FINAL PROJECT REPORT – CS 545 Machine Learning – Winter 2024

HEART DISEASE PREDICTION

Team Members:

Sriram Nurani Subramanyam

Mahesh Vankayalapati

Team contribution:

For this project, we worked in different phases, finding an appropriate dataset, data preprocessing, and model implementation. We divided this work equally amongst us and did the work as follows:
Data preprocessing and Cleaning: **Mahesh**
Data Normalisation and Handling Imbalance Data: **Sriram**
Algorithm proposal: **Mahesh and Sriram**
Algorithm Implementation: **Mahesh and Sriram**
Report Writeup: **Mahesh and Sriram**

Introduction:

Heart disease is one of the leading causes of death and disability worldwide, Therefore, it is crucial to develop effective methods for early detection and prevention of heart disease. Machine learning is a powerful tool that can help achieve these goals by leveraging the large and complex data sets available in the medical domain.

This project's aim is to apply a machine learning algorithm using various algorithms like 'KNN', 'Logistic Regression', and 'XGBoost' algorithms to the task of heart disease prediction based on demographic, clinical, and behavioral features.

- KNN: KNN stands for K-Nearest-neighbors, It is a method for supervised machine learning that is applied to regression and classification problems. It categorizes new data points according to the majority class of their k closest neighbors in the feature space, using the proximity principle of operation.
- Logistic Regression: A statistical technique called logistic regression is used to predict the probability that an observation will fall into a certain class in a binary classification. In contrast to linear regression, logistic regression is appropriate for binary outcomes as it uses the logistic function (sigmoid) to convert the output into a probability distribution.
- XGBoost: XGBoost stands for eXtreme Gradient Boosting, a type of gradient boosting method that can handle large-scale and complex data sets. It is an ensemble learning method that builds a series of decision trees and combines their predictions. It also provides important feature scores, which can be used to understand the contribution of each feature to the model's predictions.

Objective:

The main objectives of this project are:

Dataset used:

https://www.kaggle.com/datasets/aasheesh200/framingham-heart-study-dataset

- To perform data preprocessing and feature engineering on the Framingham Heart Study data set, which contains information about 4,240 participants who were followed for 10 years to assess their risk of developing coronary heart disease.
- To train, tune, and evaluate the model for binary classification of heart disease using various performance metrics, such as accuracy, precision, and recall score.
- To compare all three models, XGBoost, logistic regression, and KNN, and to analyze the strengths and weaknesses of each model.

Data Pre-processing :

Feature Selection: In our dataset "Education" attribute is not relevant hence removed it from our dataset and the remaining attributes became our final dataset for analysis.

Checking for Null values and cleaning data: The initial dataset was imbalanced and contained non-preprocessed data, almost 880 records were missing. After data pre-processing all the missing values were replaced with the mean of the corresponding featured column.

Data Normalization: We got our data ready for training by using a tool called MinMaxScaler. This tool makes our data fit into a scale from 0 to 1, helping us work with it better.

Handling Imbalance Data: The original dataset had a significant 5.58:1 imbalance, causing problems for accuracy measurements. To tackle the issue of imbalanced data, undersampling was used. This means reducing the number of instances from the overrepresented class to create a balanced 50:50 ratio, specifically for fraudulent transactions. This helps prevent bias in favor of the majority class in algorithms, ensuring a more accurate performance evaluation.

Implementation of Algorithms:

1. KNN Algorithm

Implementation steps:

- The model was initialized within the K Nearest Neighbour classifier by specifying the number of neighbors (K) to be taken into account while making predictions.
- Calculating the distance between this new, unseen data point and all other records within the training model,
- Once the distances and similarities were calculated, the model sorted the instances from shortest to longest distances by placing them in increasing order of proximity to one another.
- From this sorted collection, the model first picked K entries, and it then obtained the labels that went with each of these selected entries.
- From the given K we choose the best K and its accuracy score as the result.

Results:

