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

Machine Learning-Driven Breast Cancer Diagnosis Software Integrated with Explainable Artificial Intelligence Based on Fine Needle Aspirate Findings

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

Tristan Paul Bachini

June 2023

Permission is given for the following people to have access to this SP:

Available to the general public	Yes
Available only after consultation with author/SP adviser	No
Available only to those bound by confidentiality agreement	No

ACCEPTANCE SHEET

The Special Problem entitled "Machine Learning-Driven Breast Cancer Diagnosis Software Integrated with Explainable Artificial Intelligence Based on Fine Needle Aspirate Findings" prepared and submitted by Tristan Paul Bachini in partial fulfillment of the requirements for the degree of Bachelor of Science in Computer Science has been examined and is recommended for acceptance.

Ma. Sheila A. Magboo, Ph.D. (*cand.*)
Adviser

EXAMINERS:

	Approved	**Disapproved**
1. Avegail D. Carpio, M.Sc.	_____	_____
2. Richard Bryann L. Chua, M.Sc.	_____	_____
3. Perlita E. Gasmen, M.Sc. (*cand.*)	_____	_____
4. Vincent Peter C. Magboo, M.D.	_____	_____
5. Marbert John C. Marasigan, M.Sc. (*cand.*)	_____	_____
6. Geoffrey A. Solano, Ph.D.	_____	_____

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

Vio Jianu C. Mojica, M.Sc.
Unit Head
Mathematical and Computing Sciences Unit
Department of Physical Sciences and Mathematics

Marie Josephine M. De Luna, Ph.D.
Chair
Department of Physical Sciences and Mathematics

Maria Constancia O. Carrillo, Ph.D.
Dean
College of Arts and Sciences

Abstract

Around the world, breast cancer remains to be the most frequent type of all cancers, and the major cause of death in women worldwide. A major factor in why the diagnosis of breast cancer through Fine Needle Aspiration results is still done after manual review of doctors, is because of the lack of explainability by the traditional black box machine learning models. This paper aims to incorporate a simple web user interface, and explainibility through the LIME python package. The performance of four machine learning models (K-Nearest Neighbors, Logistic Regression, Random Forest, and Support Vector Machine) were compared by its metrics (accuracy, precision, f1-score, and area-under-curve) produced when predicting breast cancer diagnosis, and its applicability with the LIME python package. The four models were utilized with the Breast Cancer Wisconsin Diagnostic Dataset with 10 different configurations a) only scaling applied, b) scaling then random oversampling, c) scaling, random oversampling, then feature extraction, d) scaling then feature extraction, e) scaling, feature extraction, then random oversampling. Configurations f-j are similar configurations, except it does not include scaling. The results show that in terms of metrics and applicability towards the LIME model, random forest with random oversampling produced the best results. As such, random forest with random oversampling was the model and configuration chosen to be applied towards the web application.

Keywords: LIME, random oversampling, accuracy, precision, f1-score, area-under-curve, explainability, support vector machine, logistic regression, random forest, k-nearest-neighbors, fine needle aspiration

Contents

Acceptance Sheet i

Abstract ii

List of Figures v

List of Tables vi

I. Introduction 1

A. Background of the Study 1

B. Statement of the Problem 2

C. Objectives of the Study 3

C..1 General Objectives 3

C..2 Specific Objectives 3

D. Significance of the Project 4

E. Scope and Limitations 5

F. Assumptions 5

II. Review of Related Literature 6

III. Theoretical Framework 10

IV. Design and Implementation 18

V. Results 21

A. Testing for the best performing model 21

B. Building the web application 29

VI. Discussions 33

A. Machine Learning Models 33

B. Objectives 35

VII. Conclusions **37**

VIII. Recommendations **38**

IX. Bibliography **39**

X. Appendix **41**

A. Source Code 41

XI. Acknowledgment **65**

List of Figures

1 Accuracy of the testing set 6

2 AUC computed from the different machine learning models. 7

3 Boston dataset taken from scikit-learn python library. 14

4 Sample output of LIME. 14

5 Overview of project implementation from acquisition of dataset to building the web application. 18

6 LIME sample output from dataset using feature scaling. 24

7 LIME sample output from dataset using feature scaling and PCA . . 24

8 LIME sample output from dataset using only oversampling 25

9 Visualization of slight imbalance in the dataset. 26

10 Visualization of dataset instances after applying random oversampling. 27

11 Elbow graph of PCA Variance on dataset without feature scaling. . . 28

12 Elbow graph of PCA Variance on dataset with feature scaling. 28

13 Homepage interface. 30

14 Page where FNA input values are entered. 30

15 After entering FNA values, the prediction and explanations are generated in this results page. 31

16 About page which contains instructions on how components are used and interpreted. 31

17 Updated overview of project implementation from acquisition of dataset to building the web application. 34

List of Tables

1 Table of metrics with feature scaling applied, given 5 different configurations with 4 machine learning models. 22

2 Table of metrics without feature scaling, given 5 different configurations with 4 machine learning models. 23

3 Confusion matrix of random forest model with resampling. 23

4 Numerical values of variance explained for figure 11 per principal component. 27

5 Numerical values of variance explained for figure 12 per principal component. 29

I. Introduction

A. Background of the Study

Around the world, breast cancer remains to be the most frequent type of all cancers, and the major cause of death in women worldwide. The most effective measure against breast cancer is early detection and prevention [1][2]. The reliability of data mining algorithms applied towards disease detection and diagnosis has been thoroughly proven through numerous studies [3][4][5], yet diagnosis is done by manual review of radiologists—which are prone to human errors. In fact, in 2020, an international team from Google Health and Imperial College London developed an AI model to predict breast cancer diagnosis, and it was found to be more accurate than doctors, when pitted against each other [6].

Fine Needle Aspiration (FNA) is a type of breast screening procedure that is among the least invasive of the available breast cancer screening procedures. Apart from the breast, FNA is also applied towards different parts of the body, such as the thyroid, to check for cancer. The process involves the insertion of a thin, hollow needle into a suspicious area, in order to extract a small amount of tissue or fluid, to be checked for cancer.

FNA can be used as a breast screening procedure, among many other breast cancer screening procedures. The advantages of the FNA against other breast screening procedures is its cost-effectiveness, low complication rate, and the rapidness at which it is administered [7].

As it is speedily applied towards a patient undergoing breast cancer screening, it has become a widely used and accepted breast cancer diagnostic tool. Because of its affordability, it is especially popular in developing countries such as the Philippines

[7].

The dataset to be utilized in the study is the Breast Cancer Wisconsin Diagnostic Dataset, with 569 instances of which 212 are malignant and 357 are benign. The dataset has 30 features, which is why the paper will explore dimentionality reduction techniques. Additionally, from the number of instances of each class, it can be said that there exists a slight data imbalance, which is why random oversampling will be applied.

B. Statement of the Problem

Despite the support of recent studies for the implementation of machine learning algorithms towards breast cancer diagnosis on FNA, the diagnosis remains to be a result of manual reviews by doctors—with reliability of diagnosis often relying solely on a FNA practitioner's skill [8]. A study showed that, in sub-Saharan Africa, there exists a lengthy buffer between a patient's first visit to a doctor for breast cancer screening, and this patient's first procedure to treat breast cancer. One of the main determinants of this buffer is a misdiagnosis on the part of the doctor [9]. Doctors sometimes tend to have different interpretations on FNA results– some doctors conclude that the cells are malignant, some do not. This is why asking for a second opinion from fellow doctors is common practice. Expounding on that idea, the implementation of artificial intelligence to augment human decision making has been shown to improve overall diagnostic accuracy. Such a tool would be useful as an AI assistant for a doctor assessing the FNA results of a patient, especially during the cases where a doctor does not have access to a second opinion.

There is also the variable of human errors that should be taken into consideration. There are a number of factors that can cloud the judgment of a human, in this

case, a doctor, such as being overworked or stressed, which hinders decision-making performance.

Another problem with the implementation of modern machine learning solutions towards the domain of medicine is the lack of reasoning and basis behind the predictions made by the model [10]. Since the doctor would want to know the how and why of the machine learning model predicting the diagnosis for a number of reasons including reasons of trust, sociality, legality, and practicality [10], then simply having a high reliability and a simple interface is not enough to practically apply machine learning towards cancer diagnosis.

C. Objectives of the Study

C..1 General Objectives

The study aims to solve the problem above by bridging the gap between the intricacies of machine learning and practical use by a licensed professional, by creating a simplified web application with Explainable Artificial Intelligence (XAI) component and maximized predicting power that can be utilized to assist with the diagnosis.

C..2 Specific Objectives

1. Train, test, and compare the performance of Random Forest, K Nearest Neighbors, Logistic Regression, and Support Vector Machines, on the prediction of breast cancer diagnosis.
    (a) Apply feature scaling to the dataset.
    (b) Apply feature scaling, then random oversampling to the dataset
    (c) Apply feature scaling, random oversampling, then feature extraction to the dataset.

(d) Apply feature scaling, then feature extraction to the dataset.

(e) Apply feature scaling, feature extraction, then random oversampling to the dataset.

(f) Train and test the Random Forest, K Nearest Neighbors, Logistic Regression, and Support Vector Machines, using the preprocessed dataset.

(g) Compare the 20 model performances to determine the best-performing model to be implemented in the system.

(h) Incorporate Explainable Artificial Intelligence through the Local Interpretable Model-Agnostic Explanations (LIME) Python package, to the optimized model.

2. Build the web application with the optimized machine learning model.

(a) Create a simple web application that takes input and produces an output decision variable

(b) Add quality-of-life (QOL) pages such as help, about, etc, in order to improve the overall experience and assist the user in using the web application, and interpreting the outputs.

(c) Host the website on a server.

D. Significance of the Project

As earlier mentioned, Breast Cancer remains to be the leading type of cancer afflicting women in the world. Although there are already multiple different breast cancer screening methods developed, the diagnosis made for the FNA procedure is ultimately made after manual review. The application aims to further validate the decision of the doctor, on top of the several screening methods, by allowing the input of real numbers associated with the features derived from FNA.

For breast cancer, early diagnosis is key. Sometimes, doctors are vulnerable to errors despite the available procedures [9]. The application of a high-performance machine learning algorithm with XAI to help with this decision will serve as another layer of ensuring the correct diagnosis is made. Such technology would be important for doctors and patients alike.

E. Scope and Limitations

1. The intended users are doctors who are in the process of deciding the diagnosis of a patient who had undergone FNA.

2. The machine learning model used in the study will utilize the Breast Cancer Wisconsin Diagnostic dataset from University of Wisconsin Hospitals Madison.

3. The system should only be used to augment, and not be the main decision maker in the diagnosis of a patient.

4. There is also a limitation on what constitutes a "good reasoning" in XAI as it is a topic still in active debate across various fields [10]. In this study, the reasoning that will be integrated in the project will be done through the LIME python package.

F. Assumptions

1. The user is a doctor who has knowledge on the procedure of FNA, as well as familiarity on the inputs being asked.

2. The user will only enter correct inputs.

3. The user has access to the values from the digitized image of the breast sample.

II. Review of Related Literature

There are numerous studies that have been done regarding the study, most of which show promising results in terms of the metrics provided by the machine learning models applied [3][5][4][11]. A system had also been implemented before, in a paper published in 1994 [11]. This tells us that ever since 1994, and possibly even before that, the feasibility of the implementation of machine learning towards breast cancer prediction through FNA results has been tested. Despite that, diagnosis remains to be a manual process. Most of these works were limited in its practical application, that is for actual use by a physician. As for the system that had been implemented, it lacks an integral tool that is necessary to acquire the trust of the physicians to whom this system aims to assist. That is the concept of XAI.

As earlier mentioned, what has been done regarding the problem that the paper aims to solve are extensive studies on machine learning models that best fit the challenge of predicting the diagnosis of breast cancer in patients. Performance of models were taken from different metrics such as accuracy, precision, and area-under-curve, as well as its applicability towards the LIME python package, which will be discussed in more detail later on. Across the different studies, accuracy was often the most-highlighted metric which ranged from 96-97 percent for the best-performing model. Aside from metrics, what will also be checked is how well the model fits with the LIME package.

Algorithms	Accuracy Training Set (%)	Accuracy Testing Set (%)
SVM	98.4%	97.2%
Radom Forest	99.8%	96.5%
Logistic Regression	95.5%	95.8%
Decision Tree	98.8%	95.1%
K-NN	94.6%	93.7%

Figure 1: Accuracy of the testing set

Algorithms	AUC (%)
SVM	0.966
Random Forests	0.960
Logistic Regression	0.947
Decision Tree	0.945
K-NN	0.952

Figure 2: AUC computed from the different machine learning models.

For example, Naji, et al [3] ran the different machine learning models utilizing the Breast Cancer Wisconsin Diagnostic dataset, and was able to extract the metrics above, which are relatively good metrics to have. Across the different studies, the best-performing model also varied between logistic regression and support vector machines. It should also be noted that these journals made use of the same dataset, the Wisconsin Breast Cancer dataset, that will be applied in this study [3][5][4][11]. As such, it can be deduced that the reason for these variations in metrics would be the fine tuning or preprocessing techniques applied towards the dataset. Upon analysis of their approaches and limitations, a commonly recurring limitation is the selection of preprocessing techniques that was applied towards the dataset. Also, several studies did not mention whether the model performances were compared before and after application of each preprocessing technique.

