > "{3}"'.format(svmScalePath,
                                                   rangeFilePath,
                                                   inputFilePath,
                                                   outputFilePath)
        Popen(cmd, shell = True, stdout = PIPE).communicate()

        features175 = np.concatenate((features128, features47))

        fileMan = FileManipulator("assets/datasets_webdesigner/scaledFeatures128.txt")
        string = fileMan.readContents().split('\n')[0]
        #mapping
        for key in self.websiteFeatureDict:
            if self.imagesPaths.find(key) >= 0:
                string = "1 " + self.websiteFeatureDict.get(key)

        for i in range(len(features128), len(features175)):
            string += str(i + 1) + ":" + str(features175[i]) + " "

        fileMan = FileManipulator("assets/datasets_webdesigner/in.txt")
        fileMan.writeContents(string)

        svmPredictPath = "assets/gpu_libsvm/svm-predict.exe"
        inputFilePath = "assets/datasets_webdesigner/in.txt"
        svmModelPath = self.name + ".model"
        outputFilePath = "assets/datasets_webdesigner/out.txt"

        cmd = '"{0}" -q "{1}" "{2}" "{3}"'.format(svmPredictPath,
                                                 inputFilePath,
                                                 svmModelPath,
                                                 outputFilePath)
        Popen(cmd, shell = True, stdout = PIPE).communicate()

        fileMan = FileManipulator("assets/datasets_webdesigner/out.txt")
        contentOfOut = fileMan.readContents()
        prediction = contentOfOut.split('\n')[0]
        if prediction == '1':
            return "Prediction: Visually appealing"
        elif prediction == '0':
            return "Prediction: Not visually appealing"

        return ""

    def modifiedAlexNet(self, loaded_weights=None):
        #output layer = 1
        alexnet = convnet('alexnet', weights_path=None, heatmap=False)

        input_cnn = alexnet.input
        img_representation = alexnet.get_layer("dense_2").output

        classifier = Dense(1,name='classifier')(img_representation)
        classifier = Activation("sigmoid", name="sigmoid")(classifier)

        model = Model(input=input_cnn,output=classifier)
        if loaded_weights:
            model.load_weights(loaded_weights)

        return model

    def modifiedAlexNet4096(self, weights=None):
        #output layer = 4096
        modified_model = self.modifiedAlexNet(loaded_weights=weights)

        input = modified_model.input

        dense_1_output = modified_model.get_layer("dense_1").output
        featureVector = Dropout(0.5)(dense_1_output)
        featureVector = Dense(4096,name='featureVector',
                              activation='relu')(dense_1_output)

        model = Model(input=input,output=featureVector)
        return model
```

A..13 HybridModelTester.py

```
from sklearn.decomposition import PCA
from sklearn.externals import joblib

from FileManipulator import FileManipulator
from BinaryFeature import BinaryFeature

import numpy as np
import subprocess
import os
import csv
import ast

from subprocess import *


class HybridModelTester():

    def __init__(self, modelName, datasetName):
        self.modelName = modelName
        self.datasetName = datasetName

    def startTest(self):

        print("\nEvaluating performance of hybrid CNN-SVM model...")
        print("Please wait, this may take a while.")

        svmPath = "assets/models/" + self.modelName + "/"
                + self.modelName + ".model"

        fileMan = FileManipulator("assets/models/" + self.modelName
                                  + "/mapping.txt")
        websiteFeatureDict = ast.literal_eval(fileMan.readContents())

        inputList = self.getInputList()

        countryDict = {"Argentina" : 0, "Australia" : 1, "Austria" : 2,
                       "Belgium" : 3, "Brazil" : 4, "Bulgaria" : 5,
                       "Canada" : 6, "Chile" : 7, "Denmark" : 8,
                       "Finland" : 9, "France" : 10, "Germany" : 11,
                       "Greece" : 12, "Hungary" : 13, "India" : 14,
                       "Ireland" : 15, "Israel" : 16, "Italy" : 17,
                       "Lithuania" : 18,
                       "Macedonia The Former Yugoslav Republic Of" : 19,
                       "Mexico" : 20, "Netherlands" : 21, "New Zealand" : 22,
                       "Norway" : 23, "Poland" : 24, "Portugal" : 25,
                       "Romania" : 26, "Serbia" : 27, "Singapore" : 28,
                       "Spain" : 29, "Sweden" : 30, "United Kingdom" : 31,
                       "United States" : 32}

        educLevelDict = {"pre-high school" : 0, "high school" : 1,
                         "college" : 2, "professional school" : 3,
                         "graduate school" : 4, "PhD" : 5, "postdoctoral" : 6}

        fileMan = FileManipulator("testSVM.txt")
        fileMan.writeContents("")

        for i in range(0, len(inputList)):
            scaledFeatures = websiteFeatureDict.get(inputList[i][0])
            ageID = self.getAgeID(int(inputList[i][1]))
            genderID = int(inputList[i][2])
            countryID = countryDict.get(inputList[i][3])
            educLevelID = educLevelDict.get(inputList[i][4])
            binaryFeatures = BinaryFeature(ageID, genderID, countryID, educLevelID)
            featureVector47 = binaryFeatures.getFeatures()
            label = str(inputList[i][5])
            strToWrite = label + " "
            strToWrite += scaledFeatures
            for j in range(0, len(featureVector47)):
                strToWrite += str(j + 129) + ":" + str(featureVector47[j]) + " "
            strToWrite += "\n"
            fileMan.appendToFile(strToWrite)

        svmPredictPath = "assets/gpu_libsvm/svm-predict.exe"
        inputFilePath = "testSVM.txt"
        outputFilePath = "testSVMOut.txt"

        cmd = '"{0}" -q "{1}" "{2}" "{3}"'.format(svmPredictPath,
                                                  inputFilePath,
                                                  svmPath,
                                                  outputFilePath)
        Popen(cmd, shell = True, stdout = PIPE).communicate()

        fileMan = FileManipulator("testSVMOut.txt")
        predictions = fileMan.readContents().split("\n")

        fileMan = FileManipulator("assets/predictions/" + self.modelName
                                  + "-" + self.datasetName + ".txt")
        accuracyStr = self.computeAccuracy(inputList, predictions)

        strToWrite = "Model: " + self.modelName + "\n"
        strToWrite += "Dataset: " + self.datasetName + "\n"
        strToWrite += "Accuracy: " + accuracyStr + "\n"
        strToWrite += "0: Not visually appealing\n"
        strToWrite += "1: Visually appealing\n"
        strToWrite += "Website name | Age | Gender | Country "
        strToWrite += "| Educational level | Label | Prediction\n"

        fileMan.writeContents(strToWrite)

        for i in range(0, len(inputList)):
            if len(inputList[i]) == 6 and len(predictions[i]) == 1:
                websiteName = inputList[i][0]
                age = inputList[i][1]
                gender = inputList[i][2]
                country = inputList[i][3]
                educLevel = inputList[i][4]
                label = inputList[i][5]
                prediction = predictions[i][0]
                strToWrite = websiteName + " | " + age + " | " + gender
                             + " | " + country + " | " + educLevel
                             + " | " + label + " | " + prediction + "\n"
                fileMan.appendToFile(strToWrite)

        fileMan = FileManipulator("assets/models/" + self.modelName
                                  + "/performance.txt")
        strToWrite = self.datasetName + ": " + accuracyStr + "\n"
        fileMan.appendToFile(strToWrite)
        print("Evaluating performance done!")

    def getInputList(self):
        datasetPath = "assets/datasets/" + self.datasetName + "/test.txt"

        inputList = []
        try:
            with open(datasetPath, 'r+', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        arrStr = line[0].split(",")
                        if len(arrStr) == 6:
                            websiteName = arrStr[0]
                            age = arrStr[1]
                            gender = arrStr[2]
                            country = arrStr[3]
                            educLevel = arrStr[4]
                            label = arrStr[5]
                            inputList.append([websiteName, age, gender,
                                              country, educLevel, label])

            return inputList

        except IOError:
            pass

        return []

    def getAgeID(self, age):
        if age >= 12 and age <= 20:
            return 0
        elif age >= 21 and age <= 30:
            return 1
        elif age >= 31 and age <= 40:
            return 2
        elif age >= 41 and age <= 50:
            return 3
        elif age >= 51:
            return 4

        return -1

    def computeAccuracy(self, inputList, predictions):
        total = 0
        correct = 0
        for i in range(0, len(inputList)):
            if len(inputList[i]) == 6 and len(predictions[i]) == 1:
                label = float(inputList[i][5])
                prediction = float(predictions[i][0])
                if (label == prediction):
                    correct += 1
                total += 1
        accuracy = (float(correct) / float(total)) * 100
        accuracyStr = ("{0:.2f}".format(accuracy)) + "% (" + str(correct)
                      + " / " + str(total) + ")"

        return accuracyStr
```

A..14 HybridModelTrainer.py

```
from convnetskeras.convnets import convnet, preprocess_image_batch
from keras.models import Model
from keras.layers import Dense, Dropout, Activation
from keras.optimizers import *
import keras

from sklearn.decomposition import PCA
from sklearn.externals import joblib

from FileManipulator import FileManipulator
from BinaryFeature import BinaryFeature

import numpy as np
import subprocess
import os
import csv
import random
from subprocess import *

class LossHistory(keras.callbacks.Callback):
    def on_train_begin(self, logs={}):
        self.losses = []
        self.valLosses = []

    def on_epoch_end(self, batch, logs={}):
        self.losses.append(logs.get('loss'))
        self.valLosses.append(logs.get('val_loss'))

class HybridModelTrainer():

    def __init__(self, name):
        self.name = name

    def modifiedAlexNet(self, loaded_weights=None):
        #output layer = 1
        alexnet = convnet('alexnet', weights_path=None, heatmap=False)

        input_cnn = alexnet.input
        img_representation = alexnet.get_layer("dense_2").output

        classifier = Dense(1,name='classifier')(img_representation)
        classifier = Activation("sigmoid", name="sigmoid")(classifier)

        model = Model(input=input_cnn,output=classifier)
        if loaded_weights:
            model.load_weights(loaded_weights)

        return model

    def modifiedAlexNet4096(self, weights=None):
        #output layer = 4096
        modified_model = self.modifiedAlexNet(loaded_weights=weights)

        input = modified_model.input

        dense_1_output = modified_model.get_layer("dense_1").output
        featureVector = Dropout(0.5)(dense_1_output)
        featureVector = Dense(4096,name='featureVector',
                              activation='relu')(dense_1_output)

        model = Model(input=input,output=featureVector)
        return model

    def startTrainCNN(self):

        print("\nTraining CNN model...")
        print("Please wait, this may take a while.")
        modelName = self.name
        dataset = self.getDataset(modelName)
        trainEpoch = int(self.getTrainEpoch(modelName))
        batchSize = int(self.getBatchSize(modelName))
        modelOptimizer = self.getOptimizer(modelName)

        baseWebsitePath = "assets/reinecke_datasets/base_website.csv"
        x, y, xVal, yVal, xTest, yTest = self.getImagesAndLabelCNN(baseWebsitePath)

        weightsPath = "assets/cnn_architecture/AlexNet/modified_alexnet_weights.h5"

        model = self.modifiedAlexNet(loaded_weights=weightsPath)

        model.compile(optimizer=modelOptimizer, loss='binary_crossentropy',
                      metrics=['accuracy'])

        history = LossHistory()
        trainValLossesFile = FileManipulator("assets/models/" + modelName
                                             + "/trainValLosses.txt")
        xList = []
        yList = []
        xValList = []
        yValList = []
        for i in range(0, trainEpoch):
            #
            xCrop = preprocess_image_batch(x, img_size=(256,256),
                                           crop_size=(227,227),
                                           color_mode="rgb")
            xValCrop = preprocess_image_batch(xVal, img_size=(256,256),
                                              crop_size=(227,227),
                                              color_mode="rgb")
            #
            model.fit(xCrop, y, nb_epoch=1, batch_size=batchSize,
                      callbacks=[history], validation_data=(xValCrop, yVal))
            trainLoss = history.losses[0]
            validateLoss = history.valLosses[0]
            currEpoch = i + 1
            strToWrite = str(currEpoch) + "," + str(trainLoss) + ","
                         + str(currEpoch) + "," + str(validateLoss) + "\n"
            trainValLossesFile.appendToFile(strToWrite)


        xTest = preprocess_image_batch(xTest, img_size=(256,256),
                                       crop_size=(227,227), color_mode="rgb")

        model.save_weights("assets/models/"+modelName+"/"+ modelName+".h5")
        scores = model.evaluate(xTest, yTest)
        accuracy = scores[1] * 100

        fileMan = FileManipulator("assets/models/" + modelName + "/cnn.txt")
        fileMan.appendToFile("Training images: " + str(len(x)) + "\n")
        fileMan.appendToFile("Validation images: " + str(len(xVal)) + "\n")
        fileMan.appendToFile("Test images: " + str(len(xTest)) + "\n")
        fileMan.appendToFile("Accuracy: " + ("{0:.2f}".format(accuracy)) + "%\n")

        print("CNN model training done!")
        print("CNN model saved!")
```

```
def startTrainPCA(self):

    print("\nTraining PCA model...")
    print("Please wait, this may take a while.")

    modelName = self.name
    dataset = self.getDataset(modelName)

    x, websiteNames = self.getImagesPCA("assets/reinecke_datasets/base_website.csv")

    model4096 = self.modifiedAlexNet4096("assets/models/" + modelName
                                         + "/" + modelName+".h5")

    featureVector4096 = model4096.predict(x)
    pca = PCA(n_components=128)
    pca.fit(featureVector4096)

    featureVector128 = pca.transform(featureVector4096)

    sampleSize = str(len(featureVector4096))
    sampleReducSize = str(len(featureVector128))
    origFeatSize = str(len(featureVector4096[0]))
    reducFeatSize = str(len(featureVector128[0]))

    fileObj = FileManipulator("assets/models/"+modelName+"/origFeatures.txt")
    strToWrite = "Number of images: " + sampleSize + "\n"
    strToWrite += "Size of original feature vector: " + origFeatSize + "\n"
    strToWrite += np.array_str(featureVector4096)
    fileObj.writeContents(strToWrite)

    fileObj = FileManipulator("assets/models/"+modelName+"/reducFeatures.txt")
    strToWrite = "Number of images: " + sampleReducSize + "\n"
    strToWrite += "Size of reduced feature vector: " + reducFeatSize + "\n"
    strToWrite += np.array_str(featureVector128)
    fileObj.writeContents(strToWrite)

    fileObj = FileManipulator("assets/models/"+modelName+"/pca.txt")
    strToWrite = "\n=== PCA Details ===\n"
    strToWrite += "Number of images: " + sampleSize + "\n"
    strToWrite += "Size of original feature vector: " + origFeatSize + "\n"
    strToWrite += "Number of images: " + sampleReducSize + "\n"
    strToWrite += "Size of reduced feature vector: " + reducFeatSize + "\n"
    fileObj.writeContents(strToWrite)

    joblib.dump(pca, "assets/models/"+modelName+"/"+ modelName+".pkl")

    print("PCA model training done!")

def startParamSearch(self, cStart, cEnd, cStep, gStart, gEnd, gStep):

    print("\nPreparing for parameter selection...")
    print("Please wait, this may take a while.")