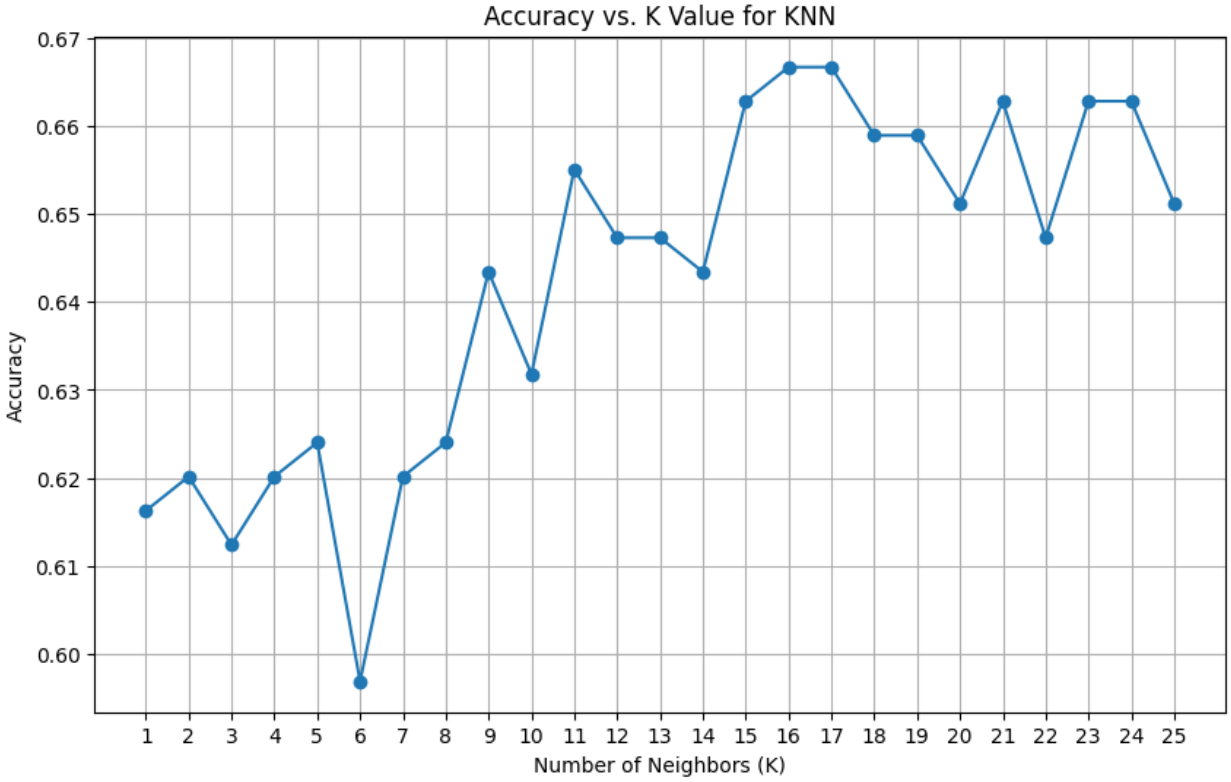
When we first applied the KNN algorithm, it showed a very high accuracy of 85%. However, a closer look at the confusion matrix exposed that the high accuracy was misleading. The model mainly predicts the majority class. Further analysis revealed that the problem was occurring because of having an imbalanced dataset. After addressing the imbalance in the data and ensuring both classes were equal, we achieved the highest accuracy of **66.67%.** This indicates accurate predictions of heart disease without being inclined toward mostly predicting the majority class.

The best accuracy of the model obtained is **66.67%** with a precision: **74%** and recall **65%**.

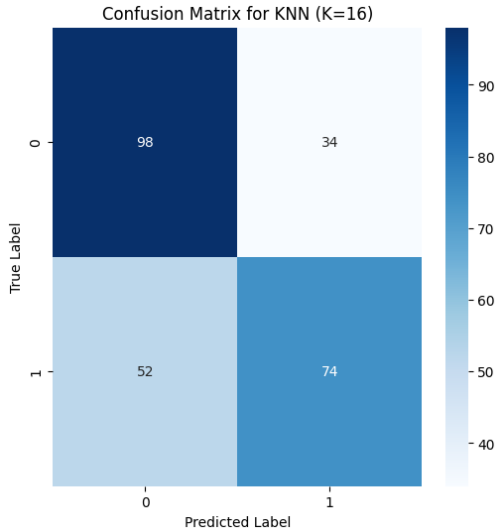
Output:

```
Best K for KNN: 16
KNN Accuracy: 66.67%
Confusion Matrix:
[[98 34]
 [52 74]]
Precision: 0.74
Recall: 0.65
Accuracy: 66.67
```

Choosing the best K value based on the accuracy rate: Accuracy is calculated for each K and added to the list. A graph is then plotted using this list to find the best K value that makes the accuracy more consistent.



From the confusion matrix, it will give the classification report with an accuracy of 66.67%. But this prediction has been done for K=15. Therefore, we need to select the best K value as 16.



2. Logistic Regression:

Model Evaluation:

Using the Scikit-Learn module in Python, we constructed a logistic regression model. The program looks at characteristics like age, gender, BMI, heart rate, glucose etc., while the algorithm runs through the dataset of cardiovascular study on residents of the town of Framingham, Massachusetts.. The chance of a person getting a heart attack is calculated by the model after analyzing the correlations among these attributes. The model can be trained on the given dataset to acquire knowledge about different elements of heart disease prediction. Binary classification issues (Yes/No, True/False, 0/1) are a good fit for logistic regression. The dependent variable's log odds and the independent variables are assumed to have a linear relationship.

Implementation steps:

The method shown is a binary classification implementation of logistic regression using Python and NumPy. The comprehensive instructions and explanations for every part are listed below.