Gbenosi et al. examined the factors that determined the length of the buffer between a patient's first screening for breast cancer, and this patient's first day of breast cancer treatment. The study concluded by naming diagnostic errors as one of the main determinants of the length of this buffer [9]. As this study is a paper on the application of machine learning algorithms to aid in the diagnosis of a patient of breast cancer, and its implementation for a full-fledged system that can potentially be used by doctors, then this system is aimed at directly giving solution to that main determinant cited by the study.

XAI has also been applied before towards the prediction of Autism Spectrum Disorder (ASD) of a patient exhibiting a specific set of behaviors. The performance of the machine learning algorithms, in terms of its metrics, proved to be highly reliable, and this reliability came with the transparency that XAI provided. The system not only predicted with a certain amount of accuracy, but also it gave insights to how the algorithm arrived at a certain conclusion. With that said, it was stated in the paper that this insightful and reliable system is potentially something of practical use by physicians to predict autism in children [12]. This is exactly the aim of this paper: create a system that predicts breast cancer from FNA results, that could be of practical use by doctors in predicting the diagnosis of breast cancer. In that sense, since reliability has already been proven by the extensive research done, which resulted in a 96-97 percent-accuracy classifier model, we must then provide the insight, through the application of XAI, and its implementation into a system that can be put into practical use in the medical domain.

There are many reasons as to why XAI is important in the context of integration of machine learning systems towards the different sectors of society. These reasons include legal, practical, and social reasons. Among those reasons is that, with XAI, we are able to enhance the robustness of a system, since it helps with troubleshooting. Apart from that, the ability to provide reasoning to how a decision was made has become an increasingly desirable property of intelligent systems. All of these combined would directly relate to the users' trust and persuasion, which are essential variables to consider in this paper [10][13].

If we analyze each article, we conclude that each journal study falls short of being of practical use by doctors in predicting the diagnosis of breast cancer. There are three components that will allow for its potential use in the medical domain. As concluded in the study of the prediction of ASD in children, having reliability and insight in a

machine learning model is enough for the idea to be potentially applied in medicine [12]. However, for the gap between computer technicality and medicine to be truly bridged, then it requires a user interface that will take in the input, and automatically run it through the chosen machine learning algorithm, to extract the decision variable, as well as the reasoning behind it. This is how the study will augment all of these studies; by incorporating all three components, that is: reliability, reasoning, and a user interface.

The study will be a synthesis of past work on breast cancer diagnosis based on machine learning, diagnosis systems with integrated XAI, and a user interface that simplifies the process of input and output. The system to be implemented will address the following hindrances towards the implementation of machine learning models in the medical domain: trust, accuracy, and simplicity. By doing so, we create a readily-applicable machine learning-driven software system that predicts the diagnosis of breast cancer from FNA results.

III. Theoretical Framework

Breast Cancer

Breast cancer is a phenomenon that occurs when cells in the breast grow abnormally. It is the most common form of cancer among women globally, as well as the leading cause of mortality in women worldwide [2].

Fine Needle Aspiration

Fine Needle Aspiration is a type of breast screening procedure that is among the least invasive of the available breast cancer screening procedures. Apart from the breast, FNA is also applied towards different parts of the body, such as the thyroid, to check for cancer. The process involves the insertion of a thin, hollow needle into a suspicious area, in order to extract a small amount of tissue or fluid, to be checked for cancer. It is rapid, cost-effective, and accurate, which makes it a popular breast cancer screening procedure in developing countries such as the Philippines [7].

Dataset

The dataset used to train the different machine learning models in the study is the Breast Cancer Wisconsin Diagnostic Dataset. In this dataset, 10 features that have real values are taken into consideration to predict whether the observation is classified as benign or malignant. The 10 features, namely: radius, texture, perimeter, area, smoothness, compactness, concavity, concave points, symmetry, and fractal dimension, were computed from a digitized image of the fine needle aspirate of a breast mass. For each feature, the mean, standard error, and the largest of these features were computed, resulting in 30 attributes, excluding the ID and decision variable, for a total of 32 attributes. The dataset has a total of 569 observations, of

which 357 are classified as benign, while the remaining 212 are classified as malignant.

Feature Scaling

Feature scaling is a preprocessing technique used in order to normalize the different values in the dataset so that each value would weigh the same whether they be large or small. This prevents one large value skewing the way the machine learning model learns. This is applicable to the study, since the dataset contains only real values.

Resampling

Resampling is the creation of more samples based on a sample. It is used to deal with the problem of having imbalance in a dataset. There are a few techniques that can be used, such as upsampling and downsampling.

Feature Extraction

Feature extraction is the process of lessening the number of attributes in the dataset. This is done in order to lessen the complexity with regards to the data being dealt with, and potentially increase the predicting power of the machine learning model.

Linear Discriminant Analysis

Linear Discriminant Analysis is a supervised method of feature extraction used to reduce the dimensionality of a dataset. LDA maximizes the distance between the mean of each class and minimizes the spreading within the class itself.

Machine Learning

Machine Learning is a subfield of AI that allows computers to learn from existing data, and draw inferences without explicitly being instructed, through the use of different models and statistical tools [14].

Explainable Artificial Intelligence

XAI is a component of artificial intelligence that provides the "reasoning" on which how the computer arrived at a decision. It adds on to the feasibility of having a prediction system aid physicians' diagnosis of breast cancer, since XAI allows for more transparency during the decision-making process of the computer [13].

XAI adds towards the interpretability of a machine learning model. There are many goals that exist that define the reasons on why there is a need for interpretability. Lipton mentions 3 goals for interpretability: a means to engender trust, a desirable model that may help uncover causal structure in observable data, and a means to simply gather more information [13].

First, there is interpretability as a means to engender trust. This is because XAI helps a user understand how and why a decision was made [10]. Transparency and scrutability are key characteristics of XAI that fosters the trust of its users, whereas transparency allows for the understandability of the model for the user, and scrutability allows for the user to tell when the machine learning model may have made a mistake [13].

Second, interpretability as a desirable model that may help uncover causal structure in observable data. Machine learning, when applied in different contexts, has its social, practical, and legal aspects to be considered. These aspects is what interpretability aims to satisfy. For example, when legal action is taken against a user who made a controversial decision augmented by a machine learning model, having

an explanation for how that decision was made allows for a defense from the user. This highlights the need to know the causal structure behind the decision-making of the machine learning model [10].

Third, interpretability as a means to simply gather more information. This is a general goal for the need of interpretability of machine learning models, since the information gathered can be used to accomplish a set of varying goals. One example is using the information to test the accuracy of the machine learning model as time passes by, since sometimes datasets that are used to train machine learning models lose relevance over time. By understanding how a machine learning model arrives at a decision, engineers pinpoint and troubleshoot the errors in the model.

Schemmer et. al. (2022) states that there is an observable positive impact of XAI towards user performance, versus the use of machine learning without XAI. In this context, user performance is defined by the manual user's decision-making performance [15].

Local Interpretable Model-Agnostic Explanations

Local Intepretable Model-Agnostic Explanation is a method of application of explainable artificial intelligence to provide the reasoning as to how the model had arrived at a particular decision variable.

	CRIM	ZN	INDUS	CHAS	NOX	RM	AGE	DIS	RAD	TAX	PTRATIO	B	LSTAT	target
464	7.83932	0.0	18.10	0.0	0.655	6.209	65.4	2.9634	24.0	666.0	20.2	396.90	13.22	21.4
290	0.03502	80.0	4.95	0.0	0.411	6.861	27.9	5.1167	4.0	245.0	19.2	396.90	3.33	28.5
273	0.22188	20.0	6.96	1.0	0.464	7.691	51.8	4.3665	3.0	223.0	18.6	390.77	6.58	35.2
144	2.77974	0.0	19.58	0.0	0.871	4.903	97.8	1.3459	5.0	403.0	14.7	396.90	29.29	11.8

Figure 3: Boston dataset taken from scikit-learn python library.

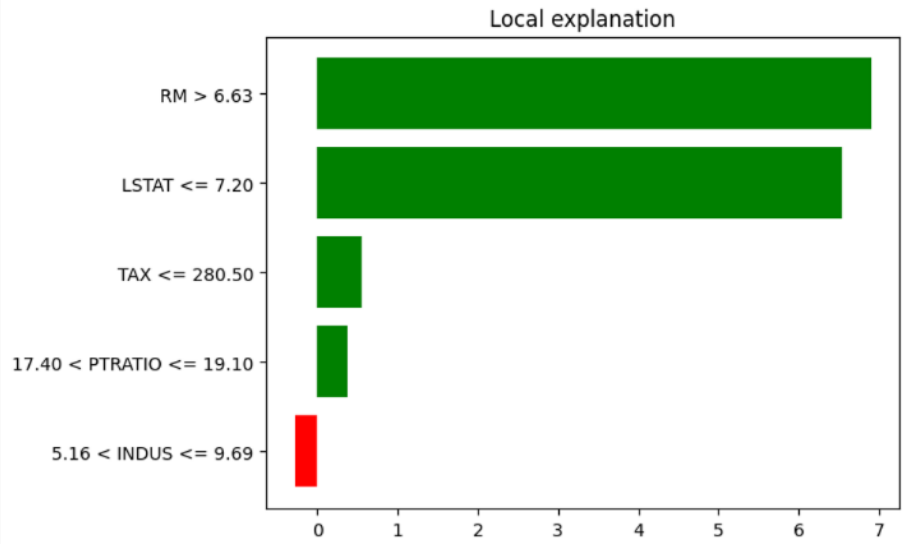

Figure 4: Sample output of LIME.

Above, we have a sample output of the LIME python package. The dataset utilized is the boston dataset taken from the sci-kit library. The model attempted to predict the 47th data point in the testing dataset. LSTAT stands for lower status of society, RM stands for amount of room per dwelling, TAX is the tax-rate of the property, PTRATIO is the ratio of students to teacher, and INDUS is the number of non-retails near the society.

The model predicts the value of a property around Boston. For this datapoint, the value predicted was 34.324 units. Looking at the sample output of LIME, we are able to understand the reasoning behind this prediction of the value of the property. We can notice that RM and LSTAT played a major role in the prediction of a relatively high value (depicted by the majority of the bars being positive). The amount of room per dwelling, which had a value greater than 6.63, and a low value of LSTAT, being below or equal to 7.20 units. In other words, since the property had a relatively larger amount of room per dwelling, as well as a grand status of its society in terms of its education and employment, then it resulted in a relatively higher predicted property value.

Supervised Learning

Supervised learning is a subgroup under machine learning that deals with data columns that have labels, for both inputs and outputs. Since the dataset that will be used in the study is labelled, then the machine learning models that will be applied are supervised learning models.

K-Nearest Neighbors

K-Nearest Neighbors is a supervised learning method that predicts based on the proximity of other data points near it or its "neighbors". An odd value k is selected,

and the k number of datapoints near the datapoint being predicted are considered when making the prediction.

Logistic Regression

Logistic Regression is a supervised learning method where the linear combination of the independent variables are taken into account, and used in order to predict the the dependent variable.

Support Vector Machine

Support vector machine is a supervised learning method where the optimal hyperplane is found by maximizing the margin between the two sets of classification. The datapoints nearest to the hyperplane are called the support vectors, and it is between these support vectors and the hyperplane that we want to maximize the distance, that is the margin.

Random Forest

Random Forest is a supervised learning method that is based on multiple decision trees. It is an ensemble-type model, meaning that it utilizes multiple decision trees, and aggregates their decisions. The prediction that is made by the most decision trees will be the decision made by the random forest model.

Area-Under-Curve

The Area-under-curve (AUC) tests for the area under the Receiver Operator Characteristic (ROC) curve. A value of AUC that is close to 1 suggests that the classifier is good, while a value of 0.5 suggests that the classifier is not able to distinguish between positive and negative class points.

Precision

The precision calculates the number of correctly classified positive predictions, over the total number of positive predictions. It measures the performance of a model in terms of its ability to classify a sample as positive.

Accuracy

The Accuracy is used in order to get the general performance of the model across the classes, and is often used when classes are of equal importance.

F1 Score

The F1 score is the harmonic mean between the precision and recall. Its best value is at 1, and its worst value is at 0.

IV. Design and Implementation

The goal is to create a software system that integrates the best performing machine learning model before and after the application of different preprocessing techniques. The system is a web application, with a simple user interface that accepts input, and displays output. It will also have a help page, that will assist users and give more information regarding the web application.