    modelName = self.name

    x, websiteNames = self.getImagesPCA("assets/reinecke_datasets/base_website.csv")

    model4096 = self.modifiedAlexNet4096("assets/models/" + modelName
                                         + "/" + modelName+".h5")

    featureVector4096 = model4096.predict(x)
    pca = joblib.load("assets/models/"+modelName+"/"+ modelName+".pkl")

    featureVector128 = pca.transform(featureVector4096)
    # x[i] == featureVector128[i]
    fileMan = FileManipulator("assets/gpu_libsvm/features128.txt")
    fileMan.writeContents("")
    for i in range(0, len(x)):
        strToWrite = "1 " #dummy label
        for j in range(0, len(featureVector128[i])):
            strToWrite += str(j + 1) + ":" + str(featureVector128[i][j]) + " "
        strToWrite += "\n"
        fileMan.appendToFile(strToWrite)

    svmScalePath = "assets/gpu_libsvm/svm-scale.exe"
    inputFilePath = "assets/gpu_libsvm/features128.txt"
    outputFilePath = "assets/gpu_libsvm/scaledFeatures128.txt"
    rangeFilePath = "assets/models/" + modelName + "/" + modelName + ".range"

    cmd = '"{0}" -l 0 -s "{1}" "{2}" > "{3}"'.format(svmScalePath,
                                                     rangeFilePath,
                                                     inputFilePath,
                                                     outputFilePath)
    Popen(cmd, shell = True, stdout = PIPE).communicate()

    fileMan = FileManipulator("assets/gpu_libsvm/scaledFeatures128.txt")
    scaledFeatures128 = fileMan.readContents().split("\n")
    websiteFeatureDict = {}
    for i in range(0, len(websiteNames)):

        websiteFeatureDict[websiteNames[i]] = scaledFeatures128[i][2:]

    fileMan = FileManipulator("assets/models/" + modelName + "/mapping.txt")
    fileMan.writeContents(str(websiteFeatureDict))

    dataset = self.getDataset(modelName)

    countryDict = {"Argentina" : 0, "Australia" : 1, "Austria" : 2,
                   "Belgium" : 3, "Brazil" : 4, "Bulgaria" : 5,
                       "Canada" : 6, "Chile" : 7, "Denmark" : 8,
                       "Finland" : 9, "France" : 10, "Germany" : 11,
                       "Greece" : 12, "Hungary" : 13, "India" : 14,
                       "Ireland" : 15, "Israel" : 16, "Italy" : 17,
                       "Lithuania" : 18,
                       "Macedonia The Former Yugoslav Republic Of" : 19,
                       "Mexico" : 20, "Netherlands" : 21, "New Zealand" : 22,
                       "Norway" : 23, "Poland" : 24, "Portugal" : 25,
                       "Romania" : 26, "Serbia" : 27, "Singapore" : 28,
                       "Spain" : 29, "Sweden" : 30, "United Kingdom" : 31,
                       "United States" : 32}

        educLevelDict = {"pre-high school" : 0, "high school" : 1, "college" : 2,
                         "professional school" : 3, "graduate school" : 4,
                         "PhD" : 5, "postdoctoral" : 6}

        fileMan = FileManipulator("assets/datasets/" + dataset + "/train.txt")
        input = fileMan.readContents().split("\n")

        fileMan = FileManipulator("trainSVM.txt")
        fileMan.writeContents("")
        subsetFile = FileManipulator("subset.txt")
        subsetFile.writeContents("")
        counter = 0
        maxNum = 10000
        #input and filename
        for i in range(0, len(input)):
            values = input[i].split(",")
            if len(values) == 6:
                website = values[0]
                ageID = self.getAgeID(int(values[1]))
                genderID = int(values[2])
                countryID = countryDict.get(values[3])
                educLevelID = educLevelDict.get(values[4])
                label = values[5]
                binaryFeatures = BinaryFeature(ageID, genderID, countryID, educLevelID)
                featureVector47 = binaryFeatures.getFeatures()
                strToWrite = label + " "
                strToWrite += websiteFeatureDict.get(website)
                for j in range(0, len(featureVector47)):
                    strToWrite += str(j + 129) + ":" + str(featureVector47[j]) + " "
                strToWrite += "\n"
                fileMan.appendToFile(strToWrite)
                if(counter < maxNum):
                    subsetFile.appendToFile(strToWrite)
                counter += 1

        fileMan = FileManipulator("assets/datasets/" + dataset + "/validate.txt")
        input = fileMan.readContents().split("\n")

        fileMan = FileManipulator("trainSVM.txt")

        for i in range(0, len(input)):
            values = input[i].split(",")
            if len(values) == 6:
                website = values[0]
                ageID = self.getAgeID(int(values[1]))
                genderID = int(values[2])
                countryID = countryDict.get(values[3])
                educLevelID = educLevelDict.get(values[4])
                label = values[5]
                binaryFeatures = BinaryFeature(ageID, genderID, countryID, educLevelID)
                featureVector47 = binaryFeatures.getFeatures()
                strToWrite = label + " "
                strToWrite += websiteFeatureDict.get(website)
                for j in range(0, len(featureVector47)):
                    strToWrite += str(j + 129) + ":" + str(featureVector47[j]) + " "
                strToWrite += "\n"
                fileMan.appendToFile(strToWrite)

        cmd = 'python grid.py -log2c "{0}","{1}","{2}" -log2g "{3}","{4}","{5}" -svmtrain svm-train-gpu.exe -m




        print('Cross validation...')
        f = Popen(cmd, shell = True, stdout = PIPE).stdout

        line = ''
        while True:
            last_line = line
            line = f.readline()
            if not line: break
        c,g,rate = map(float,last_line.split())
        fileMan = FileManipulator("assets/models/" + modelName
                                  + "/bestSVMParams.txt")
        fileMan.writeContents("Cost," + str(c) + ",Gamma," + str(g)
                              + ",CV rate," + str(rate) + "%\n")

        print("Parameter selection done!")

    def startTrainSVM(self, cost, gamma, strToWrite):

        print("\nTraining SVM model...")
        print("Please wait, this may take a while.")

        modelName = self.name
        svmTrainPath = "assets/gpu_libsvm/svm-train-gpu.exe"
        modelSavePath = "assets/models/" + modelName + "/" + modelName + ".model"

        cmd = '"{0}" -c "{1}" -g "{2}" -m 500 trainSVM.txt "{3}"'.format(svmTrainPath,
                                                                        cost,
                                                                        gamma,
                                                                        modelSavePath)
        Popen(cmd, shell = True, stdout = PIPE).communicate()
        fileMan = FileManipulator("assets/models/" + modelName + "/svm.txt")
        fileMan.writeContents(strToWrite)

        fileMan = FileManipulator("assets/models/" + modelName + "/performance.txt")
        fileMan.writeContents("\n=== Performance ===\n")

        fileMan = FileManipulator("assets/app_guide.txt")
        fileMan.writeContents(self.getAppGuide(cost, gamma))
        print("SVM model training done!")

    def getAppGuide(self, cost, gamma):
        strToWrite = "Step 1: Click Load button\n"
        strToWrite += "  - File to load: trainSVM.txt\n"
        strToWrite += "  - If trainSVM.txt takes too long or crashes the app, try loading subset.txt\n\n"
        strToWrite += "Step 2: Copy the code below and paste it on the bottom-right of the app\n"
        strToWrite += "  Code: -c " + str(cost) + " -g " + str(gamma) + " -m 500\n"
        strToWrite += "  -c cost, -g gamma\n"
        strToWrite += "  -m cachesize : set cache memory size in MB (default 100)\n"
        strToWrite += "  If the app crashes, adjust -m accordingly\n\n"
        strToWrite += "Step 3: Click the Run button\n"
        strToWrite += "\n0 - Violet 1 - Light Blue\n"
        strToWrite += "Please note that this is just a visualization of the separating hyperplane.\n"

        return strToWrite

    def getAgeID(self, age):
        if age >= 12 and age <= 20:
            return 0
        elif age >= 21 and age <= 30:
            return 1
        elif age >= 31 and age <= 40:
            return 2
        elif age >= 41 and age <= 50:
            return 3
        elif age >= 51:
            return 4

        return -1

    def getDataset(self, modelName):
        datasetReader = FileManipulator("assets/models/" + modelName + "/details.txt")
        details = datasetReader.readContents()
        for line in details.split('\n'):
            content = line.split(': ')
            if content[0] == "Dataset used for training":
                return content[1]
        return ""

    def getTrainEpoch(self, modelName):
        trainReader = FileManipulator("assets/models/" + modelName + "/cnn.txt")
        details = trainReader.readContents()
        for line in details.split('\n'):
            content = line.split(': ')
            if content[0] == "Training epoch":
                return content[1]
        return ""

    def getBatchSize(self, modelName):
        batchReader = FileManipulator("assets/models/" + modelName + "/cnn.txt")
        details = batchReader.readContents()
        for line in details.split('\n'):
            content = line.split(': ')
            if content[0] == "Batch size":
                return content[1]
        return ""

    def getOptimizer(self, modelName):
        optReader = FileManipulator("assets/models/" + modelName + "/cnn.txt")
        details = optReader.readContents()
        optimizer = None
        numOfParam = 0
        lines = details.split('\n')
        for i in range(0, len(lines)):
            content = lines[i].split(': ')
            if content[0] == "Optimizer":
                optimizerName = content[1]
                numOfParam = self.getNumOfParams(optimizerName)
                count = i + 1
                max = numOfParam + i + 1
                inputList = []
                while (count < max):
                    content = lines[count].split(': ')
                    inputList.append(content[1])
                    count += 1

                return self.getOptObject(optimizerName, inputList)

        return None

    def getOptObject(self, optimizer, inputList):
        if optimizer == "Adadelta":
            lrInput = float(inputList[0])
            epsilonInput = float(inputList[1])
            decayInput = float(inputList[2])
            return Adagrad(lr=lrInput, epsilon=epsilonInput, decay=decayInput)

        elif optimizer == "Adagrad":
            lrInput = float(inputList[0])
            epsilonInput = float(inputList[1])
            decayInput = float(inputList[2])
            return Adagrad(lr=lrInput, epsilon=epsilonInput, decay=decayInput)

        elif optimizer == "Adam":
            lrInput = float(inputList[0])
            beta1Input = float(inputList[1])
            beta2Input = float(inputList[2])
            epsilonInput = float(inputList[3])
            decayInput = float(inputList[4])
            return Adam(lr=lrInput, beta_1=beta1Input, beta_2=beta2Input,
                        epsilon=epsilonInput, decay=decayInput)

        elif optimizer == "Adamax":
            lrInput = float(inputList[0])
            beta1Input = float(inputList[1])
            beta2Input = float(inputList[2])
            epsilonInput = float(inputList[3])
            decayInput = float(inputList[4])
            return Adamax(lr=lrInput, beta_1=beta1Input, beta_2=beta2Input,
                          epsilon=epsilonInput, decay=decayInput)

        elif optimizer == "Nadam":
            lrInput = float(inputList[0])
            beta1Input = float(inputList[1])
            beta2Input = float(inputList[2])
            epsilonInput = float(inputList[3])
            schedDecayInput = float(inputList[4])
            return Nadam(lr=lrInput, beta_1=beta1Input, beta_2=beta2Input,
                         epsilon=epsilonInput, schedule_decay=schedDecayInput)

        elif optimizer == "RMSProp":
            lrInput = float(inputList[0])
            rhoInput = float(inputList[1])
            epsilonInput = float(inputList[2])
            decayInput = float(inputList[3])
            return RMSprop(lr=lrInput, rho=rhoInput,
                           epsilon=epsilonInput, decay=decayInput)

        elif optimizer == "Stochastic Gradient Descent":
            lrInput = float(inputList[0])
            momentumInput = float(inputList[1])
            decayInput = float(inputList[2])
            nesterovInput = False
            if inputList[3] == "1":
                nesterovInput == True

            return SGD(lr=lrInput, momentum=momentumInput,
                       decay=decayInput, nesterov=nesterovInput)
        return None

    def getNumOfParams(self, optimizer):
        if optimizer == "Adadelta":
            return 4
        elif optimizer == "Adagrad":
            return 3
        elif optimizer == "Adam":
            return 5
        elif optimizer == "Adamax":
            return 5
        elif optimizer == "Nadam":
            return 5
        elif optimizer == "RMSProp":
            return 4
        elif optimizer == "Stochastic Gradient Descent":
            return 4
        return 0

    def getImagesAndLabelCNN(self, datasetPath):
        x = []
        y = []
        xVal = []
        yVal = []
        xTest = []
        yTest = []

        trainCtr = 0
        valCtr = 0
        testCtr = 0

        try:
            with open(datasetPath, 'r+', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        arrStr = line[0].split(",")
                        if len(arrStr) == 2:
                            websiteName = "assets/reinecke_datasets/" + arrStr[0] +".png"
                            rating = int(arrStr[1])
                            choice = self.getChoices(trainCtr, valCtr, testCtr)
                            if choice == 'train':
                                x.append(websiteName)
                                y.append(rating)
                                trainCtr += 1
                            elif choice == 'validate':
                                xVal.append(websiteName)
                                yVal.append(rating)
                                valCtr += 1
                            elif choice == 'test':
                                xTest.append(websiteName)
                                yTest.append(rating)
                                testCtr += 1

            return x, y, xVal, yVal, xTest, yTest
        except IOError:
            pass

        return [], [], [], [], []

    def getChoices(self, trainCtr, valCtr, testCtr):
        choices = []
        if trainCtr != 252:
            choices.append('train')
        if valCtr != 83:
            choices.append('validate')
        if testCtr != 83:
            choices.append('test')

        if len(choices) != 0:
            return random.choice(choices)

        return ""

    def getImagesPCA(self, datasetPath):
        images = []
        websiteNames = []
        try:
            with open(datasetPath, 'r+', encoding='utf-8') as csvfile:
                reader = csv.reader(csvfile, delimiter='\n')
                for line in reader:
                    if len(line) == 1:
                        arrStr = line[0].split(",")
                        if len(arrStr) == 2:
                            websiteNames.append(arrStr[0])
                            websiteName = "assets/reinecke_datasets/" + arrStr[0] +".png"
                            images.append(websiteName)


            images = preprocess_image_batch(images, img_size=(227,227),
                                            crop_size=(227,227), color_mode="rgb")

            return images, websiteNames
        except IOError:
            pass

        return [], []
```

A..15 LoadModel.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft
from CBoxHelper import CBoxHelper

from HybridModelTester import HybridModelTester

class LoadModel(tk.Frame):


    def __init__(self, parent, controller, cnnChoice, cnnDetails):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cnnChoice = cnnChoice
        self.cnnDetails = cnnDetails

        self.cnnCboxHelper = CBoxHelper("Please select a CNN model", "assets/cnn")
        self.cnnDict = self.cnnCboxHelper.getValuesDict()
        self.cnnDetailDict = self.cnnCboxHelper.getModelsDetails()

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.selectCNNLabel = tk.Label(self.parentFrame,
                                       text="Select the CNN model to load",
                                       font=ft.LARGE_FONT)
        self.selectCNNLabel.pack(padx=10, pady=5)

        self.cnnFrame = tk.Frame(self.parentFrame)
        self.cnnFrame.pack(fill="x")

        self.cnnLabel = tk.Label(self.cnnFrame, text="Model:", font=ft.NORMAL_FONT)
        self.cnnLabel.pack(fill="x", side=tk.LEFT, padx=(10, 35), pady=10)

        self.cnnCombo = ttk.Combobox(self.cnnFrame)
        self.cnnCombo['values'] = self.cnnCboxHelper.getValues()
        self.cnnCombo.config(state="readonly")
        self.cnnCombo.current(self.cnnChoice)
        self.cnnCombo.pack(fill="x", side=tk.LEFT, expand="True")
        self.cnnCombo.bind("<<ComboboxSelected>>", self.modelNewSelection)

        self.cnnDetailLabel = tk.Label(self.parentFrame, text="Model details:",
                                       font=ft.NORMAL_FONT)
        self.cnnDetailLabel.pack(anchor="w", padx=10)

        self.cnnDetailFrame = tk.Frame(self.parentFrame)
        self.cnnDetailFrame.pack()

        self.cnnDetailEntry = tkst.ScrolledText(self.cnnDetailFrame,
                                                height=7,
                                                wrap="word")

        self.cnnDetailEntry.insert(tk.INSERT, self.cnnDetails)

        self.cnnDetailEntry.config(state="disabled")

        self.cnnDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.evaluateBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                      command=self.proceedToPCATrain)
        self.evaluateBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

    def modelNewSelection(self, event):
        cnnChoice = self.cnnDict.get(self.cnnCombo.get())
        cnnDetails = self.cnnDetailDict.get(self.cnnCombo.get())

        self.controller.createLoadModel(cnnChoice, cnnDetails)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToPCATrain(self):
        cnnChoice = self.cnnCombo.get()
        baseCnnChoice = self.cnnCboxHelper.getBaseString()

        if cnnChoice != baseCnnChoice:
            self.controller.updateLoadModel(cnnChoice)
```

A..16 LogIn.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class LogIn(tk.Frame):

    def __init__(self, parent, controller, username, password, attempt):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.username = username
        self.password = password
        self.attempt = attempt