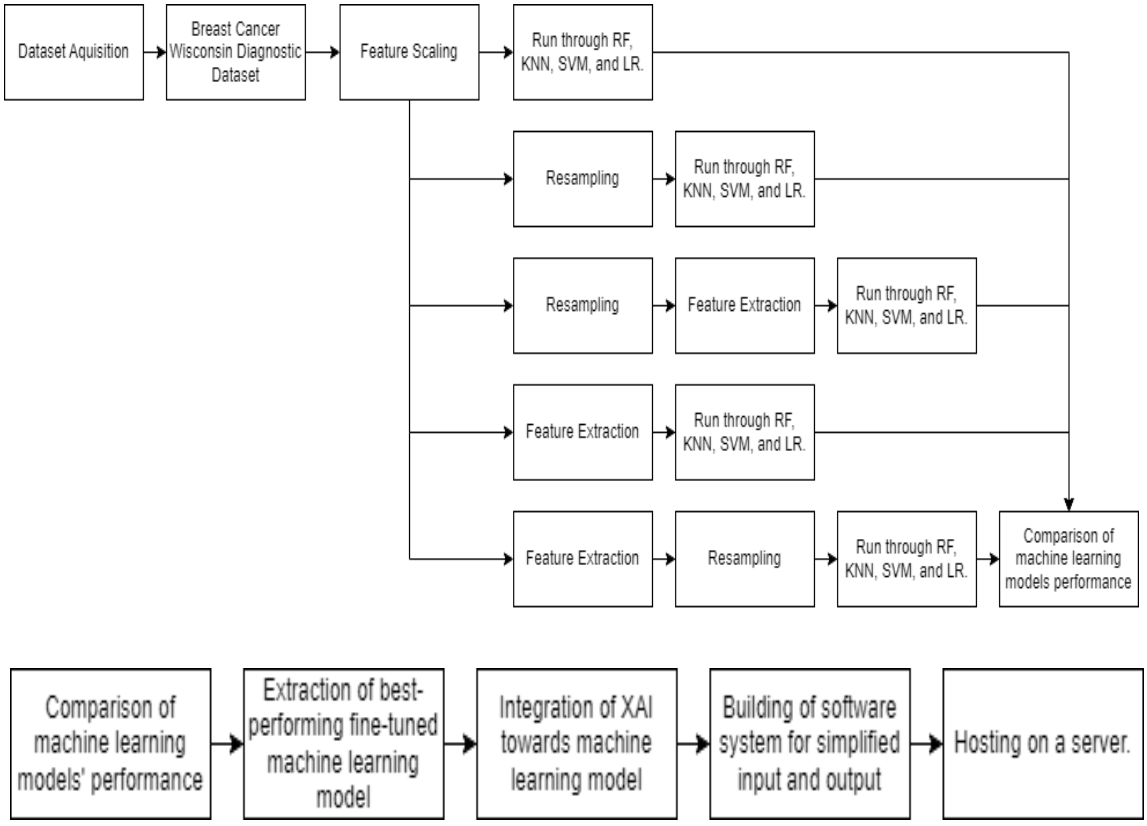

Figure 5: Overview of project implementation from acquisition of dataset to building the web application.

The target audience are the doctors or radiologists who extract the breast tissue sample, and the pathologists to whom the breast tissue sample is sent to in the laboratory [16]. Since both parties are involved in the decision-making process of classifying a breast tissue sample as benign or malignant, the system is aimed towards them.

Different machine learning methods performance will be tested with the dataset, and different preprocessing techniques will be applied to maximize model metrics. The machine learning models that will be included in the study are K-Nearest Neighbors (KNN), Logistic Regression, Support Vector Machine (SVM), and Random Forest. Aside from the implementation of different machine learning models, different fine-tunings will be tested on the dataset via preprocessing techniques namely feature scaling to normalize the range of values, feature extraction, and resampling techniques. The resulting metrics will be compared with one another, and each fine-tuning of the dataset will be tested on the LIME model interpreter to determine its applicability.

The fields of medicine and machine learning are two different domains. For the target user to be able to take advantage of the already-proven diagnostic reliability of machine learning models, then the technical gap must be filled such that the target users with little to no exposure to machine learning are able to integrate the proposed system towards diagnosis of patients who had undergone breast cancer screening. To do this, a web application software will be developed such that running an observation on the best-performing machine learning algorithm is as simple as entering an input and pushing a button. To determine the best-performing algorithm, first, there are the metrics, where slightly more emphasis will be placed on the F1-score, AUC, and precision metrics, since there is slight imbalance on the dataset. Second, there is the suitability of the fine-tuned model towards the LIME interpreter.

For the completion of the study, the tools that will be used include the sklearn and pandas packages in order to apply the different machine learning models as well as the different preprocessing techniques towards the dataset. The Lime package is used in order to provide reason and explanation to the prediction of the machine learning model. In other words, we integrate XAI to the project through the Lime python package. Towards building the web application, Django will be the framework to be

used for the backend. MySQL is the database that will be used, and Bootstrap for the frontend.

V. Results

A. Testing for the best performing model

Following the original objectives resulted in models that were substandard. The reason will be expounded on in more detail later in this section. To fix this, a few more configurations of models were added. From 20 models, this section will now compare 40 differently configured models.

Logistic Regression, Random Forest Classifier, K-Nearest Neighbors, and Support Vector Machine were tested on 10 different variations of applications of feature scaling, resampling, and feature extraction. This resulted in 40 different sets of metrics to be compared with one another.

From tables 1 and 2 below, we can see 40 different potential models, of which the best-performing model would be applied towards the web application. 10 fold cross validation was used, and then the mean of each metric of all folds was extracted to be the final metric. Highlighted in yellow and green are the models that were considered to be applied towards the web application from its metrics alone, which is indicative of its effectiveness in correctly predicting the presence of breast cancer. Upon analysis of the table, we can observe high-valued metrics across most of the differently fine-tuned datasets. The model highlighted in green is chosen to be applied towards the web application, based on its performance. Table 3 below lists down the confusion matrix of this model. It should be noted that the metrics taken in tables 1 and 2 were taken using 10-fold cross validation. The confusion matrix, on the other hand, was taken after running one instance of the model. One of the main basis for selecting the random forest with oversampling applied is the value of its area-under-curve, with the other metrics supporting this decision. This is because the auc takes into consideration each of the four elements in the confusion matrix, making it into a

Feature Scaling				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.957863	0.940001	0.961753	0.991594
K-Nearest Neighbors	0.964787	0.950533	0.980627	0.984502
Support Vector Machine	0.975376	0.966486	0.977096	0.995513
Logistic Regression	0.98067	0.973492	0.986107	0.996172
Feature Scaling then Resampling				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.983216	0.982072	0.973444	0.999414
K-Nearest Neighbors	0.973435	0.973656	0.972823	0.988785
Support Vector Machine	0.978991	0.97898	0.98647	0.996473
Logistic Regression	0.981788	0.981724	0.986186	0.997013
Feature Scaling, Resampling, then Feature Extraction				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.983236	0.984808	0.978433	0.999489
K-Nearest Neighbors	0.973435	0.973656	0.972823	0.988785
Support Vector Machine	0.978991	0.97898	0.98647	0.996473
Logistic Regression	0.981788	0.981724	0.986186	0.997013
Feature Scaling, then Feature Extraction				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.956109	0.94744	0.962254	0.990011
K-Nearest Neighbors	0.964787	0.950533	0.980627	0.984502
Support Vector Machine	0.975376	0.966486	0.977096	0.995513
Logistic Regression	0.98067	0.973492	0.986107	0.996172
Feature Scaling, Feature Extraction, then resampling				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.758979	0.779984	0.71679	0.760679
K-Nearest Neighbors	0.638498	0.668279	0.618855	0.690906
Support Vector Machine	0.513869	0.604765	0.509651	0.530705
Logistic Regression	0.512617	0.55734	0.511165	0.530705

Table 1: Table of metrics with feature scaling applied, given 5 different configurations with 4 machine learning models.

robust metric to measure the predictive power of this model.

No preprocessing technique applied				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.961341	0.954323	0.956728	0.989943
K-Nearest Neighbors	0.92619	0.897563	0.928011	0.950248
Support Vector Machine	0.954323	0.936822	0.958166	0.989716
Logistic Regression	0.942043	0.920452	0.939575	0.99143
Resampling				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.981808	0.986105	0.975802	0.999491
K-Nearest Neighbors	0.952387	0.953106	0.946764	0.984209
Support Vector Machine	0.96207	0.961738	0.966723	0.992899
Logistic Regression	0.943916	0.943552	0.949685	0.989685
Resampling then Feature Extraction				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.981827	0.984884	0.981342	0.999493
K-Nearest Neighbors	0.952387	0.953106	0.946764	0.984209
Support Vector Machine	0.96207	0.961738	0.966723	0.992899
Logistic Regression	0.943916	0.943552	0.949685	0.989685
Feature Extraction				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.961404	0.959085	0.953185	0.990267
K-Nearest Neighbors	0.92619	0.897563	0.928011	0.950248
Support Vector Machine	0.954323	0.936822	0.958166	0.989716
Logistic Regression	0.942043	0.920452	0.939575	0.99143
Feature Extraction, then resampling				
Model	Accuracy	F1 Score	Precision	Area under curve
Random Forest	0.787011	0.804577	0.747033	0.789243
K-Nearest Neighbors	0.647046	0.680604	0.623807	0.714687
Support Vector Machine	0.52099	0.626235	0.51376	0.556248
Logistic Regression	0.526624	0.603239	0.519114	0.556248

Table 2: Table of metrics without feature scaling, given 5 different configurations with 4 machine learning models.

90	1
3	49

Table 3: Confusion matrix of random forest model with resampling.

Performance is based off of 2 factors. First, the values of its metrics namely accuracy, F1 score, precision, and AUC. These values range from 0 to 1, where 1 denotes a model that perfectly correctly predicts breast cancer. Second, its suitability towards being used in tandem with the LIME package. Although there exists in the tables other models with slightly better metrics that are highlighted in yellow, it is worth mentioning that it was observed that application of PCA or feature scaling towards the dataset yielded sub-optimal explainability, which is shown below.

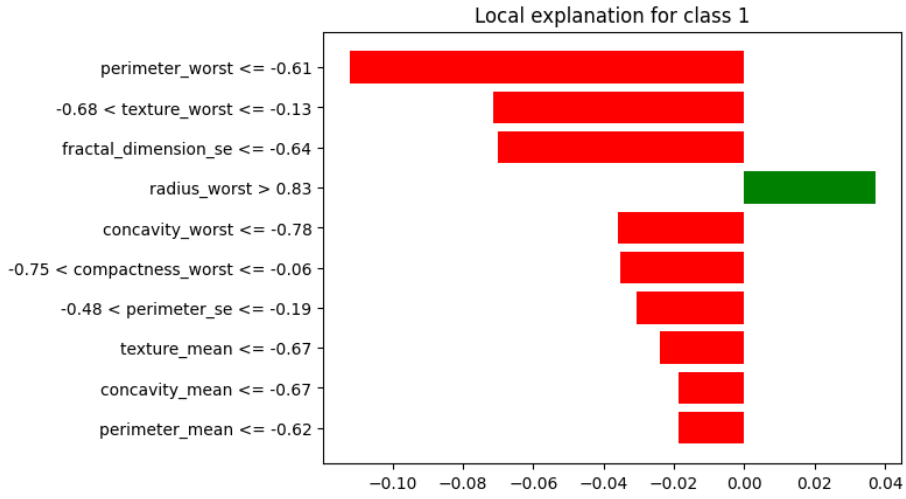

Figure 6: LIME sample output from dataset using feature scaling.

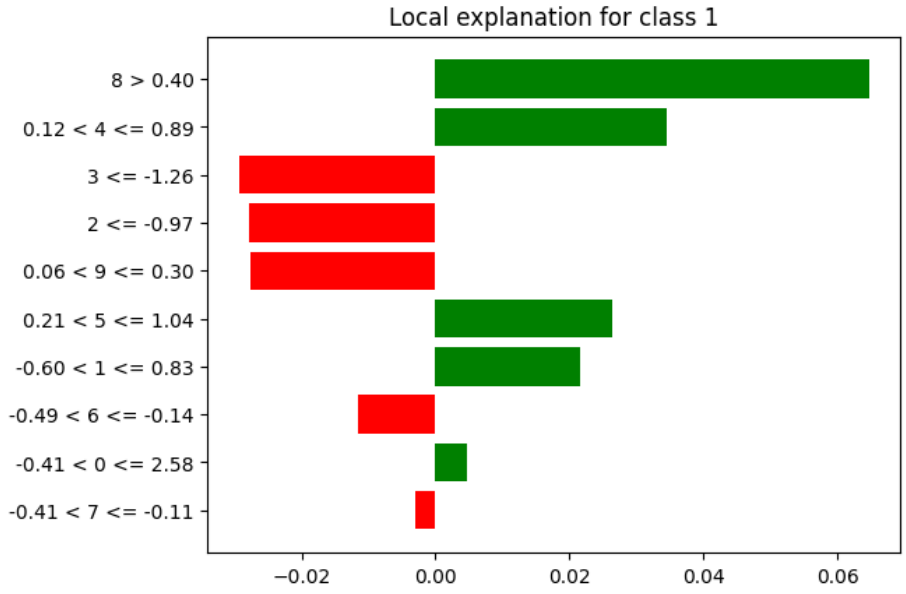

Figure 7: LIME sample output from dataset using feature scaling and PCA

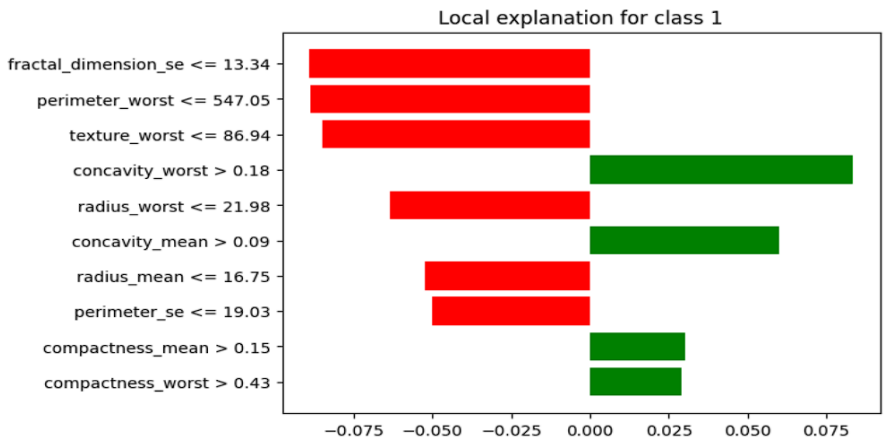

Figure 8: LIME sample output from dataset using only oversampling

In figures 6, 7, 8, we can compare the differences between the fine-tuning of the machine learning models. Figure 8 shows the fine-tuning used in the model highlighted in green, that is integrated in the web application. It should be noted that not the same datapoint is being tested, so the actual values can be disregarded, as well as the machine learning algorithm. What should be taken note of are the feature names listed on the y-axis. In figure 7, instead of column names from the original dataset, the LIME python package utilized the new column names after running PCA. Column names were designated from 0 to 9, for a total of 10 columns, which was the number of principal components selected.

Additionally, comparing figures 6 and 7, and figure 8, it can be observed that the values on the y axis of figures 8 and 9 are scaled. As of writing, there is no function in the lime library that was used that enables one to extract these values in the y axis, and apply inverse scaling techniques.

Since a principal component is a linear combination of, or in other words, an aggregate of, the initial variables that are used in the dataset, the LIME package cannot determine how each initial variable in a principal component impacted the final decision variable. What happens is that LIME takes a principal component as

it is. The problem with this, as can be observed in figure 7, is that the Explanation object cannot properly convey to the user pertinent information regarding how a decision was made, since feature names such as 0 to 9 provide zero explainability, whereas in figures 6 and 7, features can be distinguished from one another, and thus ample explainability is provided. This is why PCA as a feature extraction method is not suitable, if the model will be paired afterwards with LIME as an explainability model. This is why it is concluded that a model fine-tuned without scaling and PCA is most suitable to be applied towards the lime python package, to offer maximum explainability.

In an attempt to address the slight imbalance in the dataset,and possibly yield a better set of metrics, oversampling was used as a resampling method.

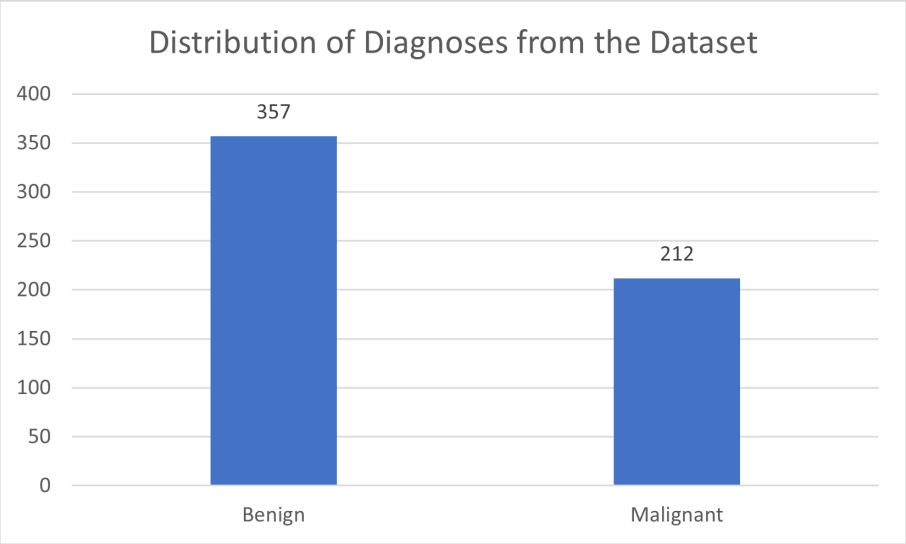

Figure 9: Visualization of slight imbalance in the dataset.

Figure 9 shows the class imbalance in the dataset. Oversampling was applied to the minority class, that is the malignant breast cancer diagnosis, in order to be of

equal occurrences as the benign diagnoses. We are able to see a visualization of this in figure 10 below.

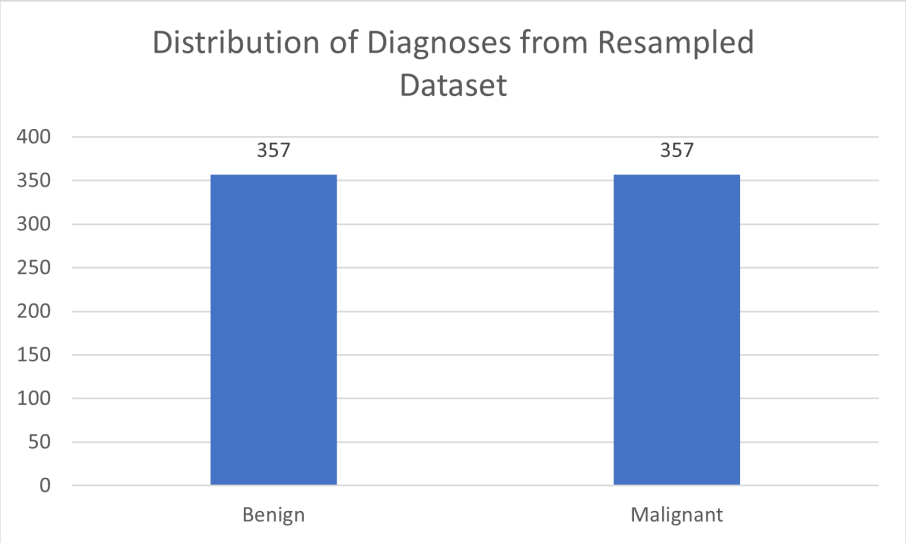

Figure 10: Visualization of dataset instances after applying random oversampling.

As such, datasets which underwent resampling have 714 samples, while the rest have the original 569

PCA was applied to both datasets which did and did not undergo feature scaling. The number of principal components varied between these two configurations, as variability has been tested by the elbow method (figures 11 and 12) and actual values for variability (tables 4 and 5).

Principal Component	Variance
PC1	0.9840
PC2	0.0148
PC3	0.0011
PC4	0.0001
PC5	0.0001
PC6	0.0000

Table 4: Numerical values of variance explained for figure 11 per principal component.

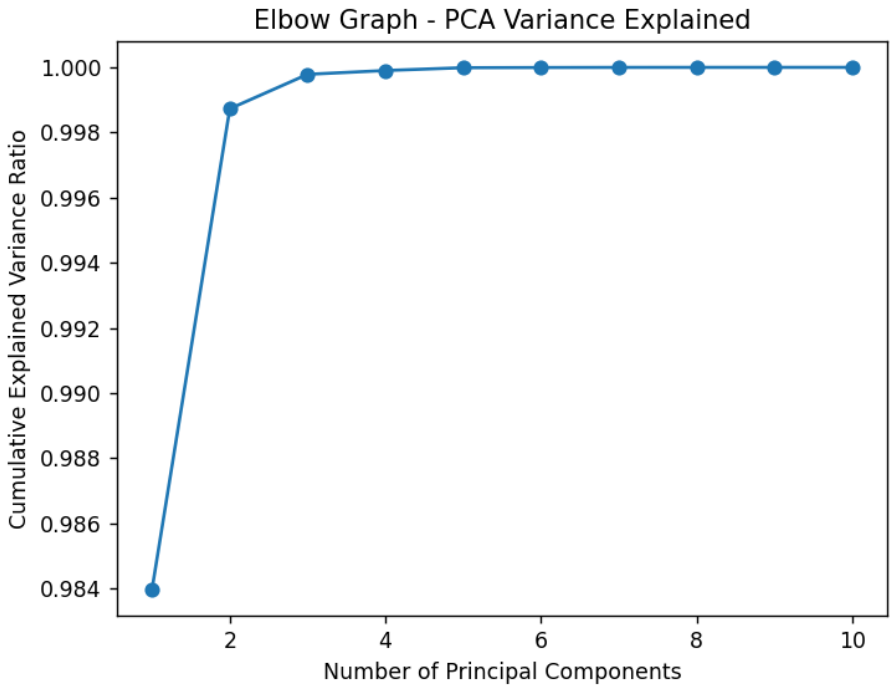

Figure 11: Elbow graph of PCA Variance on dataset without feature scaling.

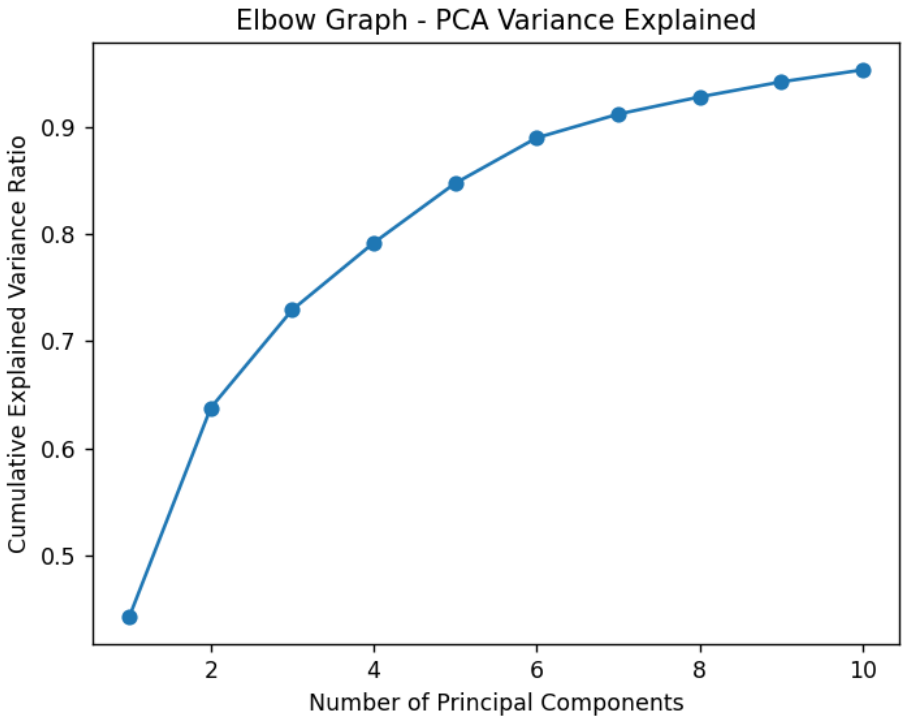

Figure 12: Elbow graph of PCA Variance on dataset with feature scaling.

Principal Component	Variance
PC1	0.4428
PC2	0.1951
PC3	0.0915
PC4	0.0625
PC5	0.0556
PC6	0.0424
PC7	0.0221
PC8	0.0161
PC9	0.0141
PC10	0.0111

Table 5: Numerical values of variance explained for figure 12 per principal component.

Comparing the elbow graphs in figure 11 and 12, the number of principal components to be used is more obvious in the model that did not undergo feature scaling versus the model that did. This is apparent in the flattening of the line, which indicates that from principal component n to n+1, the increase in variance explained is less than the increase from principal component n-1 to n. When the lessening of variance explained is significant enough, we select that amount of principal components to use towards the models. That is how 1 principal component was selected in the models without feature scaling, and 3 were selected in the models with feature scaling.

B. Building the web application

The web application consists of 4 pages: The predict page where a user may input the 30 necessary attributes, then lead to the results page where the prediction and explanation will be displayed in graphs and sentences, the about page where the user may learn more about the different components that appear in the results and how they are interpreted, and the homepage which contains a small introduction on the

web application and a links to the predict page and the about us page. Additionally, all pages are linked by a navigation bar to make it easier to browse through the web pages.

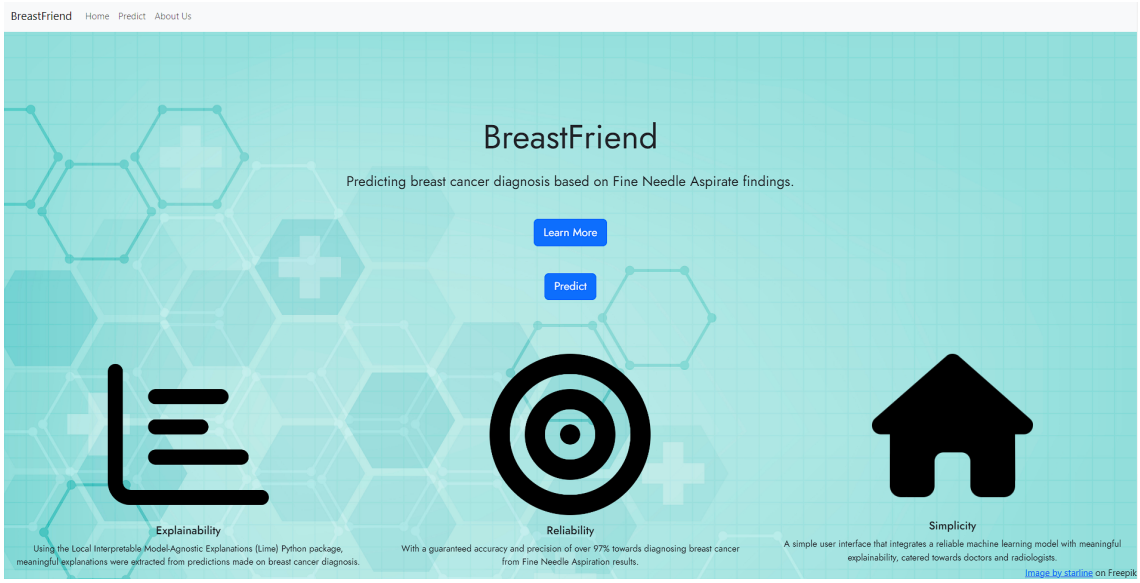

Figure 13: Homepage interface.