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.logInLabel = tk.Label(self.parentFrame, text="AI Expert Log In",
                                   font=ft.TITLE_FONT)
        self.logInLabel.pack(padx=10, pady=20)

        self.unameFrame = tk.Frame(self.parentFrame)
        self.unameFrame.pack(fill="x")

        self.unameLabel = tk.Label(self.unameFrame, text="Username:",
                                   font=ft.NORMAL_FONT)
        self.unameLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=15)

        self.unameEntry = tk.Entry(self.unameFrame)
        self.unameEntry.pack(fill="x", padx=10, expand="True")

        self.passFrame = tk.Frame(self.parentFrame)
        self.passFrame.pack(fill="x")

        self.passLabel = tk.Label(self.passFrame, text="Password:",
                                  font=ft.NORMAL_FONT)
        self.passLabel.pack(fill="x", side=tk.LEFT, padx=(10, 14), pady=15)
        self.passEntry = tk.Entry(self.passFrame)

        self.passEntry.config(show="*")
        self.passEntry.pack(fill="x", padx=10, expand="True")

        self.attemptsFrame = tk.Frame(self.parentFrame)
        self.attemptsFrame.pack(fill="x")

        self.attemptsEntry = tk.Label(self.attemptsFrame,
                                      text=self.attempt, font=ft.NORMAL_FONT)
        self.attemptsEntry.pack(fill="x", padx=(12, 8), pady=10 )

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.backBtn = ttk.Button(self.buttonFrame, text="Back",
                                  command= self.backToWelcome )
        self.backBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.logInBtn = ttk.Button(self.buttonFrame, text="Log in",
                                   command= self.updateLogIn )
        self.logInBtn.pack(side=tk.LEFT, ipady=5,  pady=10)

        self.signUpFrame = tk.Frame(self.parentFrame)
        self.signUpFrame.pack(fill="x")

        self.signUpLabel = tk.Label(self.signUpFrame,
                                    text="No account? Click the button below",
                                    font=ft.NORMAL_FONT)
        self.signUpLabel.pack(padx=10, pady=(20, 0))

        self.signUpBtn = ttk.Button(self.signUpFrame,
                                    text="Sign Up",
                                    command= self.proceedToSignUp )
        self.signUpBtn.pack(padx=15, ipady=5, pady=10)

    def backToWelcome(self):
        self.controller.createWelcomePage()

    def updateLogIn(self):
        self.controller.updateLogIn(self.unameEntry.get(),
                                    self.passEntry.get(),
                                    self.attempt)

    def proceedToSignUp(self):
        self.controller.createSULogIn("", "")
```

A..17 PCATrain.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from HybridModelTrainer import HybridModelTrainer
from FileManipulator import FileManipulator

class PCATrain(tk.Frame):

    def __init__(self, parent, controller, origFeatCont, reducFeatCont):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.origFeatCont = origFeatCont
        self.reducFeatCont = reducFeatCont

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.visualizationLabel = tk.Label(self.parentFrame,
                                text="Dimension Reduction using the PCA Model",
                                font=ft.LARGE_FONT)
        self.visualizationLabel.pack(padx=10, pady=10)

        self.origFeatureLabel = tk.Label(self.parentFrame,
                                         text="Original Features:",
                                         font=ft.NORMAL_FONT)
        self.origFeatureLabel.pack(anchor="w", padx=10)

        self.origFeatureFrame = tk.Frame(self.parentFrame)
        self.origFeatureFrame.pack()

        self.origFeatureEntry = tkst.ScrolledText(self.origFeatureFrame,
                                                  height=10,
                                                  wrap="word")

        self.origFeatureEntry.insert(tk.INSERT, self.origFeatCont)

        self.origFeatureEntry.config(state="disabled")

        self.origFeatureEntry.pack(fill="x", pady=(0, 20))

        self.reducFeatureLabel = tk.Label(self.parentFrame,
                                          text="Reduced Features:",
                                        font=ft.NORMAL_FONT)
        self.reducFeatureLabel.pack(anchor="w", padx=10)

        self.reducFeatureFrame = tk.Frame(self.parentFrame)
        self.reducFeatureFrame.pack()

        self.reducFeatureEntry = tkst.ScrolledText(self.reducFeatureFrame,
                                                    height=10,
                                                    wrap="word")

        self.reducFeatureEntry.insert(tk.INSERT, self.reducFeatCont)

        self.reducFeatureEntry.config(state="disabled")

        self.reducFeatureEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame,
                                text="Back to Home",
                                command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.proceedBtn = ttk.Button(self.buttonFrame,
                                    text="Proceed",
                                    command= self.proceedToSetSVMTrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToSetSVMTrain(self):
        self.controller.createSetParamSearch("-5", "15", "2", "3", "-15", "2", "")
```

A..18 PredictingVisualAppealApp.py

```
import matplotlib
import os
import datetime
matplotlib.use("TkAgg")

from matplotlib.backends.backend_tkagg import FigureCanvasTkAgg
from matplotlib.backends.backend_tkagg import NavigationToolbar2TkAgg
from matplotlib.figure import Figure
import matplotlib.animation as animation
from matplotlib import style

import tkinter as tk
from tkinter import ttk
from PIL import Image, ImageTk
from tkinter import filedialog as fd
import tkinter.scrolledtext as tkst

import urllib
import json

import pandas as pd
import numpy as np
import Font as ft
import shutil

from WelcomePage import WelcomePage
from HomePage import HomePage
from LogIn import LogIn
from SULogIn import SULogIn
from SignUp import SignUp
from SuccessCreateAccount import SuccessCreateAccount
from PredictVisualAppeal import PredictVisualAppeal
from CreateDataset import CreateDataset
from SuccessDataset import SuccessDataset
from CreateModel import CreateModel
from SelectDatasetTrain import SelectDatasetTrain
from SelectCNNTrain import SelectCNNTrain
from SetCNNTrain import SetCNNTrain
from CNNTrain import CNNTrain
from PCATrain import PCATrain
from SetParamSearch import SetParamSearch
from SetSVMTrain import SetSVMTrain
from SVMTrain import SVMTrain
from SuccessTraining import SuccessTraining
from CreateModelLoad import CreateModelLoad
from LoadModel import LoadModel
from SelectCNNTest import SelectCNNTest
from SuccessEvaluation import SuccessEvaluation
from UpdateModel import UpdateModel
from SuccessUpdateModel import SuccessUpdateModel

from FileManipulator import FileManipulator
from DatasetPartitioner import DatasetPartitioner
from FormValidator import FormValidator
from HybridModelTrainer import HybridModelTrainer

from VisualAppealPredictor import VisualAppealPredictor


style.use("ggplot")
```

```
figure = Figure(figsize=(10,2.5), dpi=100)
a = figure.add_subplot(111) #111 1x1 1 chart
a.plot([1,2,3,4,5,6,7,8], [2,3,5,7,2,7,5,4])

def center_window(toplevel):
    toplevel.update_idletasks()
    width = toplevel.winfo_screenwidth()
    height = toplevel.winfo_screenheight()
    size = tuple(int(_) for _ in toplevel.geometry().split('+')[0].split('x'))
    x = width / 2 - size[0] / 2
    y = height / 2 - size[1] / 2
    toplevel.geometry("%dx%d+%d+%d" % (size + (x, y)))

def popupmsg(title, msg, window_size):
    popup = tk.Tk()
    popup.wm_title(title)
    popup.geometry(window_size)
    popup.resizable(height="False", width="False")
    center_window(popup)
    label = ttk.Label(popup, text=msg, font=ft.LARGE_FONT, justify=tk.CENTER)
    label.pack(pady=10)
    closeBtn = ttk.Button(popup, text="Close", command = popup.destroy)
    closeBtn.pack()

    popup.mainloop()

class PredictVisualAppealApp(tk.Tk):

    def __init__(self, *args, **kwargs):
        tk.Tk.__init__(self, *args, **kwargs)
        tk.Tk.iconbitmap(self, default="assets/images/upmlogo.ico")
        tk.Tk.wm_title(self, "Predicting Website's Visual Appeal Using Hybrid CNN-SVM")

        self.container = tk.Frame(self)
        self.container.pack(side="top", fill="both", expand=True)
        self.container.grid_rowconfigure(0, weight=1)
        self.container.grid_columnconfigure(0, weight=1)

        self.currentFrame = None
        self.currentAIExpert = None
        self.modelToCreate = None

        menubar  = tk.Menu(self.container)

        infoMenu = tk.Menu(menubar, tearoff=0)
        infoMenu.add_separator()
        infoMenu.add_command(label="Model used by the system", command = self.createInfoMenuContent)

        menubar.add_cascade(label="More Information", menu=infoMenu)

        tk.Tk.config(self, menu=menubar)

        self.createWelcomePage()


    def createInfoMenuContent(self):

        title = "Hybrid CNN-SVM Model Information"
        name, details = self.getCurrentModel()

        currModelLabel = "Name of model used by the system:\n\n" + name

        popup = tk.Tk()

        popup.wm_title(title)
        popup.geometry("720x300")
        popup.resizable(height="False", width="False")
        center_window(popup)
        label = ttk.Label(popup, text=currModelLabel,
                          font=ft.LARGE_FONT, justify=tk.CENTER)
        label.pack(pady=10)

        cnnDetailLabel = tk.Label(popup, text="Model details:", font=ft.NORMAL_FONT)
        cnnDetailLabel.pack(anchor="w", padx=30)

        cnnDetailFrame = tk.Frame(popup)
        cnnDetailFrame.pack()

        cnnDetailEntry = tkst.ScrolledText(cnnDetailFrame, height=8, wrap="word")
        cnnDetailEntry.insert(tk.INSERT, details)
        cnnDetailEntry.config(state="disabled")
        cnnDetailEntry.pack(fill="x", pady=(0, 15))

        closeBtn = ttk.Button(popup, text="Close", command = popup.destroy)
        closeBtn.pack()
        popup.mainloop()

    def createDeleteWarning(self):

        title = "Warning"
        popup = tk.Tk()

        popup.wm_title(title)
        popup.geometry("520x200")
        popup.resizable(height="False", width="False")
        center_window(popup)
        label1 = ttk.Label(popup,
                        text="Going back without finishing all the steps for",
                        font=ft.LARGE_FONT,
                    justify=tk.CENTER)
        label1.pack(pady=5)

        label2 = ttk.Label(popup,
                            text="training will delete all your progress.",
                            font=ft.LARGE_FONT,
                            justify=tk.CENTER)
        label2.pack(pady=5)

        label3 = ttk.Label(popup, text="Are you sure?",
                            font=ft.LARGE_FONT, justify=tk.CENTER)
        label3.pack(pady=5)

        self.parentFrame = tk.Frame(popup)
        self.parentFrame.pack(expand="True")

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.yesBtn = ttk.Button(self.buttonFrame, text="Yes, I am sure",
                                  command= lambda: self.deleteModel(popup))
        self.yesBtn.pack(side=tk.LEFT, padx=15, ipady=6, pady=5)

        self.noBtn = ttk.Button(self.buttonFrame, text="No, go back",
                                 command= popup.destroy)
        self.noBtn.pack(side=tk.LEFT, padx=15, ipady=6, pady=5)

        popup.mainloop()

    def deleteModel(self, popup):
        popup.destroy()
        self.createHomePage()

    def createHelpMenuContent(self):
        title = "User Manual"
        message = "Insert image here of user manual"
        size = "720x640"
        popupmsg(title, message, size)

    def updateCurrentFrame(self, frame):
        if self.currentFrame != None:
            self.currentFrame.destroy()

        self.currentFrame = frame
        self.currentFrame.tkraise()

    def createWelcomePage(self):
        frame = WelcomePage(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.currentAIExpert = None
        self.updateCurrentFrame(frame)

    def createHomePage(self):
        frame = HomePage(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        if self.modelToCreate != None:
            shutil.rmtree("assets/models/" + self.modelToCreate)

        self.modelToCreate = None
        self.updateCurrentFrame(frame)

    def createLogIn(self, username, password, attempt):
        frame = LogIn(self.container, self, username, password, attempt)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSULogIn(self, password, attempt):
        frame = SULogIn(self.container, self, password, attempt)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSignUp(self, username, password, confPass, agreedToTerms, errorMsg):
        frame = SignUp(self.container, self, username, password,
                       confPass, agreedToTerms, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSuccessCreateAccount(self, username):
        frame = SuccessCreateAccount(self.container, self, username)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createPredictVisualAppeal(self, fileStr, ageInt, genderInt,
                                  countryInt, educLevelInt, predictStr):
        frame = PredictVisualAppeal(self.container, self, fileStr, ageInt,
                                    genderInt, countryInt, educLevelInt, predictStr)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createCreateDataset(self, nameStr, trainingInt,
                            validationInt, testInt, errorMsg):
        frame = CreateDataset(self.container, self, nameStr, trainingInt,
                              validationInt, testInt, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSuccessDataset(self, nameStr, trainingInt, validationInt, testInt):
        frame = SuccessDataset(self.container, self, nameStr,
                               trainingInt, validationInt, testInt)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createCreateModel(self, nameStr, errorMsg):
        frame = CreateModel(self.container, self, nameStr, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createCreateModelLoad(self, nameStr, errorMsg):
        frame = CreateModelLoad(self.container, self, nameStr, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSelectDatasetTrain(self, datasetChoice, datasetDetails):
        frame = SelectDatasetTrain(self.container, self, datasetChoice,
                                   datasetDetails)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSelectCNNTrain(self):
        frame = SelectCNNTrain(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSetCNNTrain(self, trainEpoch, batchSize, optChoice,
                          hyperList, hyperEntry, errorMsg):
        frame = SetCNNTrain(self.container, self, trainEpoch, batchSize,
                            optChoice, hyperList, hyperEntry, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createCNNTrain(self):
        frame = CNNTrain(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createPCATrain(self, origFeatCont, reducFeatCont):
        frame = PCATrain(self.container, self, origFeatCont, reducFeatCont)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSetParamSearch(self, cStart, cEnd, cStep, gStart, gEnd, gStep, errorMsg):
        frame = SetParamSearch(self.container, self, cStart, cEnd,
                               cStep, gStart, gEnd, gStep, errorMsg)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSetSVMTrain(self):
        frame = SetSVMTrain(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSVMTrain(self):
        frame = SVMTrain(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSuccessTraining(self):
        frame = SuccessTraining(self.container, self)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createLoadModel(self, cnnChoice, cnnDetails):
        frame = LoadModel(self.container, self, cnnChoice, cnnDetails)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSelectCNNTest(self, cnnChoice, cnnDetails,
                            datasetChoice, datasetDetails):
        frame = SelectCNNTest(self.container, self, cnnChoice,
                              cnnDetails, datasetChoice, datasetDetails)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSuccessEvaluation(self, cnnChoice, datasetChoice):
        frame = SuccessEvaluation(self.container, self, cnnChoice, datasetChoice)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createUpdateModel(self, cnnChoice, cnnDetails):
        frame = UpdateModel(self.container, self, cnnChoice, cnnDetails)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def createSuccessUpdateModel(self, cnnChoice, userAIExpert):
        frame = SuccessUpdateModel(self.container, self, cnnChoice, userAIExpert)
        frame.grid(row=0, column=0, sticky="nsew")
        self.updateCurrentFrame(frame)

    def showFrame(self, frame):
        frameToRaise = self.GlobalFrames.get(frame)
        frameToRaise.tkraise() #raise on front
        if frame == "HomePage":
            print(self.getUserAIExpert())

    def updatePredictVisualAppeal(self, fileStr, ageInt, genderInt, countryInt,
                                  educLevelInt, predictStr, predict=False, img=None):

        if predict:
            print("\nPerforming prediction...")
            print("Please wait, this may take a while.")
            currModelName, detail = self.getCurrentModel()
            vsp = VisualAppealPredictor(fileStr, ageInt, genderInt, countryInt,
                                        educLevelInt, currModelName)
            predictStr = vsp.getPrediction()
            print("Prediction done!")

        else:
            self.file_opt = options = {}
            options['defaultextension'] = ''
            options['filetypes'] = [('Images', '.png'), ('Images', '.jpg'),
                                   ('Images', '.jpeg')]
            options['initialdir'] = ''
            options['initialfile'] = ''
            options['parent'] = self
            options['title'] = 'Choose a file'
            fileStr = fd.askopenfilename(**self.file_opt)
            predictStr = ""

        self.createPredictVisualAppeal(fileStr, ageInt, genderInt,
                                       countryInt, educLevelInt, predictStr)

    def updateLogIn(self, username, password, attempt):

        fileObj = FileManipulator("assets/accounts.txt")
        userExists = fileObj.checkUserPass(username, password)
        if userExists:

            self.currentAIExpert = username
            self.createHomePage()
        else:
            self.createLogIn(username, password, "Incorrect username and/or password")

    def updateSULogIn(self, password):

        fileObj = FileManipulator("")
        correctSUPass = fileObj.checkSUPass(password)
        if correctSUPass:
            self.createSignUp("", "", "", 0, "")
        else:
            self.createSULogIn(password, "Incorrect superuser password")

    def createAccount(self, username, password, confPass, agreedToTerms):

        formObj = FormValidator()

        errorMsg = formObj.checkNullInput([username, password, confPass])
        errorMsg += formObj.checkLength("Username", username)
        errorMsg += formObj.checkLength("Password", password)
        errorMsg += formObj.checkAgreeTerms(agreedToTerms)
        errorMsg += formObj.checkUniqueUsername(username)
        errorMsg += formObj.checkPasswordsEquality(password, confPass)

        if errorMsg == "":
            fileObj = FileManipulator("assets/accounts.txt")
            fileObj.appendToAccounts(username, password)
            self.createSuccessCreateAccount(username)
        else:
            self.createSignUp(username, password, confPass, agreedToTerms, errorMsg)

    def updateSuccessDataset(self, nameStr, trainingInt, validationInt, testInt):

        formObj = FormValidator()

        errorMsg = formObj.checkNullInput([nameStr, trainingInt, validationInt, testInt])
        errorMsg += formObj.checkLength("Dataset name", nameStr)
        errorMsg += formObj.checkNumericInput(trainingInt, validationInt, testInt)
        errorMsg += formObj.checkUniqueDataset(nameStr)

        if errorMsg == "":
            print("\nPartitioning the dataset...")
            print("Please wait, this may take a while.")

            totalPartition = int(trainingInt) + int(validationInt) + int(testInt)
            totalData = 436631

            if totalPartition == 100:
                trainingInt = int(totalData * (float(trainingInt) / 100))
                validationInt = int(totalData * (float(validationInt) / 100))
                testInt = totalData - (trainingInt + validationInt)
            else:
                trainingInt = int(totalData * (float(trainingInt) / 100))
                validationInt = int(totalData * (float(validationInt) / 100))
                testInt = int(totalData *(float(testInt) / 100))

            partitioner = DatasetPartitioner(nameStr, trainingInt, validationInt,
                                             testInt, self.getCurrentAIExpert())
            success = partitioner.partitionTheDataset()
            if success:
                self.createSuccessDataset(nameStr, trainingInt, validationInt, testInt)
                print("Partitioning done!")
            else:
                self.createCreateDataset("", 0, 0, 0, "- An error occurred during creation of dataset.\n")


        else:
            self.createCreateDataset(nameStr, trainingInt, validationInt, testInt, errorMsg)
```

```
def updateCreateModel(self, nameStr):

    formObj = FormValidator()
    errorMsg = formObj.checkNullInput([nameStr])
    errorMsg += formObj.checkLength("Model name", nameStr)
    errorMsg += formObj.checkUniqueModel(nameStr)

    if errorMsg == "":
        self.modelToCreate = nameStr
        os.mkdir("assets/models/" + self.modelToCreate)
        fileObj = FileManipulator("assets/models/" + self.modelToCreate + "/details.txt")
        fileObj.appendToFile("Model name: " + self.modelToCreate +"\n")
        fileObj.appendToFile("AI Expert: " + self.getCurrentAIExpert()+"\n")
        self.createSelectDatasetTrain(0, "")
    else:
        self.createCreateModel(nameStr, errorMsg)

def updateCreateModelLoad(self, nameStr):
    formObj = FormValidator()
    errorMsg = formObj.checkNullInput([nameStr])
    errorMsg += formObj.checkLength("Model name", nameStr)
    errorMsg += formObj.checkUniqueModel(nameStr)

    if errorMsg == "":
        self.modelToCreate = nameStr
        os.mkdir("assets/models/" + self.modelToCreate)
        self.createLoadModel(0, "")
    else:
        self.createCreateModelLoad(nameStr, errorMsg)

def updateSelectDatasetTrain(self, dataset):
    fileObj = FileManipulator("assets/models/" + self.modelToCreate + "/details.txt")
    fileObj.appendToFile("Dataset used for training: " + dataset + "\n")
    self.createSelectCNNTrain()

def updateSelectCNNTrain(self, architecture):
    fileObj = FileManipulator("assets/models/" + self.modelToCreate + "/cnn.txt")
    fileObj.appendToFile("\n=== CNN Details ===\n")
    fileObj.appendToFile("Architecture: " + architecture + "\n")
    self.createSetCNNTrain("", "", 0, [], {}, "")

def updateSetCNNTrain(self, trainEpoch, batchSize, optChoice, optStr, hyperList, hyperEntry):
    formObj = FormValidator()

    errorMsg = formObj.checkNullInput([trainEpoch, batchSize])
    errorMsg += formObj.checkNullInputDict(hyperList, hyperEntry)
    errorMsg += formObj.checkHyperInput(hyperList, hyperEntry)
    errorMsg += formObj.checkEpochBatch(trainEpoch, batchSize)

    if errorMsg == "":
        fileObj = FileManipulator("assets/models/" + self.modelToCreate + "/cnn.txt")
        fileObj.appendToFile("Training epoch: " + trainEpoch + "\n")
        fileObj.appendToFile("Batch size: " + batchSize + "\n")
        fileObj.appendToFile("Optimizer: " + optStr + "\n")
        for param in hyperList:
            label = param[0]
            value = hyperEntry.get(label)
            fileObj.appendToFile(label + ": " + value + "\n")

        #
        trainer = HybridModelTrainer(self.getModelToCreate())
        trainer.startTrainCNN()
        os.mkdir("assets/cnn/" + self.modelToCreate)
        shutil.copyfile("assets/models/" + self.modelToCreate
                        + "/details.txt", "assets/cnn/"
                        + self.modelToCreate +"/details.txt")
        shutil.copyfile("assets/models/" + self.modelToCreate
                        + "/cnn.txt", "assets/cnn/"
                        + self.modelToCreate +"/cnn.txt")
        shutil.copyfile("assets/models/" + self.modelToCreate
                        + "/trainValLosses.txt", "assets/cnn/"
                        + self.modelToCreate +"/trainValLosses.txt")
        shutil.copyfile("assets/models/" + self.modelToCreate +"/"
                        + self.modelToCreate + ".h5", "assets/cnn/"
                        + self.modelToCreate +"/" + self.modelToCreate + ".h5")

        self.createCNNTrain()
    else:
        self.createSetCNNTrain(trainEpoch, batchSize, optChoice, hyperList, hyperEntry, errorMsg)

def updateLoadModel(self, cnnChoice):
    self.cnnToLoad = cnnChoice
    shutil.copyfile("assets/cnn/" + self.cnnToLoad
                    + "/details.txt", "assets/models/"
                    + self.modelToCreate +"/details.txt")

    shutil.copyfile("assets/cnn/" + self.cnnToLoad
                    + "/cnn.txt", "assets/models/"
                    + self.modelToCreate + "/cnn.txt")

    shutil.copyfile("assets/cnn/" + self.cnnToLoad
                    + "/trainValLosses.txt", "assets/models/"
                    + self.modelToCreate +"/trainValLosses.txt")

    shutil.copyfile("assets/cnn/" + self.cnnToLoad
                    + "/" + self.cnnToLoad + ".h5", "assets/models/"
                    + self.modelToCreate +"/" + self.modelToCreate + ".h5")

    fileMan = FileManipulator("assets/models/" + self.modelToCreate + "/details.txt")
```

```
        strToWrite = fileMan.readContents()
        strToWrite = "Model Name: " + self.modelToCreate + "\nAI Expert: "
                    + self.getCurrentAIExpert() + "\nLoaded CNN " + strToWrite
        fileMan.writeContents(strToWrite)
        self.createCNNTrain()

    def updateSetParamSearch(self, cStart, cEnd, cStep, gStart, gEnd, gStep):
        formObj = FormValidator()

        errorMsg = formObj.checkNullInput([cStart, cEnd, cStep, gStart, gEnd, gStep])
        errorMsg += formObj.isValidCandG([cStart, cEnd, gStart, gEnd])
        errorMsg += formObj.isValidStep([cStep, gStep])
        errorMsg += formObj.isEqualC(cStart, cEnd)
        errorMsg += formObj.isEqualG(gStart, gEnd)

        if errorMsg == "":
            if (float(cStart) > float(cEnd)):
                cStep = "-" + cStep
            if (float(gStart) > float(gEnd)):
                gStep = "-" + cStep
            modelName = self.getModelToCreate()
            trainer = HybridModelTrainer(modelName)
            trainer.startParamSearch(cStart, cEnd, cStep, gStart, gEnd, gStep)
            self.createSetSVMTrain()

        else:
            self.createSetParamSearch(cStart, cEnd, cStep, gStart, gEnd, gStep, errorMsg)

    def getCurrentModel(self):
        name = ""
        AIExpert = ""
        details = ""
        fileMan = FileManipulator("assets/current_model.txt")
        arrStr = fileMan.readContents().split('\n')

        if len(arrStr) == 3:
            name = arrStr[0]
            AIExpert = arrStr[1]
            fileMan = FileManipulator("assets/models/" + name + "/details.txt")
            details = AIExpert + "\n" + fileMan.readContents()
            fileMan = FileManipulator("assets/models/" + name + "/cnn.txt")
            details += fileMan.readContents()
            fileMan = FileManipulator("assets/models/" + name + "/pca.txt")
            details += fileMan.readContents()
            fileMan = FileManipulator("assets/models/" + name + "/svm.txt")
            details += fileMan.readContents()
            fileMan = FileManipulator("assets/models/" + name + "/performance.txt")
            details += fileMan.readContents()

        return name, details

    def updateCurrentModel(self, cnnChoice, userAIExpert):
        fileMan = FileManipulator("assets/current_model.txt")
        content = cnnChoice + "\n" + "Updated by: " + userAIExpert + " on "
                  + datetime.datetime.now().strftime("%A, %d %B %Y %I:%M%p") + "\n"
        fileMan.writeContents(content)

    def getCurrentAIExpert(self):
        return self.currentAIExpert

    def getModelToCreate(self):
        return self.modelToCreate


app = PredictVisualAppealApp()
app.geometry("720x720")
app.resizable(height="False", width="False")
center_window(app)
app.mainloop()
```