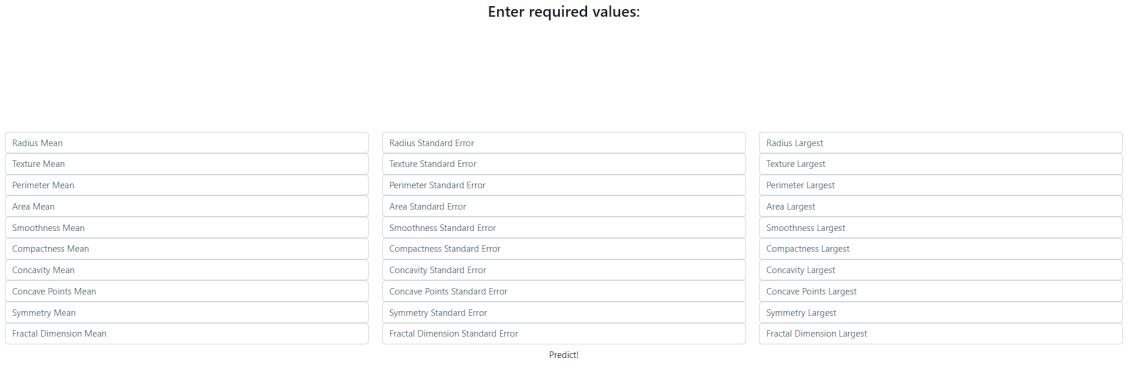

Figure 14: Page where FNA input values are entered.

Figure 15: After entering FNA values, the prediction and explanations are generated in this results page.

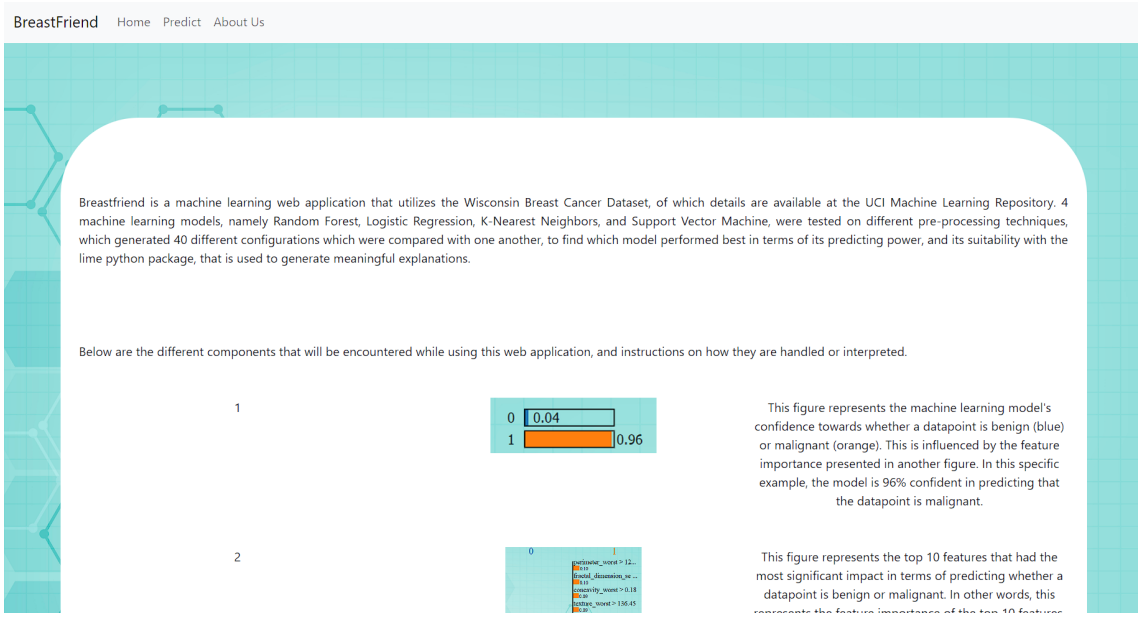

Figure 16: About page which contains instructions on how components are used and interpreted.

Figures 13 to 16 show the different pages of Breastfriend. Figure 13 is the homepage, where a user first lands upon entering the base url in the search bar. It gives a brief explanation on the main features of the application mainly reliability, explainability, and simplicity. Figure 14 shows the page where FNA values are input to create the datapoint that will be predicted. Figure 15 is the results page that is generated after pressing the predict button from the input page. It contains the prediction probabilities, feature importance, input values of the top 10 features in terms of impact, a plot generated from matplotlib, the prediction of whether the datapoint is benign or malignant, the accuracy at which the prediction was made, and the feature importance explanation in textual form. Finally, figure 16 shows the about page, which contains a link to where the dataset used in the application was sourced from, as well as instructions on how the different components in the application are interpreted. This page also serves as the help page.

VI. Discussions

A. Machine Learning Models

Initially, only 20 machine learning models were considered for this paper. Feature scaling was to be applied to all models, before undergoing resampling or feature extraction techniques. However, upon comparison of the machine learning models, it was found that the models produced sub-optimal results in terms of the degree of explainability demonstrated. Issues with feature scaling are highlighted in figures 6 and 7. Additionally, issues with PCA as a feature extraction technique are highlighted also in figure 7. As such, it was decided to include a repeat of all configurations of preprocessing techniques, without feature scaling. As such, from originally 20 configurations to choose from, there are now 40. The updated pipeline is shown in figure 17 below.

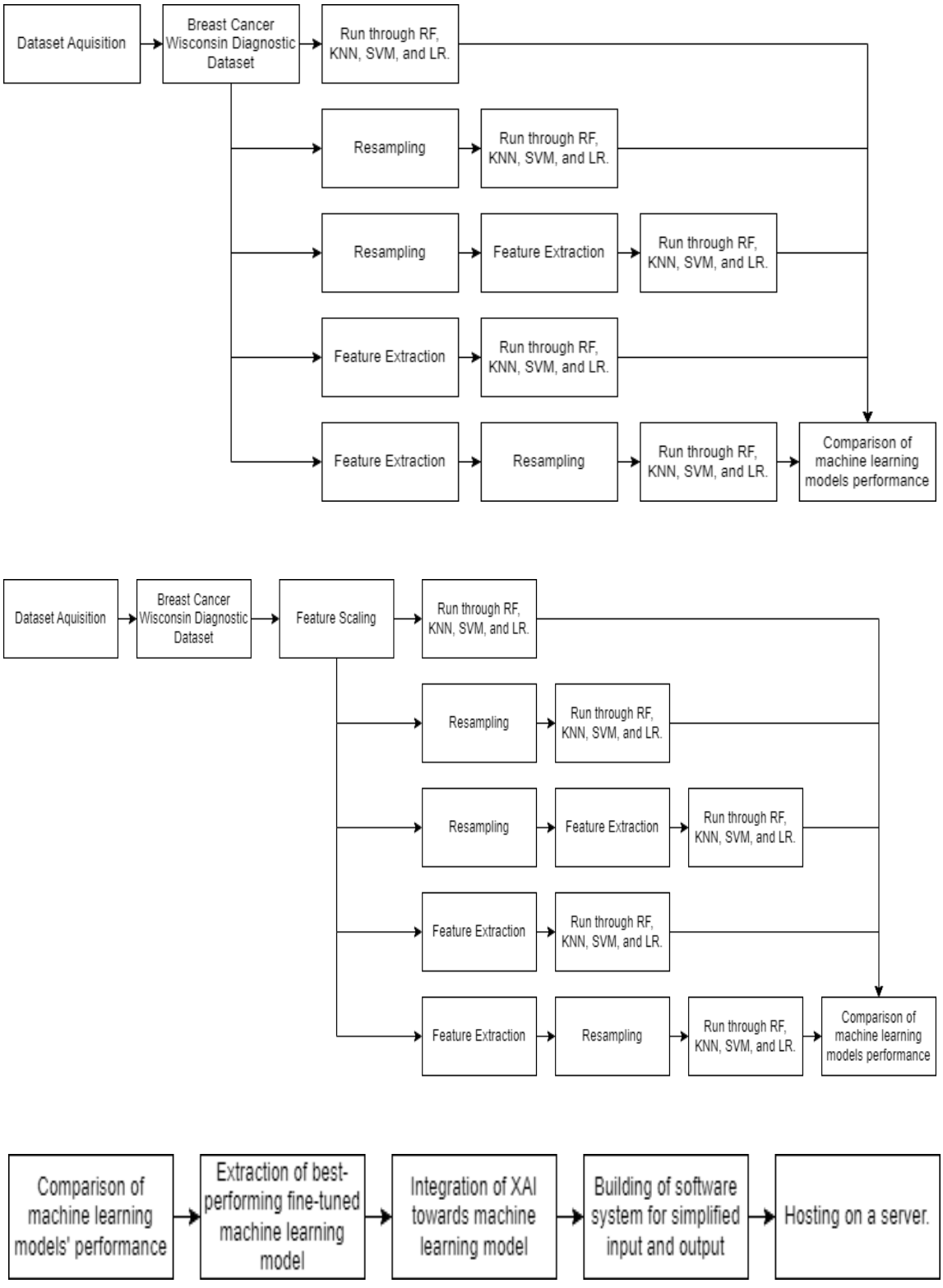

Figure 17: Updated overview of project implementation from acquisition of dataset to building the web application.

The machine learning models metrics shown in tables 1 and 2 are limited by the application of feature scaling and PCA which were found to be unsuitable towards the LIME model, which is why despite slightly better metrics from other configurations, random forest with only oversampling applied was ultimately chosen. The difference in metrics is only minimal. This is the reason why the pipeline was updated with additional configurations, in order to maximize the explainability of the LIME python package. Because 10-fold cross-validation was used in testing each configuration of machine learning model, the listed metrics are consistent.

While there is a method in order to unscale values that were scaled using StandardScaler() from the scikit-learn library, it was not possible to extract the values that needed to be unscaled from the graphs.

Selection of the number of principal components was done visually by analyzing the elbow graph and referring to the table of values. Whenever there is no more significant gain in the variance explained by the next principal component, then we retain that amount of principal components.

B. Objectives

The final program was able to achieve each of the objectives that were proposed in this paper: the best-performing machine learning model was created by testing a combination of no feature scaling, with feature scaling, resampling, and feature extraction. The final model to be used was Random Forest with resampling, which had an accuracy of 0.981808, an F1 score of 0.986105, a precision of 0.975802, and an AUC of 0.999491. Aside from these metrics, the suitability towards the LIME model is also considered in evaluating the performance of a model, which this model was able to demonstrate.

Lastly, the best-performing model incorporated with XAI in the form of the LIME interpreter, was built into the backend of the Django web application. The web application has 4 pages: the home page, the page where inputs are applied, an about page that also provides instruction for interpretation, and the results page. The web application is hosted on the Heroku platform, with an eco subscription.

VII. Conclusions

This study is about aiding the process of diagnosing breast cancer from values obtained through Fine Needle Aspiration. These inputs obtained consist of 30 real values. 10 different configurations of 4 machine learning models were tested, for a total of 40 machine learning models. The chosen model and configuration is random forest with resampling, which achieved an accuracy of 0.98324, an F1 score of 0.982249, a precision of 0.976471, and an AUC of 0.983521. Aside from that, it was also able to maximize the explainability that the LIME interpreter provides. A web application was built on the Django platform that incorporates this machine learning model with the LIME interpreter, and generates the results that consist of the predicted value, and the outputs from LIME.

VIII. Recommendations

While the machine learning model itself exhibits high metrics and is fully compatible with the LIME interpreter, it is still recommended that other preprocessing techniques be explored, particularly on the application of feature extraction. As of writing, while PCA is not recommended to be applied with the lime interpreter since it diminishes the provided explainability, there are other feature extraction techniques that may be applied such as the selectKbest and RFECV (Recursive Feature Elimination with Cross-Validation). This can possibly retain the model metrics or minimally diminish it, while the necessary features may be reduced significantly, to allow for more ease with regards to the input of values.

Also, it is also recommended that other XAI packages be tested aside from LIME, such as SHAP (SHapley Additive exPlanations). This may allow for better explainability in the context of breast cancer, or it may give entirely different explanations, that may be compared with the output of LIME.

In this study, the Breast Cancer Wisconsin Diagnostic Dataset was used. For future work, it is suggested that other datasets be explored so as to see if the results, particularly the metrics, are consistent with what was demonstrated in tables 1 and 2.

IX. Bibliography

[1] T. Wu and J. Lee, "Promoting breast cancer awareness and screening practices for early detection in low-resource settings," *Pubmed Central*, vol. 15, 2019.

[2] C. H. Barrios, "Global challenges in breast cancer detection and treatment," *The breast*, vol. 62, 2022.

[3] M. A. Naji, S. E. Filali, K. Aarika, E. H. Benlahmar, R. A. Abdelouhadid, and O. Debauche, "Machine learning algorithms for breast cancer prediction and diagnosis," *Procedia computer science*, vol. 191, 2021.

[4] V. P. C. Magboo and M. S. A. Magboo, "Machine learning classifiers on breast cancer recurrences," *Procedia computer science*, vol. 192, 2021.

[5] S. Islam, S. Sarkar, F. I. Ayaz, M. K. Ananda, T. Tazin, A. A. Albraikan, and F. A. Almalki, "Machine learning based comparative analysis for breast cancer prediction," *Journal of healthcare engineering: advanced circuits and systems for healthcare and security applications*, 2022.

[6] F. Walsh, "Ai 'outperforms' doctors diagnosing breast cancer." https://www.bbc.com/news/health-50857759, October 2022.

[7] A. San Juan, A. Salillas, "Is fna still a useful tool in the diagnosis of breast masses? a 5-year review with cytohistopathologic correlation," 2017.

[8] J. S. Abele, "Private practice outpatient fine needle aspiration clinic: A 2018 update," *Cancer cytopathology*, vol. 126, 2018.

[9] G. Gbenosi, M. Boucham, Z. Belrhiti, C. Nejjari, I. Huybrechts, and M. Khalis, "Health system factors that influence diagnostic and treatment intervals in women with breast cancer in sub-saharan africa: A systematic review," *BMC Public Health*, vol. 21, 2021.

[10] R. Confalonieri, L. Coba, B. Wagner, and T. R. Besold, "A historical perspective of explainable artificial intelligence," *WIREs data mining and knowledge discovery*, vol. 11, 2020.

[11] W. H. Wohlberg, W. N. Street, and O. L. Mangasarian, "Machine learning techniques to diagnose breast cancer from image-processed nuclear features of fine needle aspirates," *Cancer letters*, vol. 77, 1994.

[12] V. P. C. Magboo and M. S. A. Magboo, "Explainable ai for autism classification in children," *Agents and multi-agent systems: Technologies and applications*, vol. 306, 2022.

[13] Z. C. Lipton, "The mythos of model interpretability," *Acmqueue*, vol. 16, 2018.

[14] K. Selig, "What is machine learning: A definition." https://www.expert.ai/blog/machine-learning-definition/, November 2022.

[15] M. Schemmer, P. Hemmer, M. Nitsche, N. Kuhl, and M. Vossing, "A meta-analysis of the utility of explainable artificial intelligence in human-ai decision-making," *Proceedings of the 2022 AAAI/ACM Conference on AI, Ethics, and Society*, 2021.

[16] S. Hibbs, "Fine needle aspiration: How to prepare and what to expect." https://www.cancer.net/blog/2021-10/fine-needle-aspiration-biopsy-how-prepare-and-what-expect#:~:text=To20perform20a20fine20needle,fine20needle20biopsy20or20FNA., November 2021.

X. Appendix

A. Source Code

```
{% extends 'predict/homepage.html' %}
{% load static %}
{% block content %}

<div class="container" style="background-color: white; border-radius: 100px;">
    <div class="top container-md about-us-text">Breastfriend is a machine learning web application
         that utilizes the Wisconsin Breast Cancer Dataset, of which details are available
        at the UCI Machine Learning Repository. 4 machine learning models, namely Random Forest,
             Logistic Regression, K-Nearest Neighbors,
        and Support Vector Machine, were tested on different pre-processing techniques, which
             generated 40 different configurations which were compared with one another, to find
             which
        model performed best in terms of its predicting power, and its suitability with the lime
             python package, that is used to generate meaningful explanations.</div>

        <div class="container-md about-us-text">Below are the different components that will be
             encountered while using this web application, and instructions on how they are handled
            or interpreted.</div>



        <div class="container about-us-grid ">
            <div class="row about-us-row">
                <div class="col ">
                1
                </div>
              <div class="col">
                <img  src="{% static 'predict/images/predict.png' %}">
              </div>
              <div class="col ">
                 This figure represents the machine learning model's confidence towards whether a
                      datapoint is benign (blue) or malignant (orange). This is influenced by
                 the feature importance presented in another figure. In this specific example, the
                      model is 96% confident in predicting that the datapoint is malignant.
              </div>
            </div>
            <div class="row about-us-row">
                <div class="col ">
                    2
                </div>
                <div class="col">
                    <img style="height: 2in;" src="{% static 'predict/images/importance.png' %}">
                </div>
                <div class="col">
                    This figure represents the top 10 features that had the most significant
                         impact in terms of predicting whether a datapoint is benign or malignant.
                         In other
                    words, this represents the feature importance of the top 10 features. How is
                         is interpreted is, for the 3rd row in the figure for example, because the
                         input
                    value for concavity worst is greater than 0.18, this attribute alone has a 9%
                         impact towards why the model predicted malignant.
                </div>
            </div>
              <div class="row about-us-row">
                <div class="col ">
                    3
                    </div>
                <div class="col">
                    <img style="height: 2in;" src="{% static 'predict/images/values.png' %}">
                </div>
                <div  class="col">
                    This figure displays the respective input values of the top 10 most
                         significant features, and its color denotes whether this attribute
                         contributed towards the
                    datapoint being benign or malignant.
                </div>
              </div>
              <div class="row about-us-row">
                <div class="col ">
                    4
                    </div>
                <div class="col">
                    <img style="width: 4in;" src="{% static 'predict/images/lime.png' %}">
                </div>
                <div  class="col">
                    This figure represents a more detailed version, where all values and
                         attributes in the y axis is present, of the previous figure on feature
                         importance. In this
                    figure alone, a negative red value denotes benignity, whereas a positive green
                          value denotes malignance.
```

```
                </div>
              </div>
              <div class="row about-us-row">
                <div class="col ">
                    5
                    </div>
                <div class="col">
                    <img src="{% static 'predict/images/diagnosis.png' %}">
                </div>
                <div class="col">
                    This text gives the final predicted class of the model, as well as the real-
                        time accuracy that the model exhibited from the testing set in the machine
                         learning
                    process.
                </div>
              </div>
              <div class="row about-us-row">
                <div class="col ">
                    6
                    </div>
                <div class="col">
                    <img style="width: 4in;"     src="{% static 'predict/images/textexplanation.png
                        ' %}">
                </div>
                <div class="col">
                    This is an explanation, in text, of the attributes, and their significance
                        towards the conclusion that the model had predicted -- if the model
                        predicted that a
                    datapoint is negative for cancer, then it only gives an explanation for the
                        attributes that the model deems to have contributed towards the benignity
                        of a datapoint.
                </div>
              </div>
              <div class="row about-us-row bottom">
                <div class="col ">
                    7
                    </div>
                <div class="col">
                    <img style="width: 4in;" src="{% static 'predict/images/inputs.png' %}">
                </div>
                <div class="col">
                    This is the input page, where 30 of the required inputs, that are values
                        extracted from the digitized image of a breast tissue sample, taken from a
                         FNA test, are applied.
                </div>
              </div>
            </div>

</div>



{% endblock %}

{% load static %}
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>BreastFriend</title>
    <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/css/bootstrap.min.css" rel="
        stylesheet" integrity="sha384-
        rbsA2VBKQhggwzxH7pPCaAqO46MgnOM80zW1RWuH61DGLwZJEdK2Kadq2F9CUG65" crossorigin="anonymous">
    <link rel="stylesheet" href="{% static 'predict/css/website.css' %}">
    <link rel="preconnect" href="https://fonts.googleapis.com">
    <link rel="preconnect" href="https://fonts.gstatic.com" crossorigin>
    <link href="https://fonts.googleapis.com/css2?family=Flow+Circular&family=Jost:ital,wght@0
        ,400;0,500;1,100&display=swap" rel="stylesheet">
    <script src="https://kit.fontawesome.com/767e01c769.js" crossorigin="anonymous"></script>
  </head>
<body background="{% static 'predict/images/61802.jpg' %}">
    <nav class="navbar navbar-expand-lg bg-light">
        <div class="container-fluid">
          <a class="navbar-brand" href="#">BreastFriend</a>
          <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#
              navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle
              navigation">
            <span class="navbar-toggler-icon"></span>
          </button>
          <div class="collapse navbar-collapse" id="navbarNav">
            <ul class="navbar-nav">
              <li class="nav-item">
                <a class="nav-link" href="{%url 'homepage'%}">Home</a>
              </li>
              <li class="nav-item">
                <a class="nav-link" href="{%url 'predict-page'%}">Predict</a>
              </li>
              <li class="nav-item">
```

```
            <a class="nav-link" href="{%url 'about-us'%}">About Us</a>
          </li>
        </ul>
      </div>
    </div>
  </nav>

{%block content%}
<div class="cright">
  <a  href="https://www.freepik.com/free-vector/health-care-science-medical-dna-background-
      banner_5129974.htm">Image by starline</a> on Freepik
</div>
<div class="title">
  <p style="font-size: 60px; text-align: center;">BreastFriend</p>
  <p style="text-align: center; font-size: 25px;"> Predicting breast cancer diagnosis based on
       Fine Needle Aspirate findings. </p>
  <div >
    <a class="nav-link buttons" href="{%url 'about-us'%}"><button  class="btn btn-primary btn-
        lg">Learn More</button></a>
    <a class="nav-link buttons" href="{%url 'predict-page'%}"><button class="btn btn-primary
        btn-lg"> Predict</button></a>
  </div>
  <br>
  <br>
  <br>
  <br>



  <div class="row row-cols-1 row-cols-md-3 g-4" style="margin-bottom: 2in; text-align: center
      ;">
    <div class="col">
      <div class="card h-100 bg-transparent border-dark mb3" >
        <div >
          <img style="width: 3in; text-align: center;" src="{% static 'predict/images/chart.
              png' %}" class="card-img-top" alt="...">
        </div>

        <div class="card-body">
          <h5 class="card-title">Explainability</h5>
          <p class="card-text">Using the Local Interpretable Model-Agnostic Explanations (Lime
              ) Python package, meaningful explanations were extracted
            from predictions made on breast cancer diagnosis.
          </p>
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card h-100 bg-transparent border-dark mb-3">
        <div>
          <img style="width: 3in; text-align: center;" src="{% static 'predict/images/bullseye
              .png' %}" class="card-img-top" alt="...">
        </div>

        <div class="card-body">
          <h5 class="card-title">Reliability</h5>
          <p class="card-text">With a guaranteed accuracy and precision of over 97% towards
              diagnosing breast cancer from Fine Needle Aspiration results.</p>
        </div>
      </div>
    </div>
    <div class="col">
      <div class="card h-100 bg-transparent border-dark mb-3">
        <div>
          <img style="width: 3in; text-align: center;" src="{% static 'predict/images/house.
              png' %}" class="card-img-top" alt="...">
        </div>

        <br>
        <div class="card-body">
          <h5 class="card-title">Simplicity</h5>
          <p class="card-text"> A simple user interface that integrates a reliable machine
              learning model with meaningful explainability, catered towards doctors and
              radiologists.
          </p>
        </div>
      </div>
    </div>
  </div>


</div>



{% endblock %}
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.2.3/dist/js/bootstrap.bundle.min.js"
    integrity="sha384-kenU1KFdBIe4zVF0s0G1M5b4hcpxyD9F7jL+jjXkk+Q2h455rYXK/7HAuoJl+0I4"
    crossorigin="anonymous"></script>
```

```
</body>
</html>


# Importing the necessary libraries
import matplotlib.pyplot as plt
import pandas as pd
import sklearn
# Loading the dataset using sklearn
import numpy as np # linear algebra
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import recall_score,precision_score,accuracy_score,f1_score,roc_auc_score,
    confusion_matrix,roc_curve
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import os


os.chdir(os.path.dirname(os.path.abspath(__file__)))


dataset = pd.read_csv("data.csv")
dataset = pd.get_dummies(data = dataset, drop_first = True)
dataset = dataset.drop(columns = 'Unnamed: 32')
X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:,-1].values

X_scaled = X

X_imbalanced = np.vstack((X_scaled[y == 1], X_scaled[y == 0]))
y_imbalanced = np.hstack((y[y == 1], y[y == 0]))



from sklearn.utils import resample
#
# Create oversampled training data set for minority class
#
X_oversampled, y_oversampled = resample(X_imbalanced[y_imbalanced == 1],
                                        y_imbalanced[y_imbalanced == 1],
                                        replace=True,
                                        n_samples=X_imbalanced[y_imbalanced == 0].shape[0],
                                        random_state=123)
#
# Append the oversampled minority class to training data and related labels
#

X_balanced = np.vstack((X_scaled[y == 0], X_oversampled))
y_balanced = np.hstack((y[y == 0], y_oversampled))


from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y, test_size = 0.25, random_state = 45)

dataframe = pd.DataFrame(X_scaled)

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

acc = accuracy_score(y_test,y_pred)
conf_matrix = confusion_matrix(y_test,y_pred)
fpr,tpr,_ = roc_curve(y_test,y_pred)
plt.plot(fpr,tpr)
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()


#X_train, X_test, y_train, y_test = train_test_split(
#    X, y, train_size=0.90, random_state=50)

# Creating a dataframe of the data, for a visual check
#df = pd.concat([pd.DataFrame(X), pd.DataFrame(y)], axis=1)
#df.columns = np.concatenate((features, np.array(['label'])))
#print("Shape of data =", df.shape)

# Printing the top 5 rows of the dataframe
#print(df.head())

# Instantiating the prediction model - an extra-trees regressor
#from sklearn.ensemble import ExtraTreesRegressor
#reg = ExtraTreesRegressor(random_state=50)

# Fitting the predictino model onto the training set
#reg.fit(X_train, y_train)

# Checking the model's performance on the test set
#print('R2 score for the model on test set =', reg.score(X_test, y_test))
```

```
# Importing the module for LimeTabularExplainer
import lime.lime_tabular

# Instantiating the explainer object by passing in the training set, and the extracted features
explainer_lime = lime.lime_tabular.LimeTabularExplainer(x_train,
                                                        feature_names=dataframe.columns,
                                                        verbose=True, mode='classification')

# Index corresponding to the test vector
i = 25

# Number denoting the top features
k = 10

# Calling the explain_instance method by passing in the:
#    1) ith test vector
#    2) prediction function used by our prediction model('reg' in this case)
#    3) the top features which we want to see, denoted by k
exp_lime = explainer_lime.explain_instance(
    x_test[i], rfc.predict_proba, num_features=k)

plot = exp_lime.as_pyplot_figure()

import io
from PIL import Image
import matplotlib.pyplot as plt

plt.rcParams["figure.figsize"] = [100, 50]
plt.rcParams["figure.autolayout"] = True

plt.figure(plot)

img_buf = io.BytesIO()
plt.savefig(img_buf, format='png',bbox_inches = 'tight')

im = Image.open(img_buf)
im.show(title="My Image")

img_buf.close()



# Finally visualizing the explanations


{% extends 'predict/homepage.html' %}
{% block content %}
 <body class="predict">

  <form action="" method="post">
    {% csrf_token %}
    <center>
      <div class="card">
        <div class="card-body">
          <h3 style="margin-bottom: 10%">Enter required values:</h3>
          <div class="row">
            <div class="col">
              <input
                type="number"
                step="any"
                name="radiusm"
                class="form-control"
                id="floatingInput"
                placeholder="Radius Mean"
                required
              />
            </div>
            <div class="col">
                <input
                  type="number"
                  step="any"
                  name="radiusse"
                  class="form-control"
                  id="floatingPassword"
                  placeholder="Radius Standard Error"
                  required
                />
            </div>
            <div class="col">
                <input
                  type="number"
                  step="any"
                  name="radiusl"
                  class="form-control"
                  id="floatingPassword"
                  placeholder="Radius Largest"
                  required
                />
            </div>
          </div>
```

```
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="texturem"
      class="form-control"
      id="floatingInput"
      placeholder="Texture Mean"
      required
    />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="texturese"
        class="form-control"
        id="floatingPassword"
        placeholder="Texture Standard Error"
        required
      />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="texturel"
        class="form-control"
        id="floatingPassword"
        placeholder="Texture Largest"
        required
      />
  </div>
</div>
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="perimeterm"
      class="form-control"
      id="floatingInput"
      placeholder="Perimeter Mean"
      required
    />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="perimeterse"
        class="form-control"
        id="floatingPassword"
        placeholder="Perimeter Standard Error"
        required
      />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="perimeterl"
        class="form-control"
        id="floatingPassword"
        placeholder="Perimeter Largest"
        required
      />
  </div>
</div>
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="aream"
      class="form-control"
      id="floatingInput"
      placeholder="Area Mean"
      required
    />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="arease"
        class="form-control"
        id="floatingPassword"
        placeholder="Area Standard Error"
        required
      />
```

```
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="areal"
        class="form-control"
        id="floatingPassword"
        placeholder="Area Largest"
        required
      />
  </div>
</div>
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="smoothnessm"
      class="form-control"
      id="floatingInput"
      placeholder="Smoothness Mean"
      required
    />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="smoothnessse"
        class="form-control"
        id="floatingPassword"
        placeholder="Smoothness Standard Error"
        required
      />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="smoothnessl"
        class="form-control"
        id="floatingPassword"
        placeholder="Smoothness Largest"
        required
      />
  </div>
</div>
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="compactnessm"
      class="form-control"
      id="floatingInput"
      placeholder="Compactness Mean"
      required
    />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="compactnessse"
        class="form-control"
        id="floatingPassword"
        placeholder="Compactness Standard Error"
        required
      />
  </div>
  <div class="col">
      <input
        type="number"
        step="any"
        name="compactnessl"
        class="form-control"
        id="floatingPassword"
        placeholder="Compactness Largest"
        required
      />
  </div>
</div>
<div class="row">
  <div class="col">
    <input
      type="number"
      step="any"
      name="concavitym"
      class="form-control"
      id="floatingInput"
      placeholder="Concavity Mean"
```

```
        required
      />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="concavityse"
          class="form-control"
          id="floatingPassword"
          placeholder="Concavity Standard Error"
          required
        />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="concavityl"
          class="form-control"
          id="floatingPassword"
          placeholder="Concavity Largest"
          required
        />
    </div>
  </div>
  <div class="row">
    <div class="col">
      <input
        type="number"
        step="any"
        name="concavepointsm"
        class="form-control"
        id="floatingInput"
        placeholder="Concave Points Mean"
        required
      />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="concavepointsse"
          class="form-control"
          id="floatingPassword"
          placeholder="Concave Points Standard Error"
          required
        />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="concavepointsl"
          class="form-control"
          id="floatingPassword"
          placeholder="Concave Points Largest"
          required
        />
    </div>
  </div>
  <div class="row">
    <div class="col">
      <input
        type="number"
        step="any"
        name="symmetrym"
        class="form-control"
        id="floatingInput"
        placeholder="Symmetry Mean"
        required
      />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="symmetryse"
          class="form-control"
          id="floatingPassword"
          placeholder="Symmetry Standard Error"
          required
        />
    </div>
    <div class="col">
        <input
          type="number"
          step="any"
          name="symmetryl"
          class="form-control"
          id="floatingPassword"
          placeholder="Symmetry Largest"
```

```
                required
              />
            </div>
          </div>
          <div class="row">
            <div class="col">
              <input
                type="number"
                step="any"
                name="fractaldimensionm"
                class="form-control"
                id="floatingInput"
                placeholder="Fractal Dimension Mean"
                required
              />
            </div>
            <div class="col">
                <input
                  type="number"
                  step="any"
                  name="fractaldimensionse"
                  class="form-control"
                  id="floatingPassword"
                  placeholder="Fractal Dimension Standard Error"
                  required
                />
            </div>
            <div class="col">
                <input
                  type="number"
                  step="any"
                  name="fractaldimensionl"
                  class="form-control"
                  id="floatingPassword"
                  placeholder="Fractal Dimension Largest"
                  required
                />
            </div>
          </div>
          <div class="col">
            <button class="btn btn-custom" type="submit">Predict!</button>
          </div>
        </div>
      </div>
    </center>
  </form>
</body>
{% endblock %}


{% extends 'predict/homepage.html' %}
{% block content %}



<center>
{% autoescape off %}
{{results}}
{% endautoescape %}

<div>
    <img src="{{image64}}" width="800" height="400">
    <table>
        <tr>
            <td>
                Prediction:
            </td>
            <td>
                {{pred}}
            </td>
        </tr>
        <tr>
            <td>
                Accuracy:
            </td>
            <td>
                {{accuracy}}
            </td>
        </tr>
    </table>
    </div>

    <div>
    <table>
        {% for key, value in feat_imp.items %}
        <tr>
            <td>
                {% if pred == 'Negative!' %}
                    {% if value < 0 %}
                    The attribute {{key}} contributed {% widthratio value 1 -1 %}% towards the
                        sample tissue being <p style="font-weight: bold;">Benign</p>
                    {% endif %}
```

```
                {% else %}
                    {% if value >= 0 %}
                    The attribute {{key}} contributed {{value}}% towards the sample tissue being <
                        p style="font-weight: bold;">Malignant</p>
                    {% endif %}
                {% endif %}
            </td>
        </tr>
        {% endfor %}
    </table>
    </div>
</center>
```

```
{% endblock %}
```