A..19 PredictVisualAppeal.py

```
import tkinter as tk
from tkinter import ttk #css
from PIL import Image, ImageTk
from tkinter import filedialog as fd
import Font as ft

class PredictVisualAppeal(tk.Frame):

    def __init__(self, parent, controller, fileStr, ageInt,
                 genderInt, countryInt, educLevelInt, predictStr):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.fileStr = fileStr
        self.ageInt = ageInt
        self.genderInt = genderInt
        self.countryInt = countryInt
        self.educLevelInt = educLevelInt
        self.predictStr = predictStr

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
```

```
self.label2.pack(ipady=5, fill="x")

self.parentFrame = tk.Frame(self)
self.parentFrame.pack(expand="True")

self.uploadLabel = tk.Label(self.parentFrame,
                            text="Upload a picture of a website",
                            font=ft.LARGE_FONT)
self.uploadLabel.pack(padx=10, pady=10)

self.uploadFrame = tk.Frame(self.parentFrame)
self.uploadFrame.pack(fill="x")

self.uploadBtn = ttk.Button(self.uploadFrame, text="Choose file",
                            command=lambda: self.updatePredictVisualAppeal("chooseFile"))
self.uploadBtn.pack(side=tk.LEFT, padx=10, pady=10)

self.fileEntry = tk.Entry(self.uploadFrame)
self.fileEntry.insert(0, self.fileStr)
self.fileEntry.config(state="readonly", readonlybackground='white')

self.fileEntry.pack(fill="x", padx=10, expand="True")

#insert image here
if self.fileStr:
    self.img = Image.open(self.fileStr)
else:
    self.img = Image.open("assets/images/upmlogo.png")

width, height = self.img.size
if width < 227 or height < 227:
    self.fileStr = "assets/images/227image.png"
    self.img = Image.open(self.fileStr)

self.img = self.img.resize((227, 227), Image.ANTIALIAS)
self.photo = ImageTk.PhotoImage(self.img)

self.imgPanel = tk.Label(self.parentFrame, image = self.photo,
                         width = 227, height = 227)
self.imgPanel.image = self.photo
self.imgPanel.pack()

self.selectLabel = tk.Label(self.parentFrame,
        text="Select the demographic backgrounds of the target users",
        font=ft.LARGE_FONT)
self.selectLabel.pack(padx=10, pady=10)

self.ageFrame = tk.Frame(self.parentFrame)
self.ageFrame.pack()

self.ageLabel = tk.Label(self.ageFrame, text="Age:", font=ft.NORMAL_FONT)
self.ageLabel.pack(fill="x", side=tk.LEFT, padx=(110,10), pady=5)

self.ageCombo = ttk.Combobox(self.ageFrame, width=40)
self.ageCombo['values'] = ("12 - 20 years old",
                           "21 - 30 years old",
                           "31 - 40 years old",
                           "41 - 50 years old",
                           "51 years old onwards")
self.ageCombo.config(state="readonly")
self.ageCombo.current(self.ageInt)
self.ageCombo.pack(fill="x", padx=10, expand="True")

self.genderFrame = tk.Frame(self.parentFrame)
self.genderFrame.pack()

self.genderLabel = tk.Label(self.genderFrame, text="Gender:", font=ft.NORMAL_FONT)
self.genderLabel.pack(fill="x", side=tk.LEFT, padx=(87,10), pady=5)

self.genderCombo = ttk.Combobox(self.genderFrame, width=40)
self.genderCombo['values'] = ("Male", "Female")
self.genderCombo.config(state="readonly")
self.genderCombo.current(self.genderInt)
self.genderCombo.pack(fill="x", padx=10, expand="True")

self.countryFrame = tk.Frame(self.parentFrame)
self.countryFrame.pack()

self.countryLabel = tk.Label(self.countryFrame, text="Country:", font=ft.NORMAL_FONT)
self.countryLabel.pack(fill="x", side=tk.LEFT, padx=(80,10), pady=5)

self.countryCombo = ttk.Combobox(self.countryFrame, width=40)
self.countryCombo['values'] = ("Argentina", "Australia", "Austria",
                               "Belgium", "Brazil", "Bulgaria",
                               "Canada", "Chile", "Denmark", "Finland",
                               "France", "Germany", "Greece",
                               "Hungary", "India", "Ireland",
                               "Israel", "Italy", "Lithuania",
                               "Macedonia The Former Yugoslav Republic Of",
                               "Mexico", "Netherlands", "New Zealand",
                               "Norway", "Poland", "Portugal",
                               "Romania", "Serbia", "Singapore",
                               "Spain", "Sweden", "United Kingdom",
                               "United States")
self.countryCombo.config(state="readonly")
self.countryCombo.current(self.countryInt)
self.countryCombo.pack(fill="x", padx=10, expand="True")

self.educFrame = tk.Frame(self.parentFrame)
```

```
        self.educFrame.pack()

        self.educLabel = tk.Label(self.educFrame, text="Educational Level:",
                                  font=ft.NORMAL_FONT)
        self.educLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=5)

        self.educCombo = ttk.Combobox(self.educFrame, width=40)
        self.educCombo['values'] = ("Pre-high school", "High school", "College",
                                    "Professional School", "Graduate School",
                                    "PhD", "Postdoctoral")
        self.educCombo.config(state="readonly")
        self.educCombo.current(self.educLevelInt)
        self.educCombo.pack(fill="x", side=tk.RIGHT, padx=10, expand="True")

        self.predictionFrame = tk.Frame(self.parentFrame)
        self.predictionFrame.pack()

        self.predictionLabel = tk.Label(self.predictionFrame, text=self.predictStr, font=ft.LARGE_FONT)
        self.predictionLabel.pack(side=tk.LEFT, padx=10, pady=5)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back", command= self.backToWelcome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.buttonPredict = ttk.Button(self.buttonFrame,
                                        text="Predict Visual Appeal",
                                        command=lambda: self.updatePredictVisualAppeal("predict"))

        if self.fileStr == "assets/images/227image.png" or self.fileStr == ""
           or self.fileStr == "assets/images/upmlogo.png":
            self.buttonPredict.config(state=tk.DISABLED)

        self.buttonPredict.pack(ipady = 5, pady=(10,5))

    def backToWelcome(self):
        self.controller.createWelcomePage()

    def updatePredictVisualAppeal(self, btnCaller):
        ageDict = {"12 - 20 years old" : 0, "21 - 30 years old" : 1,
                   "31 - 40 years old" : 2, "41 - 50 years old" : 3,
                   "51 years old onwards" : 4}

        genderDict = {"Male" : 0, "Female" : 1}

        countryDict = {"Argentina" : 0, "Australia" : 1, "Austria" : 2,
                       "Belgium" : 3, "Brazil" : 4, "Bulgaria" : 5,
                       "Canada" : 6, "Chile" : 7, "Denmark" : 8,
                       "Finland" : 9, "France" : 10, "Germany" : 11,
                       "Greece" : 12, "Hungary" : 13, "India" : 14,
                       "Ireland" : 15, "Israel" : 16, "Italy" : 17,
                       "Lithuania" : 18,
                       "Macedonia The Former Yugoslav Republic Of" : 19,
                       "Mexico" : 20, "Netherlands" : 21, "New Zealand" : 22,
                       "Norway" : 23, "Poland" : 24, "Portugal" : 25,
                       "Romania" : 26, "Serbia" : 27, "Singapore" : 28,
                       "Spain" : 29, "Sweden" : 30, "United Kingdom" : 31,
                       "United States" : 32}

        educLevelDict = {"Pre-high school" : 0, "High school" : 1, "College" : 2,
                         "Professional School" : 3, "Graduate School" : 4,
                         "PhD" : 5, "Postdoctoral" : 6}

        fileStr = self.fileEntry.get()

        ageInt = ageDict.get(self.ageCombo.get())
        genderInt = genderDict.get(self.genderCombo.get())
        countryInt = countryDict.get(self.countryCombo.get())
        educLevelInt = educLevelDict.get(self.educCombo.get())
        predictStr = ""
        predict = None
        img = None

        if btnCaller == "chooseFile":
            predict = False
            img = None

        elif btnCaller == "predict":
            predict = True
            img = self.img

        self.controller.updatePredictVisualAppeal(fileStr, ageInt, genderInt,
                                                  countryInt, educLevelInt, predictStr,
                                                  predict, img)
```