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


dataset = pd.read_csv("data.csv")
dataset = pd.get_dummies(data = dataset, drop_first = True)
dataset = dataset.drop(columns = 'Unnamed: 32')
X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:,-1].values

X_scaled = X

X_imbalanced = np.vstack((X_scaled[y == 1], X_scaled[y == 0]))
y_imbalanced = np.hstack((y[y == 1], y[y == 0]))



from sklearn.utils import resample

# Create oversampled training data set for minority class

X_oversampled, y_oversampled = resample(X_imbalanced[y_imbalanced == 1],
                                        y_imbalanced[y_imbalanced == 1],
                                        replace=True,
                                        n_samples=X_imbalanced[y_imbalanced == 0].shape[0],
                                        random_state=123)

# Append the oversampled minority class to training data and related labels


X_balanced = np.vstack((X_scaled[y == 0], X_oversampled))
y_balanced = np.hstack((y[y == 0], y_oversampled))

print(len(X_balanced))

from sklearn.model_selection import train_test_split,cross_val_score
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y, test_size = 0.25, random_state = 45)
x_train_b,x_test_b,y_train_b,y_test_b = train_test_split(X_scaled,y, test_size = 0.25,
    random_state = 45)



from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)
from sklearn.metrics import recall_score,precision_score,accuracy_score,f1_score,roc_auc_score

# Calculate accuracy using cross-validation
accuracy_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
```

```
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

##feature scaling then resampling
print("Feature Scaling Then resampling")
x_train,x_test,y_train,y_test = train_test_split(X_balanced,y_balanced, test_size = 0.25,
    random_state = 45)


rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)
```

```
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

##feature scaling, resampling, then feature extraction
print("Feature scaling, resampling, then feature extraction")
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=1)
X_train = pca.fit_transform(x_train)
X_test = pca.transform(x_test)

classifier = RandomForestClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
```

```
accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

print("Feature scaling, then feature extraction")
pca = PCA(n_components=1)
X_train = pca.fit_transform(x_train_b)
X_test = pca.transform(x_test_b)


classifier = RandomForestClassifier()
classifier.fit(X_train, y_train_b)
y_pred = classifier.predict(X_test)

accuracy_scores = cross_val_score(rfc, X, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train_b)
y_pred = knn.predict(X_test)

accuracy_scores = cross_val_score(knn, X, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
```

```
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train_b)
y_pred = clf.predict(X_test)

accuracy_scores = cross_val_score(clf, X, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


lr = LogisticRegression(random_state = 0)
lr.fit(X_train, y_train_b)
y_pred = lr.predict(X_test)

accuracy_scores = cross_val_score(lr, X, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
#Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X, y, cv=10, scoring='precision')
precision = precision_scores.mean()
#Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
#Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

print("Feature scaling, feature extraction, then resampling")



merged_x = np.concatenate((X_train, X_test), axis=0)


X_imbalanced = np.vstack((merged_x[y == 1], merged_x[y == 0]))
y_imbalanced = np.hstack((y[y == 1], y[y == 0]))
from sklearn.utils import resample
#
# Create oversampled training data set for minority class
#
X_oversampled, y_oversampled = resample(X_imbalanced[y_imbalanced == 1],
                                        y_imbalanced[y_imbalanced == 1],
                                        replace=True,
                                        n_samples=X_imbalanced[y_imbalanced == 0].shape[0],
                                        random_state=123)
#
# Append the oversampled minority class to training data and related labels
#
X_balanced = np.vstack((merged_x[y == 0], X_oversampled))
y_balanced = np.hstack((y[y == 0], y_oversampled))



x_train,x_test,y_train,y_test = train_test_split(X_balanced,y_balanced, test_size = 0.25,
     random_state = 45)

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
```

```
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)


dataset = pd.read_csv("data.csv")
dataset = pd.get_dummies(data = dataset, drop_first = True)
dataset = dataset.drop(columns = 'Unnamed: 32')
X = dataset.iloc[:,1:-1].values
y = dataset.iloc[:,-1].values


##feature scaling
print("Feature Scaling")
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_scaled = sc.fit_transform(X)

X_imbalanced = np.vstack((X_scaled[y == 1], X_scaled[y == 0]))
y_imbalanced = np.hstack((y[y == 1], y[y == 0]))


from sklearn.utils import resample
#
# Create oversampled training data set for minority class
#
X_oversampled, y_oversampled = resample(X_imbalanced[y_imbalanced == 1],
```

```
                                        y_imbalanced[y_imbalanced == 1],
                                        replace=True,
                                        n_samples=X_imbalanced[y_imbalanced == 0].shape[0],
                                        random_state=123)
#
# Append the oversampled minority class to training data and related labels
#

X_balanced = np.vstack((X_scaled[y == 0], X_oversampled))
y_balanced = np.hstack((y[y == 0], y_oversampled))

print(len(X_balanced))

from sklearn.model_selection import train_test_split,cross_val_score
x_train,x_test,y_train,y_test = train_test_split(X_scaled,y, test_size = 0.25, random_state = 45)
x_train_b,x_test_b,y_train_b,y_test_b = train_test_split(X_scaled,y, test_size = 0.25,
     random_state = 45)



from sklearn.ensemble import RandomForestClassifier
rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)
from sklearn.metrics import recall_score,precision_score,accuracy_score,f1_score,roc_auc_score

# Calculate accuracy using cross-validation
accuracy_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.neighbors import KNeighborsClassifier
knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.linear_model import LogisticRegression