A..20 SelectCNNTest.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft
from CBoxHelper import CBoxHelper

from HybridModelTester import HybridModelTester

class SelectCNNTest(tk.Frame):
```

```
def __init__(self, parent, controller, cnnChoice, cnnDetails,
             datasetChoice, datasetDetails):
    tk.Frame.__init__(self, parent)

    self.controller = controller
    self.cnnChoice = cnnChoice
    self.cnnDetails = cnnDetails
    self.datasetChoice = datasetChoice
    self.datasetDetails = datasetDetails

    self.cnnCboxHelper = CBoxHelper("Please select a model", "assets/models")
    self.cnnDict = self.cnnCboxHelper.getValuesDict()
    self.cnnDetailDict = self.cnnCboxHelper.getModelsDetails()

    self.datasetCboxHelper = CBoxHelper("Please select a dataset",
                                        "assets/datasets")
    self.datasetDict = self.datasetCboxHelper.getValuesDict()
    self.datasetDetailDict = self.datasetCboxHelper.getValuesDetails()

    self.label1 = tk.Label(self, text="Evaluate the Performance",
                           bg="white", font=ft.TITLE_FONT)
    self.label1.pack(ipady=5, fill="x")

    self.label2 = tk.Label(self, text="of a Hybrid CNN-SVM Model",
                           bg="white", font=ft.TITLE_FONT)
    self.label2.pack(ipady=5, fill="x")

    self.parentFrame = tk.Frame(self)
    self.parentFrame.pack(expand="True")

    self.selectCNNLabel = tk.Label(self.parentFrame,
                          text="Select the hybrid CNN-SVM model to evaluate",
                          font=ft.LARGE_FONT)
    self.selectCNNLabel.pack(padx=10, pady=5)

    self.cnnFrame = tk.Frame(self.parentFrame)
    self.cnnFrame.pack(fill="x")

    self.cnnLabel = tk.Label(self.cnnFrame, text="Model:", font=ft.NORMAL_FONT)
    self.cnnLabel.pack(fill="x", side=tk.LEFT, padx=(10, 35), pady=10)

    self.cnnCombo = ttk.Combobox(self.cnnFrame)
    self.cnnCombo['values'] = self.cnnCboxHelper.getValues()
    self.cnnCombo.config(state="readonly")
    self.cnnCombo.current(self.cnnChoice)
    self.cnnCombo.pack(fill="x", side=tk.LEFT, expand="True")
    self.cnnCombo.bind("<<ComboboxSelected>>", self.modelNewSelection)

    self.cnnDetailLabel = tk.Label(self.parentFrame, text="Model details:",
                                   font=ft.NORMAL_FONT)
    self.cnnDetailLabel.pack(anchor="w", padx=10)

    self.cnnDetailFrame = tk.Frame(self.parentFrame)
    self.cnnDetailFrame.pack()

    self.cnnDetailEntry = tkst.ScrolledText(self.cnnDetailFrame,
                                            height=7, wrap="word")

    self.cnnDetailEntry.insert(tk.INSERT, self.cnnDetails)

    self.cnnDetailEntry.config(state="disabled")

    self.cnnDetailEntry.pack(fill="x", pady=(0, 20))

    self.selectDatasetLabel = tk.Label(self.parentFrame,
                                       text="Select the dataset to use",
                                       font=ft.LARGE_FONT)
    self.selectDatasetLabel.pack(padx=10, pady=5)

    self.datasetFrame = tk.Frame(self.parentFrame)
    self.datasetFrame.pack(fill="x")

    self.datasetLabel = tk.Label(self.datasetFrame, text="Dataset:",
                                 font=ft.NORMAL_FONT)
    self.datasetLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

    self.datasetCombo = ttk.Combobox(self.datasetFrame)
    self.datasetCombo['values'] = self.datasetCboxHelper.getValues()
    self.datasetCombo.config(state="readonly")
    self.datasetCombo.current(self.datasetChoice)
    self.datasetCombo.pack(fill="x", side=tk.LEFT, padx=10, expand="True")
    self.datasetCombo.bind("<<ComboboxSelected>>", self.datasetNewSelection)

    self.datasetDetailLabel = tk.Label(self.parentFrame,
                                       text="Dataset details:",
                                       font=ft.NORMAL_FONT)
    self.datasetDetailLabel.pack(anchor="w", padx=10)

    self.datasetDetailFrame = tk.Frame(self.parentFrame)
    self.datasetDetailFrame.pack()

    self.datasetDetailEntry = tkst.ScrolledText(self.datasetDetailFrame,
                                                height=7, wrap="word")

    self.datasetDetailEntry.insert(tk.INSERT, self.datasetDetails)

    self.datasetDetailEntry.config(state="disabled")
```

```
        self.datasetDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.evaluateBtn = ttk.Button(self.buttonFrame, text="Evaluate Performance",
                                      command=self.proceedToSuccessEvaluation)
        self.evaluateBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

    def modelNewSelection(self, event):
        cnnChoice = self.cnnDict.get(self.cnnCombo.get())
        cnnDetails = self.cnnDetailDict.get(self.cnnCombo.get())
        datasetChoice = self.datasetDict.get(self.datasetCombo.get())
        datasetDetails = self.datasetDetailEntry.get(1.0, tk.END + "-1c")

        self.controller.createSelectCNNTest(cnnChoice, cnnDetails,
                                            datasetChoice, datasetDetails)

    def datasetNewSelection(self, event):
        cnnChoice = self.cnnDict.get(self.cnnCombo.get())
        cnnDetails = self.cnnDetailEntry.get(1.0, tk.END + "-1c")
        datasetChoice = self.datasetDict.get(self.datasetCombo.get())
        datasetDetails = self.datasetDetailDict.get(self.datasetCombo.get())

        self.controller.createSelectCNNTest(cnnChoice, cnnDetails,
                                            datasetChoice, datasetDetails)

    def backToHome(self):
        self.controller.createHomePage()

    def proceedToSuccessEvaluation(self):
        cnnChoice = self.cnnCombo.get()
        datasetChoice = self.datasetCombo.get()
        baseCnnChoice = self.cnnCboxHelper.getBaseString()
        baseDatasetChoice = self.datasetCboxHelper.getBaseString()

        if cnnChoice != baseCnnChoice and datasetChoice != baseDatasetChoice:
            tester = HybridModelTester(cnnChoice, datasetChoice)
            tester.startTest()
            self.controller.createSuccessEvaluation(cnnChoice, datasetChoice)
```

A..21 SelectCNNTrain.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from FileManipulator import FileManipulator


class SelectCNNTrain(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cnnName = "AlexNet"
        fileMan = FileManipulator("assets/cnn_architecture/"
                                  + self.cnnName + "/details.txt")
        self.cnnDetails = fileMan.readContents()

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.selectCNNLabel = tk.Label(self.parentFrame,
                            text="The CNN architecture to use during training",
                            font=ft.LARGE_FONT)
        self.selectCNNLabel.pack(padx=10, pady=5)

        self.cnnFrame = tk.Frame(self.parentFrame)
        self.cnnFrame.pack(fill="x")

        self.cnnLabel = tk.Label(self.cnnFrame, text="CNN Architecture:",
                                 font=ft.NORMAL_FONT)
        self.cnnLabel.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.cnnEntry = tk.Entry(self.cnnFrame)
        self.cnnEntry.insert(0, self.cnnName)
        self.cnnEntry.config(state="readonly", readonlybackground='white')
        self.cnnEntry.pack(fill="x", side=tk.LEFT, padx=10, expand="True")

        self.cnnDetailLabel = tk.Label(self.parentFrame,
                                       text="CNN Architecture details:",
                                       font=ft.NORMAL_FONT)
        self.cnnDetailLabel.pack(anchor="w", padx=10)

        self.cnnDetailFrame = tk.Frame(self.parentFrame)
        self.cnnDetailFrame.pack()

        self.cnnDetailEntry = tkst.ScrolledText(self.cnnDetailFrame,
                                                   height=10, wrap="word")

        self.cnnDetailEntry.insert(tk.INSERT, self.cnnDetails)

        self.cnnDetailEntry.config(state="disabled")
        self.cnnDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                     command= self.proceedToSetCNNTrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToSetCNNTrain(self):
        self.controller.updateSelectCNNTrain(self.cnnName)
```

A..22 SelectDatasetTrain.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft
from CBoxHelper import CBoxHelper

class SelectDatasetTrain(tk.Frame):
    def __init__(self, parent, controller, datasetChoice, datasetDetails):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.datasetChoice = datasetChoice
        self.datasetDetails = datasetDetails

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x") #padding

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.selectDatasetLabel = tk.Label(self.parentFrame,
                                           text="Select the dataset to use",
                                           font=ft.LARGE_FONT)
        self.selectDatasetLabel.pack(padx=10, pady=5)

        self.datasetFrame = tk.Frame(self.parentFrame)
        self.datasetFrame.pack(fill="x")

        self.cboxHelper = CBoxHelper("Please select a dataset", "assets/datasets")
        self.datasetDict = self.cboxHelper.getValuesDict()
        self.datasetDetailDict = self.cboxHelper.getValuesDetails()

        self.datasetLabel = tk.Label(self.datasetFrame, text="Select dataset:",
                                     font=ft.NORMAL_FONT)
        self.datasetLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.datasetCombo = ttk.Combobox(self.datasetFrame)
        self.datasetCombo['values'] = self.cboxHelper.getValues()
        self.datasetCombo.config(state="readonly")
        self.datasetCombo.current(self.datasetChoice)
        self.datasetCombo.pack(side=tk.LEFT, fill="x", padx=10, expand="True")
        self.datasetCombo.bind("<<ComboboxSelected>>", self.newSelection)

        self.datasetDetailLabel = tk.Label(self.parentFrame, text="Dataset details:",
                                           font=ft.NORMAL_FONT)
        self.datasetDetailLabel.pack(anchor="w", padx=10)

        self.datasetDetailFrame = tk.Frame(self.parentFrame)
        self.datasetDetailFrame.pack()

        self.datasetDetailEntry = tkst.ScrolledText(self.datasetDetailFrame,
                                                    height=10, wrap="word")

        self.datasetDetailEntry.insert(tk.INSERT, self.datasetDetails)

        self.datasetDetailEntry.config(state="disabled")

        self.datasetDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                     command= self.proceedToSelectCNNTrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=20)

    def newSelection(self, event):
        #String : index
        datasetChoice = self.datasetDict.get(self.datasetCombo.get())

        #String : string
        datasetDetails = self.datasetDetailDict.get(self.datasetCombo.get())
        self.controller.createSelectDatasetTrain(datasetChoice, datasetDetails)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToSelectCNNTrain(self):
        baseChoice = self.cboxHelper.getBaseString()
        choice = self.datasetCombo.get()
        if choice != baseChoice:
            self.controller.updateSelectDatasetTrain(choice)
```

A..23 SetCNNTrain.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft
from CBoxHelper import CBoxHelper

class SetCNNTrain(tk.Frame):
    def __init__(self, parent, controller, trainEpoch, batchSize,
                 optChoice, hyperList, hyperEntry, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.trainEpoch = trainEpoch
        self.batchSize = batchSize
        self.optChoice = optChoice
        self.hyperList = hyperList
        self.hyperEntry = hyperEntry
        self.errorMsg = errorMsg
        self.dynamicHyperparams = []

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.inputLabel = tk.Label(self.parentFrame,
                                   text="Please input the necessary values",
                                   font=ft.LARGE_FONT)
        self.inputLabel.pack(padx=10, pady=10)

        self.epochFrame = tk.Frame(self.parentFrame)
        self.epochFrame.pack(fill="x")

        self.epochLabel = tk.Label(self.epochFrame, text="Training epoch:",
                                   font=ft.NORMAL_FONT)
        self.epochLabel.pack(fill="x", side=tk.LEFT, expand="True", pady=10)

        self.epochEntry = tk.Entry(self.epochFrame)
        self.epochEntry.insert(0, self.trainEpoch)
        self.epochEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.epochGuide = tk.Label(self.epochFrame, text="int >= 1",
                                   font=ft.NORMAL_FONT)
        self.epochGuide.pack(fill="x", side=tk.LEFT, expand="True")

        self.batchFrame = tk.Frame(self.parentFrame)
        self.batchFrame.pack(fill="x")

        self.batchLabel = tk.Label(self.batchFrame, text="Batch size:",
                                   font=ft.NORMAL_FONT)
        self.batchLabel.pack(fill="x", side=tk.LEFT, padx=(0, 30),
                             pady=10, expand="True")

        self.batchEntry = tk.Entry(self.batchFrame)
        self.batchEntry.insert(0, self.batchSize)
        self.batchEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.batchGuide = tk.Label(self.batchFrame, text="int >= 1",
                                   font=ft.NORMAL_FONT)
        self.batchGuide.pack(fill="x", side=tk.LEFT, expand="True")

        self.cboxHelper = CBoxHelper("Please select an optimizer",
                                     "assets/optimizers")
        self.optDict = self.cboxHelper.getValuesDict()
        self.optHyperDict = self.cboxHelper.getOptHyperDict()

        self.hyperparamLabel = tk.Label(self.parentFrame,
                                        text="Specify the optimizer and hyperparameters",
                                        font=ft.LARGE_FONT)
        self.hyperparamLabel.pack(padx=10, pady=10)

        self.optFrame = tk.Frame(self.parentFrame)
        self.optFrame.pack()

        self.optLabel = tk.Label(self.optFrame, text="Optimizer:", font=ft.NORMAL_FONT)
        self.optLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)
        self.optCombo = ttk.Combobox(self.optFrame, width=50)
        self.optCombo['values'] = self.cboxHelper.getValues()
        self.optCombo.config(state="readonly")
        self.optCombo.current(self.optChoice)
        self.optCombo.pack(side=tk.LEFT, fill="x", padx=10, expand="True")
        self.optCombo.bind("<<ComboboxSelected>>", self.newSelection)

        self.createHyper(self.hyperList)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                   command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=(10, 15))

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Train CNN",
                                      command= self.proceedToCNNTrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=(10, 15))

        self.errorLabel = tk.Label(self.parentFrame, text="Error messages:",
                                    font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame, height=5, wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)

        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x")

    def newSelection(self, event):
        trainEpoch = self.epochEntry.get()
        batchSize = self.batchEntry.get()
        optChoice = self.optDict.get(self.optCombo.get())
        hyperList = self.optHyperDict.get(self.optCombo.get())
        hyperEntry = {}
        errorMsg = self.errorEntry.get(1.0, tk.END + "-1c")

        self.controller.createSetCNNTrain(trainEpoch, batchSize, optChoice,
                                          hyperList, hyperEntry, errorMsg)

    def createHyper(self, list):
        baseFrame = tk.Frame(self.parentFrame)
        baseFrame.pack()

        for i in range(0, len(list)):
            if(len(list[i]) == 3):
                hyperName = list[i][0]
                hyperDefault = list[i][1]
                hyperGuide = list[i][2]
                paramFrame = tk.Frame(baseFrame)
                paramFrame.pack()

                paramLabel = tk.Label(paramFrame, text=hyperName + ": ",
                                      font=ft.NORMAL_FONT, width=13)
                paramLabel.pack(fill="x", side=tk.LEFT, expand="True",
                                padx=10, pady=10)

                paramEntry = tk.Entry(paramFrame)
                self.dynamicHyperparams.append(paramEntry)

                if(self.hyperEntry.get(hyperName)):
                    paramEntry.insert(0, self.hyperEntry[hyperName])
                else:
                    paramEntry.insert(0, hyperDefault)

                paramEntry.pack(fill="x", side=tk.LEFT, expand="True")

                paramGuide = tk.Label(paramFrame, text=hyperGuide,
                                      font=ft.NORMAL_FONT, width=30)
                paramGuide.pack(fill="x", side=tk.LEFT, expand="True",
                                padx=10, pady=10)

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToCNNTrain(self):
        baseChoice = self.cboxHelper.getBaseString()
        optChoice = self.optCombo.get()
        if optChoice != baseChoice:
            trainEpoch = self.epochEntry.get().strip()
            batchSize = self.batchEntry.get().strip()
            optChoice = self.optDict.get(self.optCombo.get())
            optStr = self.optCombo.get()
            hyperList = self.optHyperDict.get(self.optCombo.get())
            hyperEntry = {}
            for i in range(0, len(self.hyperList)):
                hyperEntry[self.hyperList[i][0]] = self.dynamicHyperparams[i].get().strip()

            self.controller.updateSetCNNTrain(trainEpoch, batchSize, optChoice,
                                              optStr, hyperList, hyperEntry)
```

A..24 SetParamSearch.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

class SetParamSearch(tk.Frame):
    def __init__(self, parent, controller, cStart, cEnd, cStep,
                 gStart, gEnd, gStep, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cStart = cStart
        self.cEnd = cEnd
        self.cStep = cStep
        self.gStart = gStart
        self.gEnd = gEnd
        self.gStep = gStep
        self.errorMsg = errorMsg

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.hyperparamLabel = tk.Label(self.parentFrame,
                                        text="Set the ranges for Cost and Gamma",
                                        font=ft.LARGE_FONT)
        self.hyperparamLabel.pack(padx=10, pady=10)

        self.posStepLabel = tk.Label(self.parentFrame,
            text="If start < end, range = 2^{start,...,start+k*step,...,end}, k=1,2,3,...",
            font=ft.NORMAL_FONT)
        self.posStepLabel.pack(padx=10, pady=5)

        self.negStepLabel = tk.Label(self.parentFrame,
            text="If start > end, range = 2^{start,...,start-k*step,...,end}, k=1,2,3,...",
            font=ft.NORMAL_FONT)
        self.negStepLabel.pack(padx=10, pady=5)

        self.rangeCostLabel = tk.Label(self.parentFrame,
                                       text="Set the range for the cost",
                                       font=ft.NORMAL_FONT)
        self.rangeCostLabel.pack(padx=10, pady=10)

        self.cStartFrame = tk.Frame(self.parentFrame)
        self.cStartFrame.pack()

        self.cStartLabel = tk.Label(self.cStartFrame, text="Cost Start:",
                                    font=ft.NORMAL_FONT)
        self.cStartLabel.pack(fill="x", side=tk.LEFT, padx=(0, 25), pady=10)

        self.cStartEntry = tk.Entry(self.cStartFrame)
        self.cStartEntry.insert(0, self.cStart)
        self.cStartEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.cStartGuide = tk.Label(self.cStartFrame, text="float",
                                    font=ft.NORMAL_FONT)
        self.cStartGuide.pack(fill="x", side=tk.LEFT, padx=(25,37), pady=10)

        self.cEndFrame = tk.Frame(self.parentFrame)
        self.cEndFrame.pack()

        self.cEndLabel = tk.Label(self.cEndFrame, text="Cost End:",
                                  font=ft.NORMAL_FONT)
        self.cEndLabel.pack(fill="x", side=tk.LEFT, padx=(0, 25), pady=10)

        self.cEndEntry = tk.Entry(self.cEndFrame)
        self.cEndEntry.insert(0, self.cEnd)
        self.cEndEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.cEndGuide = tk.Label(self.cEndFrame, text="float",
                                  font=ft.NORMAL_FONT)
        self.cEndGuide.pack(fill="x", side=tk.LEFT, padx=(25,30), pady=10)

        self.cStepFrame = tk.Frame(self.parentFrame)
        self.cStepFrame.pack()

        self.cStepLabel = tk.Label(self.cStepFrame, text="Cost Step:",
                                   font=ft.NORMAL_FONT)
        self.cStepLabel.pack(fill="x", side=tk.LEFT, padx=(23, 25), pady=10)

        self.cStepEntry = tk.Entry(self.cStepFrame)
        self.cStepEntry.insert(0, self.cStep)
        self.cStepEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.cStepGuide = tk.Label(self.cStepFrame, text="positive float",
                                   font=ft.NORMAL_FONT)
        self.cStepGuide.pack(fill="x", side=tk.LEFT, padx=(25,0), pady=10)
        ####

        self.gammaCostLabel = tk.Label(self.parentFrame,
                                       text="Set the range for the gamma",
                                       font=ft.NORMAL_FONT)
        self.gammaCostLabel.pack(padx=10, pady=10)

        self.gStartFrame = tk.Frame(self.parentFrame)
        self.gStartFrame.pack()
        self.gStartLabel = tk.Label(self.gStartFrame,
                                    text="Gamma Start:",
                                    font=ft.NORMAL_FONT)
        self.gStartLabel.pack(fill="x", side=tk.LEFT, padx=(0, 25), pady=10)

        self.gStartEntry = tk.Entry(self.gStartFrame)
        self.gStartEntry.insert(0, self.gStart)
        self.gStartEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.gStartGuide = tk.Label(self.gStartFrame,
                                    text="float",
                                    font=ft.NORMAL_FONT)
        self.gStartGuide.pack(fill="x", side=tk.LEFT, padx=(25,65), pady=10)

        #
        self.gEndFrame = tk.Frame(self.parentFrame)
        self.gEndFrame.pack()

        self.gEndLabel = tk.Label(self.gEndFrame,
                                  text="Gamma End:",
                                  font=ft.NORMAL_FONT)
        self.gEndLabel.pack(fill="x", side=tk.LEFT, padx=(0, 25), pady=10)

        self.gEndEntry = tk.Entry(self.gEndFrame)
        self.gEndEntry.insert(0, self.gEnd)
        self.gEndEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.gEndGuide = tk.Label(self.gEndFrame,
                                  text="float",
                                  font=ft.NORMAL_FONT)
        self.gEndGuide.pack(fill="x", side=tk.LEFT, padx=(25,55), pady=10)


        self.gStepFrame = tk.Frame(self.parentFrame)
        self.gStepFrame.pack()

        self.gStepLabel = tk.Label(self.gStepFrame,
                                   text="Gamma Step:",
                                   font=ft.NORMAL_FONT)
        self.gStepLabel.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.gStepEntry = tk.Entry(self.gStepFrame)
        self.gStepEntry.insert(0, self.gStep)
        self.gStepEntry.pack(fill="x", side=tk.LEFT, expand="True")

        self.gStepGuide = tk.Label(self.gStepFrame,
                                   text="positive float",
                                   font=ft.NORMAL_FONT)
        self.gStepGuide.pack(fill="x", side=tk.LEFT, padx=(25,10), pady=10)

        self.defaultFrame = tk.Frame(self.parentFrame)
        self.defaultFrame.pack()

        self.defaultBtn = ttk.Button(self.defaultFrame,
                                     text="Use Default Values",
                                     command= self.useDefault )
        self.defaultBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame,
                                  text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.createBtn = ttk.Button(self.buttonFrame,
                                    text="Start Search",
                                    command= self.startParamSearch )
        self.createBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.errorLabel = tk.Label(self.parentFrame,
                                   text="Error messages:",
                                   font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame,
                                            height=5, width=70,
                                            wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)

        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x", padx=(10,0) )

    def backToHome(self):
        self.controller.createDeleteWarning()

    def useDefault(self):
        self.controller.createSetParamSearch("-5", "15", "2", "3", "-15", "2", "")

    def startParamSearch(self):
        cStart = self.cStartEntry.get().strip()
        cEnd = self.cEndEntry.get().strip()
        cStep = self.cStepEntry.get().strip()
        gStart = self.gStartEntry.get().strip()
        gEnd = self.gEndEntry.get().strip()
        gStep = self.gStepEntry.get().strip()
        self.controller.updateSetParamSearch(cStart, cEnd, cStep, gStart, gEnd, gStep)
```

A..25 SetSVMTrain.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from HybridModelTrainer import HybridModelTrainer
from FileManipulator import FileManipulator

class SetSVMTrain(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        fileMan = FileManipulator("assets/models/"
                                  + self.controller.getModelToCreate()
                                  + "/bestSVMParams.txt")

        params = fileMan.readContents().split("\n")

        svmParams = params[0].split(",")

        self.svmType = "C-support vector classification"
        self.kernelType = "Radial basis function"

        self.cost = svmParams[1]
        self.gamma = svmParams[3]
        self.rate = svmParams[5]
        self.degree = "3"
        self.coef = "0"
        self.epsilon = "0.001"
        self.shrink = "1"
        self.weight = "1"


        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.hyperparamLabel = tk.Label(self.parentFrame,
                        text="The hyperparameters to use for training the SVM",
                        font=ft.LARGE_FONT)
        self.hyperparamLabel.pack(padx=10, pady=10)

        self.svmTypeFrame = tk.Frame(self.parentFrame)
        self.svmTypeFrame.pack(padx=10)

        self.svmTypeLabel = tk.Label(self.svmTypeFrame,
                                     text="SVM Type:", font=ft.NORMAL_FONT)
        self.svmTypeLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.svmTypeEntry = tk.Label(self.svmTypeFrame,
                                     text=self.svmType, font=ft.NORMAL_FONT)
        self.svmTypeEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=10)

        self.kernelTypeFrame = tk.Frame(self.parentFrame)
        self.kernelTypeFrame.pack(padx=10)

        self.kernelTypeLabel = tk.Label(self.kernelTypeFrame,
                                        text="Kernel Type:",
                                        font=ft.NORMAL_FONT)
        self.kernelTypeLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.kernelTypeEntry = tk.Label(self.kernelTypeFrame,
                                        text=self.kernelType,
                                        font=ft.NORMAL_FONT)
        self.kernelTypeEntry.pack(fill="x", side=tk.LEFT, expand="True",padx=10)

        self.gridLabel = tk.Label(self.parentFrame,
                text="The following values were computed by the grid.py tool",
                font=ft.NORMAL_FONT)
        self.gridLabel.pack(padx=10, pady=10)

        self.costFrame = tk.Frame(self.parentFrame)
        self.costFrame.pack(padx=10)

        self.costLabel = tk.Label(self.costFrame, text="Cost:",
                                  font=ft.NORMAL_FONT)
        self.costLabel.pack(fill="x", side=tk.LEFT, padx=(15,10), pady=10)

        self.costEntry = tk.Label(self.costFrame, text=self.cost,
                                  font=ft.NORMAL_FONT)
        self.costEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=10)

        self.gammaFrame = tk.Frame(self.parentFrame)
        self.gammaFrame.pack(padx=10)

        self.gammaLabel = tk.Label(self.gammaFrame, text="Gamma:",
                                  font=ft.NORMAL_FONT)
        self.gammaLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.gammaEntry = tk.Label(self.gammaFrame, text=self.gamma,
                                  font=ft.NORMAL_FONT)
        self.gammaEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=10)

        self.rateFrame = tk.Frame(self.parentFrame)
        self.rateFrame.pack(padx=10)

        self.rateLabel = tk.Label(self.rateFrame, text="CV Rate:",
                                 font=ft.NORMAL_FONT)
        self.rateLabel.pack(fill="x", side=tk.LEFT, padx=(15,10), pady=10)

        self.rateEntry = tk.Label(self.rateFrame, text=self.rate,
                                 font=ft.NORMAL_FONT)
        self.rateEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=10)

        self.defLabel = tk.Label(self.parentFrame,
            text="The following hyperparameters will use their default values",
            font=ft.NORMAL_FONT)
        self.defLabel.pack(padx=10, pady=10)

        self.degFrame = tk.Frame(self.parentFrame)
        self.degFrame.pack(padx=10)

        self.degLabel = tk.Label(self.degFrame, text="Degree:",
                                font=ft.NORMAL_FONT)
        self.degLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.degEntry = tk.Label(self.degFrame, text=self.degree,
                                font=ft.NORMAL_FONT)
        self.degEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=(10,40))

        self.coefFrame = tk.Frame(self.parentFrame)
        self.coefFrame.pack(padx=10)

        self.coefLabel = tk.Label(self.coefFrame, text="Coef 0:",
                                 font=ft.NORMAL_FONT)
        self.coefLabel.pack(fill="x", side=tk.LEFT, padx=(15,10), pady=10)

        self.coefEntry = tk.Label(self.coefFrame, text=self.coef,
                                 font=ft.NORMAL_FONT)
        self.coefEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=(10,40))

        self.epsFrame = tk.Frame(self.parentFrame)
        self.epsFrame.pack(padx=10)

        self.epsLabel = tk.Label(self.epsFrame, text="Epsilon:",
                                font=ft.NORMAL_FONT)
        self.epsLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.epsEntry = tk.Label(self.epsFrame, text=self.epsilon,
                                font=ft.NORMAL_FONT)
        self.epsEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=10)

        self.shrinkFrame = tk.Frame(self.parentFrame)
        self.shrinkFrame.pack(padx=10)

        self.shrinkLabel = tk.Label(self.shrinkFrame, text="Shrink:",
                                   font=ft.NORMAL_FONT)
        self.shrinkLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.shrinkEntry = tk.Label(self.shrinkFrame, text=self.shrink,
                                   font=ft.NORMAL_FONT)
        self.shrinkEntry.pack(fill="x", side=tk.LEFT, expand="True",
                             padx=(10,35))

        self.weightFrame = tk.Frame(self.parentFrame)
        self.weightFrame.pack(padx=10)

        self.weightLabel = tk.Label(self.weightFrame, text="Weight:",
                                   font=ft.NORMAL_FONT)
        self.weightLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.weightEntry = tk.Label(self.weightFrame, text=self.weight,
                                   font=ft.NORMAL_FONT)
        self.weightEntry.pack(fill="x", side=tk.LEFT, expand="True", padx=(10,40))

        self.noteLabel = tk.Label(self.parentFrame,
            text="Note: Procedure is based from 'A Practical Guide to Support Vector Classification'",
            font=ft.NORMAL_FONT)
        self.noteLabel.pack(padx=10, pady=10)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                 command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=(10, 15))

        self.proceedBtn = ttk.Button(self.buttonFrame, text="Train SVM",
                                    command= self.proceedToSVMTrain )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=(10, 15))

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToSVMTrain(self):
        cost = float(self.cost)
        gamma = float(self.gamma)
        strToWrite = "\n=== SVM Details ===\n"
        strToWrite += "SVM Type: " + self.svmType + "\n"
        strToWrite += "Kernel Type: " + self.kernelType + "\n"
        strToWrite += "Cost: " + self.cost + "\n"
        strToWrite += "Gamma: " + self.gamma + "\n"
        strToWrite += "CV Rate: " + self.rate + "\n"
        strToWrite += "Degree: " + self.degree + "\n"
        strToWrite += "Coef 0: " + self.coef + "\n"
        strToWrite += "Epsilon: " + self.epsilon + "\n"
        strToWrite += "Shrink: " + self.shrink + "\n"
        strToWrite += "Weight: " + self.weight + "\n"

        modelName = self.controller.getModelToCreate()
        trainer = HybridModelTrainer(modelName)
        trainer.startTrainSVM(cost, gamma, strToWrite)
        self.controller.createSVMTrain()
```

A..26 SignUp.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from FileManipulator import FileManipulator

class SignUp(tk.Frame):
    def __init__(self, parent, controller, username, password,
                 confPass, agreedToTerms, errorMsg):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.username = username
        self.password = password
        self.confPass = confPass
        self.agreedToTerms = agreedToTerms
        self.errorMsg = errorMsg

        fileMan = FileManipulator("assets/terms_conds.txt")
        self.terms_conds = fileMan.readContents()