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
```

```
precision_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

##feature scaling then resampling
print("Feature Scaling Then resampling")
x_train,x_test,y_train,y_test = train_test_split(X_balanced,y_balanced, test_size = 0.25,
     random_state = 45)


rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
```

```
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

##feature scaling, resampling, then feature extraction
print("Feature scaling, resampling, then feature extraction")
from sklearn.decomposition import PCA
import matplotlib.pyplot as plt

pca = PCA(n_components=1)
X_train = pca.fit_transform(x_train)
X_test = pca.transform(x_test)

classifier = RandomForestClassifier()
classifier.fit(X_train, y_train)
y_pred = classifier.predict(X_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)

accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors------', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

print("Feature scaling, then feature extraction")
```

```
pca = PCA(n_components=1)
X_train = pca.fit_transform(x_train_b)
X_test = pca.transform(x_test_b)


classifier = RandomForestClassifier()
classifier.fit(X_train, y_train_b)
y_pred = classifier.predict(X_test)

accuracy_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(X_train, y_train_b)
y_pred = knn.predict(X_test)

accuracy_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors------', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(X_train, y_train_b)
y_pred = clf.predict(X_test)

accuracy_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


lr = LogisticRegression(random_state = 0)
lr.fit(X_train, y_train_b)
y_pred = lr.predict(X_test)

accuracy_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_scaled, y, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)

print("Feature scaling, feature extraction, then resampling")


merged_x = np.concatenate((X_train, X_test), axis=0)


X_imbalanced = np.vstack((merged_x[y == 1], merged_x[y == 0]))
```

```
y_imbalanced = np.hstack((y[y == 1], y[y == 0]))
from sklearn.utils import resample
#
# Create oversampled training data set for minority class
#
X_oversampled, y_oversampled = resample(X_imbalanced[y_imbalanced == 1],
                                        y_imbalanced[y_imbalanced == 1],
                                        replace=True,
                                        n_samples=X_imbalanced[y_imbalanced == 0].shape[0],
                                        random_state=123)
#
# Append the oversampled minority class to training data and related labels
#
X_balanced = np.vstack((merged_x[y == 0], X_oversampled))
y_balanced = np.hstack((y[y == 0], y_oversampled))



x_train,x_test,y_train,y_test = train_test_split(X_balanced,y_balanced, test_size = 0.25,
    random_state = 45)

rfc = RandomForestClassifier()
rfc.fit(x_train,y_train)
y_pred = rfc.predict(x_test)

accuracy_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(rfc, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['RandomForestClassifier', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


knn = KNeighborsClassifier(n_neighbors = 3)
knn.fit(x_train, y_train)
y_pred = knn.predict(x_test)

accuracy_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(knn, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['KNearestNeighbors-----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from sklearn.svm import SVC
clf = SVC(kernel='linear')
clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

accuracy_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
f1_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(clf, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['SupportVectorMachines-', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


lr = LogisticRegression(random_state = 0)
lr.fit(x_train, y_train)
y_pred = lr.predict(x_test)

accuracy_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='accuracy')
acc = accuracy_scores.mean()
# Calculate precision using cross-validation
precision_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='precision')
precision = precision_scores.mean()
# Calculate F1 score using cross-validation
```

```
f1_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='f1')
f1 = f1_scores.mean()
# Calculate AUC using cross-validation
auc_scores = cross_val_score(lr, X_balanced, y_balanced, cv=10, scoring='roc_auc')
auc = auc_scores.mean()
results = pd.DataFrame([['LogisticRegression----', acc, f1, precision, auc]],
                       columns = ['Model','Accuracy', 'F1', ' Precision', 'AUC'])
print(results)


from django.urls import path
from . import views

urlpatterns = [
    path('', views.homepage, name="homepage"),
    path('predict/', views.predict_page, name="predict-page"),
    path('about-us/', views.about_us, name="about-us"),
]


from django.shortcuts import render
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import precision_score,accuracy_score,f1_score,roc_auc_score
import lime.lime_tabular
import base64
from io import BytesIO
import random
import joblib


def image_to_base64(image):
    buff = BytesIO()
    image.save(buff, format="PNG")
    img_str = base64.b64encode(buff.getvalue())
    img_str = img_str.decode("utf-8")  # convert to str and cut b'' chars
    buff.close()
    return img_str

def Convert(tup, di):
    for a, b in tup:
        a = int(a)
        di.setdefault(a,[]).append(b)
    return di


def rename_features(feat_imp):
    for key in list(feat_imp):
        if key == 'radius_mean':
            feat_imp['mean of radius'] = feat_imp['radius_mean']
            del feat_imp['radius_mean']

        elif key == 'texture_mean':
            feat_imp['mean of texture'] = feat_imp['texture_mean']
            del feat_imp['texture_mean']

        elif key == 'perimeter_mean':
            feat_imp['mean of perimeter'] = feat_imp['perimeter_mean']
            del feat_imp['perimeter_mean']

        elif key == 'area_mean':
            feat_imp['mean of area'] = feat_imp['area_mean']
            del feat_imp['area_mean']

        elif key == 'smoothness_mean':
            feat_imp['mean of smoothness'] = feat_imp['smoothness_mean']
            del feat_imp['smoothness_mean']

        elif key == 'compactness_mean':
            feat_imp['mean of compactness'] = feat_imp['compactness_mean']
            del feat_imp['compactness_mean']

        elif key == 'concavity_mean':
            feat_imp['mean of concavity'] = feat_imp['concavity_mean']
            del feat_imp['concavity_mean']

        elif key == 'concave_mean':
            feat_imp['mean of concave'] = feat_imp['concave_mean']
            del feat_imp['concave_mean']

        elif key == 'symmetry_mean':
            feat_imp['mean of symmetry'] = feat_imp['symmetry_mean']
            del feat_imp['symmetry_mean']

        elif key == 'fractal_dimension_mean':
            feat_imp['mean of fractal dimension'] = feat_imp['fractal_dimension_mean']
            del feat_imp['fractal_dimension_mean']

        elif key == 'radius_se':
            feat_imp['standard error of radius'] = feat_imp['radius_se']
            del feat_imp['radius_se']
```

```
        elif key == 'texture_se':
            feat_imp['standard error of radius'] = feat_imp['radius_se']
            del feat_imp['radius_se']

        elif key == 'perimeter_se':
            feat_imp['standard error of perimeter'] = feat_imp['perimeter_se']
            del feat_imp['perimeter_se']

        elif key == 'area_se':
            feat_imp['standard error of area'] = feat_imp['area_se']
            del feat_imp['area_se']

        elif key == 'smoothness_se':
            feat_imp['standard error of smoothness'] = feat_imp['smoothness_se']
            del feat_imp['smoothness_se']

        elif key == 'compactness_se':
            feat_imp['standard error of compactness'] = feat_imp['compactness_se']
            del feat_imp['compactness_se']

        elif key == 'concavity_se':
            feat_imp['standard error of concavity'] = feat_imp['concavity_se']
            del feat_imp['concavity_se']

        elif key == 'concave_se':
            feat_imp['standard error of concave'] = feat_imp['concave_se']
            del feat_imp['concave_se']

        elif key == 'symmetry_se':
            feat_imp['standard error of symmetry'] = feat_imp['symmetry_se']
            del feat_imp['symmetry_se']

        elif key == 'fractal_dimension_se':
            feat_imp['standard error of fractal dimension'] = feat_imp['fractal_dimension_se']
            del feat_imp['fractal_dimension_se']

        elif key == 'radius_worst':
            feat_imp['largest of radius'] = feat_imp['radius_worst']
            del feat_imp['radius_worst']

        elif key == 'texture_worst':
            feat_imp['largest of texture'] = feat_imp['texture_worst']
            del feat_imp['texture_worst']

        elif key == 'perimeter_worst':
            feat_imp['largest of perimeter'] = feat_imp['perimeter_worst']
            del feat_imp['perimeter_worst']

        elif key == 'area_worst':
            feat_imp['largest of area'] = feat_imp['area_worst']
            del feat_imp['area_worst']

        elif key == 'smoothness_worst':
            feat_imp['largest of smoothness'] = feat_imp['smoothness_worst']
            del feat_imp['smoothness_worst']

        elif key == 'compactness_worst':
            feat_imp['largest of compactness'] = feat_imp['compactness_worst']
            del feat_imp['compactness_worst']

        elif key == 'concavity_worst':
            feat_imp['largest of concavity'] = feat_imp['concavity_worst']
            del feat_imp['concavity_worst']

        elif key == 'concave_worst':
            feat_imp['largest of concave'] = feat_imp['concave_worst']
            del feat_imp['concave_worst']

        elif key == 'symmetry_worst':
            feat_imp['largest of symmetry'] = feat_imp['symmetry_worst']
            del feat_imp['symmetry_worst']

        elif key == 'fractal_dimension_worst':
            feat_imp['largest of fractal dimension'] = feat_imp['fractal_dimension_worst']
            del feat_imp['fractal_dimension_worst']


    return feat_imp


# Create your views here.
def homepage(request):
    from django.core.cache import cache
    from django.contrib.sessions.backends.db import SessionStore
    cache.clear()
    random.seed(123)
    np.random.seed(123)
    session = SessionStore(session_key=request.session.session_key)
    session.flush()
```

```
    return render(request, 'predict/homepage.html')

def predict_page(request):
    if(request.method == 'POST'):
        datapoint = np.array([request.POST.get('radiusm'),request.POST.get('texturem'),request.
            POST.get('perimeterm'),request.POST.get('aream'),request.POST.get('smoothnessm'),
            request.POST.get('compactnessm'),request.POST.get('concavitym'),request.POST.get('
            concavepointsm'),request.POST.get('symmetrym'),request.POST.get('fractaldimensionm'),
                                request.POST.get('radiusse'),request.POST.get('texturese'),request.
                                    POST.get('perimeterse'),request.POST.get('arease'),request.POST.
                                    get('smoothnessse'),request.POST.get('compactnessse'),request.
                                    POST.get('concavityse'),request.POST.get('concavepointsse'),
                                    request.POST.get('symmetryse'),request.POST.get('
                                    fractaldimensionse'),
                                request.POST.get('radiusl'),request.POST.get('texturel'),request.
                                    POST.get('perimeterl'),request.POST.get('areal'),request.POST.
                                    get('smoothnessl'),request.POST.get('compactnessl'),request.POST
                                    .get('concavityl'),request.POST.get('concavepointsl'),request.
                                    POST.get('symmetryl'),request.POST.get('fractaldimensionl'),])
        dataset = pd.read_csv("predict/data/data.csv")
        dataset = pd.get_dummies(data = dataset, drop_first = True)
        dataset = dataset.drop(columns = 'Unnamed: 32')

        rfc = joblib.load("predict/data/random_forest.joblib")

        import dill
        with open('predict/data/explainer.pkl', 'rb') as f:
            explainer_lime = dill.load(f)

        ttts = datapoint.astype(float)
        dataset = dataset.drop(['id','diagnosis_M'],axis=1)
        newerdf=pd.DataFrame(ttts.reshape(1,-1), columns=dataset.columns)

        pred = rfc.predict(newerdf)
        accuracy = '98.18%'

        exp_lime = explainer_lime.explain_instance(
            ttts, rfc.predict_proba, num_features=10)

        # Save the explainer to a file

        lime_model = exp_lime.local_exp
        dictionary = {}
        for _,value in lime_model.items():
            dictionary = Convert(value,dictionary)

        columns = list(dataset.columns)

        feat_imp =  {}
        for i in range(0,len(columns)):
            for key in dictionary:
                if i == key:
                    value = round(dictionary[i][0]*100)
                    feat_imp.update({columns[i-1]:value})

        results = exp_lime.as_html(labels=None, predict_proba=True, show_predicted_value=True)

        plot = exp_lime.as_pyplot_figure()




        import io
        from PIL import Image
        import matplotlib.pyplot as plt
        from django.core.cache import cache
        from django.http import HttpResponse

        plt.figure(plot)
        plt.xlabel('', fontsize=18)
        plt.ylabel('', fontsize=16)
        img_buf = io.BytesIO()
        plt.savefig(img_buf, format='png',bbox_inches = 'tight')
        im = Image.open(img_buf)



        image64 = image_to_base64(im)



        response = HttpResponse(content_type='image/png')
        response.write(img_buf.getvalue())

        if(pred[0]==0):
            pred = "Benign"
        else:
            pred = "Malignant"
```

```
        feat_imp = rename_features(feat_imp)


        data = {'accuracy':accuracy,'results':results,'image64':image64,'pred':pred,'feat_imp':
            feat_imp,'response':response}

        cache.clear()
        return render(request, 'predict/results.html', data)


    return render(request, 'predict/predict-page.html')


def about_us(request):
    return render(request, 'predict/about-us.html')
```

```
body{
    background-size: 100%;
}

.cright{
    position: fixed;
    right: 0px; bottom: 0px;
}

.title {
    position: absolute;
    top: 10%;
    font-family: 'Flow Circular', cursive;
    font-family: 'Jost', sans-serif;
    margin-top: 1in;
}

.about-us-text{
    margin-top: 1in;
    text-align: justify;
}

.about-us-grid{
    top:30%;
    text-align: center;

}



.buttons{
    text-align: center;
    margin-top: 0.5in;
}

.middle{
    padding-left: 0.5in;
    padding-right: 0.5in;
}

.card{
    border: none;
}

.predict{
    background: none;
}

.about-us-row{
    margin-top: 0.5in;
}

.top{
    padding-top: 1in;
}

.bottom{
    padding-bottom: 1in;
}
```

XI. Acknowledgment

First and foremost, God. I would be nothing without you.

To my family, who continues to provide.

To my boys at Teampura who kept me grounded.

To my professors, especially to my Adviser who guided me through.

To the people who put dirt on me that turned to the soil from which I grew out of.