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.logInLabel = tk.Label(self.parentFrame, text="Create an Account",
                                   font=ft.TITLE_FONT)
        self.logInLabel.pack(padx=10, pady=20)

        self.unameFrame = tk.Frame(self.parentFrame)
        self.unameFrame.pack(fill="x")

        self.unameLabel = tk.Label(self.unameFrame, text="Username:",
                                   font=ft.NORMAL_FONT)
        self.unameLabel.pack(fill="x", side=tk.LEFT, padx=(10, 66), pady=15)

        self.unameEntry = tk.Entry(self.unameFrame)
        self.unameEntry.insert(0, self.username)
        self.unameEntry.pack(fill="x", expand="True")

        self.passFrame = tk.Frame(self.parentFrame)
        self.passFrame.pack(fill="x")

        self.passLabel = tk.Label(self.passFrame, text="Password:",
                                  font=ft.NORMAL_FONT)
        self.passLabel.pack(fill="x", side=tk.LEFT, padx=(10, 70), pady=15)

        self.passEntry = tk.Entry(self.passFrame)
        self.passEntry.insert(0, self.password)
        self.passEntry.config(show="*")
        self.passEntry.pack(fill="x", expand="True")

        self.pass2Frame = tk.Frame(self.parentFrame)
        self.pass2Frame.pack(fill="x")

        self.pass2Label = tk.Label(self.pass2Frame, text="Confirm Password:",
                                   font=ft.NORMAL_FONT)
        self.pass2Label.pack(fill="x", side=tk.LEFT, padx=10, pady=15)

        self.pass2Entry = tk.Entry(self.pass2Frame)
        self.pass2Entry.insert(0, self.confPass)
        self.pass2Entry.config(show="*")
        self.pass2Entry.pack(fill="x", expand="True")

        self.termsLabel = tk.Label(self.parentFrame,
                                   text="Terms and Conditions:",
                                   font=ft.NORMAL_FONT)
        self.termsLabel.pack(anchor="w", padx=10)
        self.termsFrame = tk.Frame(self.parentFrame)
        self.termsFrame.pack()

        self.termsEntry = tkst.ScrolledText(self.termsFrame,
                                            height=8, wrap="word")
        self.termsEntry.insert(tk.INSERT, self.terms_conds)

        self.termsEntry.config(state="disabled")
        self.termsEntry.pack(fill="x", padx=(10,0) )

        self.termsVar = tk.IntVar()

        self.termsCheckBtn = tk.Checkbutton(self.parentFrame,
                text="I have read and understood the terms and conditions.",
                variable=self.termsVar, font=('Verdana', 10, "bold"))
        if self.agreedToTerms == 1:
            self.termsCheckBtn.select()

        self.termsCheckBtn.pack(fill="x", pady=10)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back",
                                 command = self.backToLogIn )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=(10, 15))

        self.createAccBtn = ttk.Button(self.buttonFrame, text="Create Account",
                                      command= self.createAccount)
        self.createAccBtn.pack(side=tk.LEFT, ipady=5,  pady=(10, 15))

        self.errorLabel = tk.Label(self.parentFrame, text="Error messages:",
                                  font=ft.NORMAL_FONT)
        self.errorLabel.pack(anchor="w", padx=10)

        self.errorFrame = tk.Frame(self.parentFrame)
        self.errorFrame.pack()

        self.errorEntry = tkst.ScrolledText(self.errorFrame,
                                            height=5, wrap="word")

        self.errorEntry.insert(tk.INSERT, self.errorMsg)

        self.errorEntry.config(state="disabled")
        self.errorEntry.pack(fill="x", padx=(10,0) )

    def backToLogIn(self):
        self.controller.createLogIn("", "", "")

    def createAccount(self):
        username = self.unameEntry.get()
        password = self.passEntry.get()
        confPass = self.pass2Entry.get()
        agreedToTerms = self.termsVar.get()
        self.controller.createAccount(username.strip(), password,
                                      confPass, agreedToTerms)
```

A..27 SuccessCreateAccount.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class SuccessCreateAccount(tk.Frame):
    def __init__(self, parent, controller, username):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.username = username

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                              bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                              bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.successLabel = tk.Label(self.parentFrame,
                                    text="Account successfully created!",
                                    font=ft.TITLE_FONT)
        self.successLabel.pack(padx=10, pady=20)

        self.unameLabel = tk.Label(self.parentFrame,
                                  text="Username: " + self.username,
                                  font=ft.TITLE_FONT)
        self.unameLabel.pack(padx=10, pady=20)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.proceedBtn = ttk.Button(self.buttonFrame,
                                    text="Back to Log in",
                                    command = self.backToLogIn )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)
    def backToLogIn(self):
        self.controller.createLogIn("", "", "")
```

A..28 SuccessDataset.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class SuccessDataset(tk.Frame):

    def __init__(self, parent, controller, nameStr,
                 trainingInt, validationInt, testInt):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.nameStr = nameStr
        self.trainingInt = trainingInt
        self.validationInt = validationInt
        self.testInt = testInt

        self.label = tk.Label(self, text="Create Dataset",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.partitionLabel = tk.Label(self.parentFrame,
                                       text="Dataset successfully created!",
                                       font=ft.TITLE_FONT)
        self.partitionLabel.pack(padx=10, pady=15)

        self.nameFrame = tk.Frame(self.parentFrame)
        self.nameFrame.pack(anchor="e", fill="x")

        self.nameLabel = tk.Label(self.nameFrame, text="Name of dataset:",
                                  font=ft.LARGE_FONT)
        self.nameLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=20)

        self.nameAnswer = tk.Label(self.nameFrame, text=self.nameStr,
                                   font=ft.LARGE_FONT)
        self.nameAnswer.pack(fill="x", side=tk.LEFT, padx=10)

        self.trainingFrame = tk.Frame(self.parentFrame)
        self.trainingFrame.pack(fill="x")

        self.trainingLabel = tk.Label(self.trainingFrame,
                                      text="Number of data in training set:",
                                      font=ft.LARGE_FONT)
        self.trainingLabel.pack(fill="x", side=tk.LEFT, padx=(10, 30), pady=10)

        self.trainingAnswer = tk.Label(self.trainingFrame,
                                       text=self.trainingInt,
                                       font=ft.LARGE_FONT)
        self.trainingAnswer.pack(fill="x", side=tk.LEFT, padx=10)

        self.validationFrame = tk.Frame(self.parentFrame)
        self.validationFrame.pack(fill="x")

        self.validationLabel = tk.Label(self.validationFrame,
                                        text="Number of data in validation set:",
                                        font=ft.LARGE_FONT)
        self.validationLabel.pack(fill="x", side=tk.LEFT, padx=10, pady=10)

        self.validationAnswer = tk.Label(self.validationFrame,
                                         text=self.validationInt,
                                         font=ft.LARGE_FONT)
        self.validationAnswer.pack(fill="x", side=tk.LEFT, padx=10)

        self.testFrame = tk.Frame(self.parentFrame)
        self.testFrame.pack(fill="x")

        self.testLabel = tk.Label(self.testFrame,
                                  text="Number of data in test set:",
                                  font=ft.LARGE_FONT)
        self.testLabel.pack(fill="x", side=tk.LEFT, padx=(10, 75), pady=10)

        self.testAnswer = tk.Label(self.testFrame,
                                   text=self.testInt,
                                   font=ft.LARGE_FONT)
        self.testAnswer.pack(fill="x", side=tk.LEFT, padx=10)

        self.homeBtn = ttk.Button(self.parentFrame,
                                  text="Back to Home",
                                  command=self.backToHome )
        self.homeBtn.pack(ipady=5, pady=10)

    def backToHome(self):
        self.controller.createHomePage()
```

A..29 SuccessEvaluation.py

```
import tkinter as tk
from tkinter import ttk #css
```

```
import tkinter.scrolledtext as tkst
import Font as ft

from FileManipulator import FileManipulator

class SuccessEvaluation(tk.Frame):

    def __init__(self, parent, controller, cnnChoice, datasetChoice):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cnnChoice = cnnChoice
        self.datasetChoice = datasetChoice
        self.filePath = ("assets/predictions/" + self.cnnChoice
                        + "-" + self.datasetChoice + ".txt")

        fileMan = FileManipulator(self.filePath)
        self.content = fileMan.readContents()

        self.label1 = tk.Label(self, text="Evaluate the Performance",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="of a Hybrid CNN-SVM Model",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.partitionLabel = tk.Label(self.parentFrame,
                                       text="Performance Evaluated!",
                                       font=ft.TITLE_FONT)
        self.partitionLabel.pack(padx=10, pady=15)

        self.nameCNNFrame = tk.Frame(self.parentFrame)
        self.nameCNNFrame.pack(anchor="e", fill="x")

        self.nameCNNLabel = tk.Label(self.nameCNNFrame,
                                     text="Hybrid CNN-SVM model name:",
                                     font=ft.LARGE_FONT)
        self.nameCNNLabel.pack(fill="x", side=tk.LEFT, padx=5, pady=10)

        self.nameCNNAnswer = tk.Label(self.nameCNNFrame,
                                      text=self.cnnChoice,
                                      font=ft.LARGE_FONT)
        self.nameCNNAnswer.pack(fill="x", side=tk.LEFT, padx=5)

        self.nameDatasetFrame = tk.Frame(self.parentFrame)
        self.nameDatasetFrame.pack(anchor="e", fill="x")

        self.nameDatasetLabel = tk.Label(self.nameDatasetFrame,
                                         text="Dataset name:",
                                         font=ft.LARGE_FONT)
        self.nameDatasetLabel.pack(fill="x", side=tk.LEFT, padx=5, pady=10)

        self.nameDatasetAnswer = tk.Label(self.nameDatasetFrame,
                                          text=self.datasetChoice,
                                          font=ft.LARGE_FONT)
        self.nameDatasetAnswer.pack(fill="x", side=tk.LEFT, padx=5)

        self.logLabel = tk.Label(self.parentFrame,
                                 text="Predictions:",
                                 font=ft.LARGE_FONT)
        self.logLabel.pack(padx=10, pady=5)

        self.logFrame = tk.Frame(self.parentFrame)
        self.logFrame.pack()

        self.xScrollbar = tk.Scrollbar(self.logFrame, orient=tk.HORIZONTAL)
        self.xScrollbar.pack(side="bottom", fill="x")

        self.logEntry = tkst.ScrolledText(self.logFrame,
                                          height=10, wrap="none",
                                          xscrollcommand=self.xScrollbar.set)

        self.logEntry.insert(tk.INSERT, self.content)

        self.xScrollbar.config(command=self.logEntry.xview)
        self.logEntry.config(state="disabled")

        self.logEntry.pack(fill="x")

        self.filePathFrame = tk.Frame(self.parentFrame)
        self.filePathFrame.pack(anchor="e", fill="x")

        self.filePathLabel = tk.Label(self.filePathFrame,
                                      text="Filepath:", font=ft.NORMAL_FONT)
        self.filePathLabel.pack(fill="x", side=tk.LEFT, padx=5, pady=10)

        self.filePathAnswer = tk.Label(self.filePathFrame,
                                       text=self.filePath, font=ft.NORMAL_FONT)
        self.filePathAnswer.pack(fill="x", side=tk.LEFT, padx=5)

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

    def backToHome(self):
        self.controller.createHomePage()

    def proceedToUpdateModel(self):
        self.controller.createUpdateModel(0, "")
```

A..30 SuccessTraining.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from FileManipulator import FileManipulator

class SuccessTraining(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        modelName = self.controller.getModelToCreate()
        self.trainingDetails = ""

        fileMan = FileManipulator("assets/models/" + modelName + "/details.txt")
        self.trainingDetails += fileMan.readContents()
        fileMan = FileManipulator("assets/models/" + modelName + "/cnn.txt")
        self.trainingDetails += fileMan.readContents()
        fileMan = FileManipulator("assets/models/" + modelName + "/pca.txt")
        self.trainingDetails += fileMan.readContents()
        fileMan = FileManipulator("assets/models/" + modelName + "/svm.txt")
        self.trainingDetails += fileMan.readContents()

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.successLabel1 = tk.Label(self.parentFrame,
                                      text="Hybrid CNN-SVM model",
                                      font=ft.TITLE_FONT)
        self.successLabel1.pack(padx=10, pady=(15,5))

        self.successLabel2 = tk.Label(self.parentFrame,
                                      text="trained successfully!",
                                      font=ft.TITLE_FONT)
        self.successLabel2.pack(padx=10, pady=(5,15))

        self.cnnDetailLabel = tk.Label(self.parentFrame,
                                       text="Training details:",
                                       font=ft.NORMAL_FONT)
        self.cnnDetailLabel.pack(anchor="w", padx=10)

        self.cnnDetailFrame = tk.Frame(self.parentFrame)
        self.cnnDetailFrame.pack()

        self.cnnDetailEntry = tkst.ScrolledText(self.cnnDetailFrame,
                                                height=15, wrap="word")

        self.cnnDetailEntry.insert(tk.INSERT, self.trainingDetails)
        self.cnnDetailEntry.config(state="disabled")
        self.cnnDetailEntry.pack(fill="x", pady=(0, 20))

        self.homeBtn = ttk.Button(self.parentFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(ipady=5, pady=10)

    def backToHome(self):
        self.controller.modelToCreate = None
        self.controller.createHomePage()
```

A..31 SuccessUpdateModel.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class SuccessUpdateModel(tk.Frame):
    def __init__(self, parent, controller, cnnChoice, userAIExpert):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cnnChoice = cnnChoice
        self.userAIExpert = userAIExpert

        self.label1 = tk.Label(self, text="Update the Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Model Used by the System",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")
        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.successLabel = tk.Label(self.parentFrame,
                                     text="Updated successfully!",
                                     font=ft.TITLE_FONT)
        self.successLabel.pack(padx=10, pady=(5,15))

        self.nameCNNFrame = tk.Frame(self.parentFrame)
        self.nameCNNFrame.pack(anchor="e", fill="x")

        self.nameCNNLabel = tk.Label(self.nameCNNFrame,
                                     text="Currently used model:",
                                     font=ft.LARGE_FONT)
        self.nameCNNLabel.pack(fill="x", side=tk.LEFT, padx=5, pady=10)

        self.nameCNNAnswer = tk.Label(self.nameCNNFrame,
                                      text=self.cnnChoice,
                                      font=ft.LARGE_FONT)
        self.nameCNNAnswer.pack(fill="x", side=tk.LEFT, padx=5)

        self.nameAIExpertFrame = tk.Frame(self.parentFrame)
        self.nameAIExpertFrame.pack(anchor="e", fill="x")

        self.nameLabel = tk.Label(self.nameAIExpertFrame,
                                  text="Updated by:",
                                  font=ft.LARGE_FONT)
        self.nameLabel.pack(fill="x", side=tk.LEFT, padx=5, pady=10)

        self.nameAnswer = tk.Label(self.nameAIExpertFrame,
                                   text=self.userAIExpert,
                                   font=ft.LARGE_FONT)
        self.nameAnswer.pack(fill="x", side=tk.LEFT, padx=5)

        self.homeBtn = ttk.Button(self.parentFrame,
                                  text="Back to Home",
                                  command=self.backToHome )
        self.homeBtn.pack(ipady=5, pady=10)

    def backToHome(self):
        self.controller.createHomePage()
```

A..32 SULogIn.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft

class SULogIn(tk.Frame):

    def __init__(self, parent, controller, password, attempt):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.password = password
        self.attempt = attempt

        self.label1 = tk.Label(self, text="Predicting Website's Visual Appeal",
                               bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Using Hybrid CNN-SVM",
                               bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.logInLabel = tk.Label(self.parentFrame,
                                   text="Please enter the superuser password",
                                   font=ft.LARGE_FONT)
        self.logInLabel.pack(padx=10, pady=20)

        self.passFrame = tk.Frame(self.parentFrame)
        self.passFrame.pack(fill="x")

        self.passLabel = tk.Label(self.passFrame, text="Password:",
                                  font=ft.NORMAL_FONT)
        self.passLabel.pack(fill="x", side=tk.LEFT, padx=(10, 14), pady=15)

        self.passEntry = tk.Entry(self.passFrame)
        self.passEntry.config(show="*")
        self.passEntry.pack(fill="x", padx=10, expand="True")

        self.attemptsFrame = tk.Frame(self.parentFrame)
        self.attemptsFrame.pack(fill="x")

        self.attemptsEntry = tk.Label(self.attemptsFrame, text=self.attempt,
                                      font=ft.NORMAL_FONT)
        self.attemptsEntry.pack(fill="x", padx=(12, 8), pady=10 )

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.backBtn = ttk.Button(self.buttonFrame, text="Back",
                                  command= self.backToWelcome )
        self.backBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)
        self.logInBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                    command= self.proceedToSignUp )
        self.logInBtn.pack(side=tk.LEFT, ipady=5,  pady=10)

        self.note1Frame = tk.Frame(self.parentFrame)
        self.note1Frame.pack(fill="x")

        self.note1Label = tk.Label(self.note1Frame,
                                    text="Don't know the superuser password?",
                                    font=ft.NORMAL_FONT)
        self.note1Label.pack(padx=10, pady=(10,5))

        self.note2Frame = tk.Frame(self.parentFrame)
        self.note2Frame.pack(fill="x")

        self.note2Label = tk.Label(self.note2Frame,
                    text="Please send me an e-mail here: rtmaristela@up.edu.ph",
                    font=ft.NORMAL_FONT)
        self.note2Label.pack(padx=10, pady=5)

        self.note3Frame = tk.Frame(self.parentFrame)
        self.note3Frame.pack(fill="x")

        self.note3Label = tk.Label(self.note3Frame,
                    text="The subject of the e-mail should only be: [SU Pass]",
                    font=ft.NORMAL_FONT)
        self.note3Label.pack(padx=10, pady=5)

    def backToWelcome(self):
        self.controller.createWelcomePage()

    def proceedToSignUp(self):
        self.controller.updateSULogIn(self.passEntry.get())
```

A..33 SVMTrain.py

```
import tkinter as tk
from tkinter import ttk #css
import Font as ft
import tkinter.scrolledtext as tkst

from subprocess import *

from FileManipulator import FileManipulator

class SVMTrain(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        fileMan = FileManipulator("assets/app_guide.txt")
        self.appDetails = fileMan.readContents()

        self.label = tk.Label(self, text="Create a Hybrid CNN-SVM Model",
                              bg="white", font=ft.TITLE_FONT)
        self.label.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.plotSVMLabel1 = tk.Label(self.parentFrame,
                                    text="Click the \"Open App\" to view a 2D",
                                    font=ft.LARGE_FONT)
        self.plotSVMLabel1.pack(padx=10, pady=0)

        self.plotSVMLabel2 = tk.Label(self.parentFrame,
                                text="plot of data with separating hyperplane",
                                font=ft.LARGE_FONT)
        self.plotSVMLabel2.pack(padx=10, pady=0)

        self.openFrame = tk.Frame(self.parentFrame)
        self.openFrame.pack()

        self.openBtn = ttk.Button(self.openFrame, text="Open App",
                                  command= self.openSVMToy )
        self.openBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.appGuideLabel = tk.Label(self.parentFrame, text="App guide:",
                                      font=ft.NORMAL_FONT)
        self.appGuideLabel.pack(anchor="w", padx=10)

        self.appDetailFrame = tk.Frame(self.parentFrame)
        self.appDetailFrame.pack()

        self.appDetailEntry = tkst.ScrolledText(self.appDetailFrame,
                                                height=15, wrap="word")

        self.appDetailEntry.insert(tk.INSERT, self.appDetails)
        self.appDetailEntry.config(state="disabled")
        self.appDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)
        self.proceedBtn = ttk.Button(self.buttonFrame, text="Proceed",
                                    command= self.proceedToSuccessTraining )
        self.proceedBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

    def openSVMToy(self):
        cmd = '"{0}"'.format("assets/gpu_libsvm/svm-toy.exe")
        f = Popen(cmd, shell = True, stdout = PIPE).stdout

    def backToHome(self):
        self.controller.createDeleteWarning()

    def proceedToSuccessTraining(self):
        self.controller.createSuccessTraining()
```

A..34 UpdateModel.py

```
import tkinter as tk
from tkinter import ttk #css
import tkinter.scrolledtext as tkst
import Font as ft

from CBoxHelper import CBoxHelper
from FileManipulator import FileManipulator

class UpdateModel(tk.Frame):

    def __init__(self, parent, controller, cnnChoice, cnnDetails):
        tk.Frame.__init__(self, parent)

        self.controller = controller
        self.cnnChoice = cnnChoice
        self.cnnDetails = cnnDetails

        self.cboxHelper = CBoxHelper("Please select a model", "assets/models")

        self.cnnDict = self.cboxHelper.getValuesDict()
        self.cnnDetailDict = self.cboxHelper.getModelsDetails()

        self.bestModelName, self.bestModelDetails = self.controller.getCurrentModel()

        self.label1 = tk.Label(self, text="Update the Hybrid CNN-SVM",
                              bg="white", font=ft.TITLE_FONT)
        self.label1.pack(ipady=5, fill="x")

        self.label2 = tk.Label(self, text="Model Used by the System",
                              bg="white", font=ft.TITLE_FONT)
        self.label2.pack(ipady=5, fill="x")

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.selectCNNLabel = tk.Label(self.parentFrame,
                text="Select the hybrid CNN-SVM model to be used by the system",
                font=ft.LARGE_FONT)
        self.selectCNNLabel.pack(padx=10, pady=5)

        self.cnnFrame = tk.Frame(self.parentFrame)
        self.cnnFrame.pack(fill="x")

        self.cnnLabel = tk.Label(self.cnnFrame, text="Model:", font=ft.NORMAL_FONT)
        self.cnnLabel.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.cnnCombo = ttk.Combobox(self.cnnFrame)
        self.cnnCombo['values'] = self.cboxHelper.getValues()
        self.cnnCombo.config(state="readonly")
        self.cnnCombo.current(self.cnnChoice)
        self.cnnCombo.pack(fill="x", side=tk.LEFT, padx=10, expand="True")
        self.cnnCombo.bind("<<ComboboxSelected>>", self.modelNewSelection)

        self.cnnDetailLabel = tk.Label(self.parentFrame, text="Model details:",
                                      font=ft.NORMAL_FONT)
        self.cnnDetailLabel.pack(anchor="w", padx=10)

        self.cnnDetailFrame = tk.Frame(self.parentFrame)
        self.cnnDetailFrame.pack()

        self.cnnDetailEntry = tkst.ScrolledText(self.cnnDetailFrame,
                                               height=7, wrap="word")
        self.cnnDetailEntry.insert(tk.INSERT, self.cnnDetails)
        self.cnnDetailEntry.config(state="disabled")
        self.cnnDetailEntry.pack(fill="x", pady=(0, 20))

        self.currCNNLabel = tk.Label(self.parentFrame,
                        text="Current hybrid CNN-SVM model used by the system",
                        font=ft.LARGE_FONT)
        self.currCNNLabel.pack(padx=10, pady=5)

        self.currFrame = tk.Frame(self.parentFrame)
        self.currFrame.pack(fill="x")

        self.currLabel = tk.Label(self.currFrame, text="Model:", font=ft.NORMAL_FONT)
        self.currLabel.pack(fill="x", side=tk.LEFT, padx=(10, 25), pady=10)

        self.currEntry = tk.Entry(self.currFrame)
        self.currEntry.insert(0, self.bestModelName)
        self.currEntry.config(state="readonly", readonlybackground='white')
        self.currEntry.pack(fill="x", side=tk.LEFT, padx=10, expand="True")
        self.currDetailLabel = tk.Label(self.parentFrame, text="Model details:",
                                        font=ft.NORMAL_FONT)
        self.currDetailLabel.pack(anchor="w", padx=10)

        self.currDetailFrame = tk.Frame(self.parentFrame)
        self.currDetailFrame.pack()

        self.currDetailEntry = tkst.ScrolledText(self.currDetailFrame,
                                                 height=7, wrap="word")
        self.currDetailEntry.insert(tk.INSERT, self.bestModelDetails)
        self.currDetailEntry.config(state="disabled")
        self.currDetailEntry.pack(fill="x", pady=(0, 20))

        self.buttonFrame = tk.Frame(self.parentFrame)
        self.buttonFrame.pack()

        self.homeBtn = ttk.Button(self.buttonFrame, text="Back to Home",
                                  command= self.backToHome )
        self.homeBtn.pack(side=tk.LEFT, padx=15, ipady=5, pady=10)

        self.useBtn = ttk.Button(self.buttonFrame, text="Use Selected Model",
                                 command=self.proceedToUseSelectedModel )
        self.useBtn.pack(ipady = 5, pady=(10,5))

    def modelNewSelection(self, event):
        cnnChoice = self.cnnDict.get(self.cnnCombo.get())
        cnnDetails = self.cnnDetailDict.get(self.cnnCombo.get())

        self.controller.createUpdateModel(cnnChoice, cnnDetails)

    def backToHome(self):
        self.controller.createHomePage()

    def proceedToUseSelectedModel(self):
        cnnChoice = self.cnnCombo.get()
        userAIExpert = self.controller.getCurrentAIExpert()

        baseCnnChoice = self.cboxHelper.getBaseString()
        if cnnChoice != baseCnnChoice:
            self.controller.updateCurrentModel(cnnChoice, userAIExpert)
            self.controller.createSuccessUpdateModel(cnnChoice, userAIExpert)
```

A..35 VisualAppealPredictor.py

```
from BinaryFeature import BinaryFeature

from HybridModel import HybridModel
import ast
from FileManipulator import FileManipulator

class VisualAppealPredictor():
    def __init__(self, file, age, gender, country, educLevel, currModelName):
        self.file = file
        self.age = age
        self.gender = gender
        self.country = country
        self.educLevel = educLevel
        self.currModelName = currModelName

    def getPrediction(self):

        #mapping
        fileMan = FileManipulator("assets/models/" + self.currModelName
                                  + "/mapping.txt")
        websiteFeatureDict = ast.literal_eval(fileMan.readContents())

        hybridModel = HybridModel("assets/models/" + self.currModelName
                     + "/" + self.currModelName, websiteFeatureDict)

        featureVector4096 = hybridModel.getFeatures4096([self.file])

        featureVector128 = hybridModel.getFeatures128(featureVector4096)

        binaryFeatures = BinaryFeature(self.age, self.gender, self.country,
                                       self.educLevel)
        featureVector47 = binaryFeatures.getFeatures()

        return hybridModel.getPrediction(featureVector128, featureVector47)
```

A..36 WelcomePage.py

```
import tkinter as tk
from tkinter import ttk #css
from PIL import Image, ImageTk
import Font as ft

class WelcomePage(tk.Frame):
    def __init__(self, parent, controller):
        tk.Frame.__init__(self, parent)

        self.controller = controller

        self.bannerFrame = tk.Frame(self)
        self.bannerFrame.pack()

        self.fileStr = "assets/images/banner.png"
        self.img = Image.open(self.fileStr)
        self.photo = ImageTk.PhotoImage(self.img)

        self.imgPanel = tk.Label(self.bannerFrame, image = self.photo)
        self.imgPanel.image = self.photo
        self.imgPanel.pack()

        self.parentFrame = tk.Frame(self)
        self.parentFrame.pack(expand="True")

        self.selectLabel = tk.Label(self.parentFrame, text="Please choose one",
                                    font=ft.TITLE_FONT)
        self.selectLabel.pack(padx=10, pady=(0, 10))

        self.frameRadio = tk.Frame(self.parentFrame)
        self.frameRadio.pack(fill="x", padx=50)

        self.radioVar = tk.IntVar()
        self.radioStyle = ttk.Style()
        self.radioStyle.configure('welcomepage.TRadiobutton',
                                  font=('Verdana', 20, "bold"))

        self.predictRadio = ttk.Radiobutton(self.frameRadio,
                                            text="I'm a Web Designer",
                                            variable=self.radioVar,
                                            value=0,
                                            style="welcomepage.TRadiobutton")
        self.predictRadio.pack(anchor="w", pady=10)

        self.datasetRadio = ttk.Radiobutton(self.frameRadio,
                                            text="I'm an AI Expert",
                                            variable=self.radioVar,
                                            value=1,
                                            style="welcomepage.TRadiobutton")
        self.datasetRadio.pack(anchor="w", pady=10)

        self.radioVar.set(0)

        self.proceedBtn = ttk.Button(self.parentFrame, text="Proceed",
                                     command=self.proceedWithChoice )
        self.proceedBtn.pack(ipady=5,  pady=30)

    def proceedWithChoice(self):
        choice = self.radioVar.get()
        if choice == 0:
            self.controller.createPredictVisualAppeal("assets/images/upmlogo.png", 0, 0, 0, 0, "")
        elif choice == 1:
            self.controller.createLogIn("", "", "")
```

XI. Acknowledgement

Unang-una sa lahat, gusto kong magpasalamat sa aking pamilya sa pagsuporta sa akin. Hindi ko makakamit ang mga pangarap ko kung wala kayo. Sa aking nanay na si Lutgarda at sa aking tatay na si Arlan, maraming salamat sa inyo! Sa mga kapatid ko na sina Aldrian, Kenneth, at Lea Anne, maraming salamat rin sa inyo.

Ikalawa, gusto ko rin magpasalamat sa aking adviser na si Sir Marvin John C. Ignacio! Sir, maraming salamat po talaga! Ang dami niyo pong nagawa para sa aking SP na kung minsan ay nahihiya na po ako sa inyo dahil halos kinukuha ko na po lahat ng oras niyo kapag ako po ay nagpapaconsult ng pagkatagal-tagal. Hindi ko po talaga makakalimutan yung araw na nagstay po kayo sa school ng hanggang 8pm dahil hindi ko pa po nakukuha yung go signal mula sa inyo. Kahit na wala po kayong klase sa araw na iyon, pumunta pa rin po kayo sa school para lang po sa mga magpapa-consult ng mga SP nila. Thank you po talaga, Sir!

Ikatlo, maraming salamat sa lahat ng mga kaibigan at mga kaklase ko na nakasama ko sa limang taon ko dito sa UPM. Sa lahat ng mga tawanan at iyakan dahil sa acads, hindi ko malilimutan ang mga iyon. Pero siyempre, may mga special mention yung mga taong nagbigay sa akin ng mga mahahalagang aral na aking dadalhin hanggang sa aking pagtanda.

Para sa mga barkada ko na sina John Paul, Clarence, Arky, Gerome, at Kyle, salamat sa samahang di na matitibag. Brothers forever! Tuloy tuloy lang tayo sa pag-abot sa ating mga pangarap.

Para kay Damian Fadri, salamat sa pagpupush sa akin na maging better pagdating sa acads. Gusto ko kasi yung mga pag-uusap natin after ng mga exams at pinag-aawayan pa natin kung sino ang may tamang solusyon sa ating dalawa. Natutuwa ako kasi natututunan ko talaga yung mga lessons natin kapag tayo na mismo yung nagtuturuan kung ano ang tamang solusyon sa hindi. Magiging kasinggaling din kita sa pagcocode at sa Math! Salamat rin sa pakikinig sa mga kwento ko na pagkahaba-haba.

Para kay Blanche Zabat, salamat sa halos dalawang taong tayo ay magkasama.

Salamat talaga sa lahat! Hindi ko na idedetail dito kasi nagkausap na rin tayo at nagkabati na. I wish you all the best in life! Sana lahat ng mga pangarap mo ay matupad mo. Maraming salamat talaga, ang dami mong nagawa para sa akin.

Para kay John Rafael "The African ****" Ferrer, salamat! Siyempre, main event ka kasi ang dami kong natutunan mula sa iyo. Sa iyo ko nakita yung pagsasabuhay ng Honor and Excellence! Yung mga kwento mo sa akin na binabasa mo talaga lahat ng readings kahit na alam mo na open notes ang exam, yung mga times na hindi mo gagawin ang isang bagay kasi "easy way out" yan, at maraming pang iba. Salamat talaga pre! Dahil sa iyo, natuto rin ako manindigan sa aking mga prinsipyo sa buhay. Hindi tayo kukuha ng isang subject dahil easy 1.0 yan, kukunin natin ang isang subject dahil doon tayo pinaka-matututo. Salamat sa lahat ng mga kwento at tawanan pre! Hanap na tayo ng ating mga girlfriend sa may La Salle. HAHAHA! Sana makamit natin pareho ang mga pangarap natin! Tuloy tuloy lang tayo sa pag-aaral at pagkatuto.

Lastly, maraming salamat sa universe! Alam ko may purpose ako sa mundong ito at unti-unti ko na yata nalalaman kung ano ang aking purpose sa mundong ito. Sana ay magampanan ko ito ng maayos.