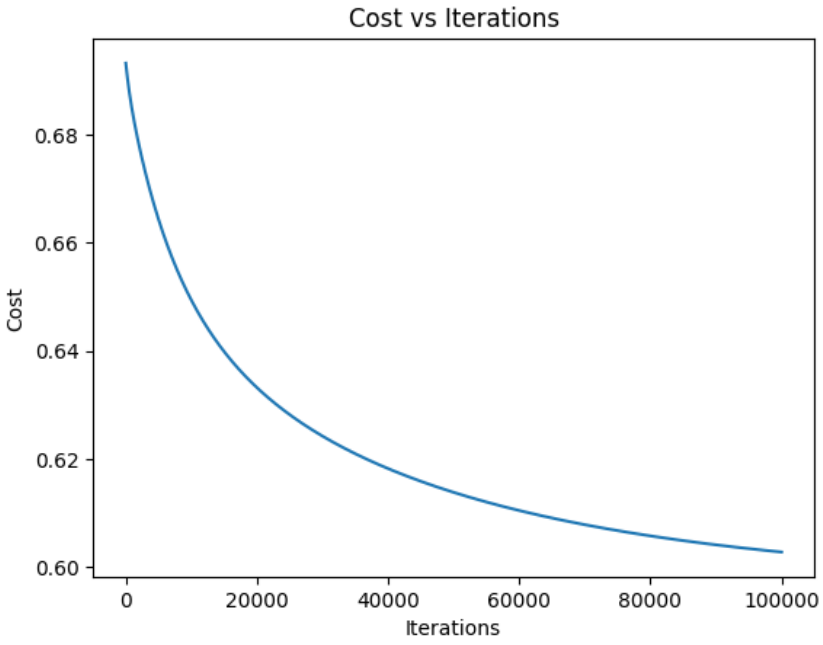
- The learning rate is defined and the model is trained for a predetermined number of iterations. An evaluation of the model's performance is done by calculating the logistic regression cost function.
- Computes predictions using the sigmoid function based on the learned weights and errors.
- The optimization procedure (typically gradient descent) is used by the model to determine the coefficients (weights) for each feature to minimize a cost function (commonly log loss or cross-entropy).
- The convergence of the model is evaluated by plotting the value of the cost function against the number of iterations.
- After training, the model predicts the probability of an instance belonging to the positive class (1) or negative class (0) using learning coefficients and input features.
- Prints the cost following 10% of the iterations. Provides the bias, weights, and cost lists that were learned during training.
- To assess the prediction performance of the model, the accuracy is finally calculated on the test set, by comparing the predictions to the actual values.

Results:

The accuracy rate of the model obtained is 65.5%

```
Accuracy of the model is :  65.5 %
```

Convergence of the model:



Here we can see that the cost function decreases with every iteration and almost gets flattened as we move towards 100k.

3. XGBoost

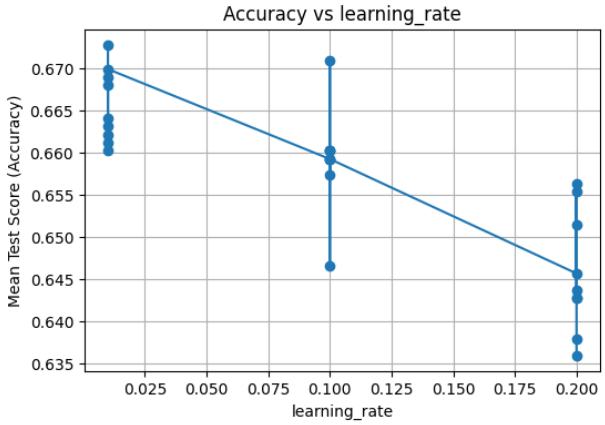
Experimental Setup:

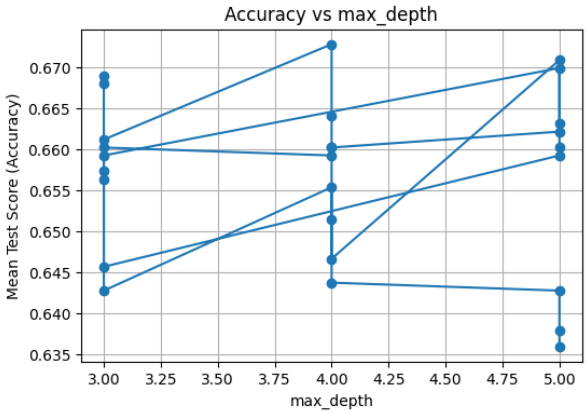
The algorithm implements the XGB algorithm using the 'cross-validation' technique on the training data to calculate the mean accuracy of the model. Three hyperparameters 'learning_rate', 'max_depth', and 'n_estimators' were used and a 'GridSearch' function was performed on the hyperparameters tuning to find the best model combination for prediction. The XGboost has the feature of evaluating the importance of the attributes and it interprets the coefficients of the model to represent the odds of each feature to predict heart disease. Though it is not a linear model it assumes it is linear to find the percentage contribution of each attribute for the CHD. The accuracy, precision-recall, and confusion matrix values are determined after using the model.

Results:

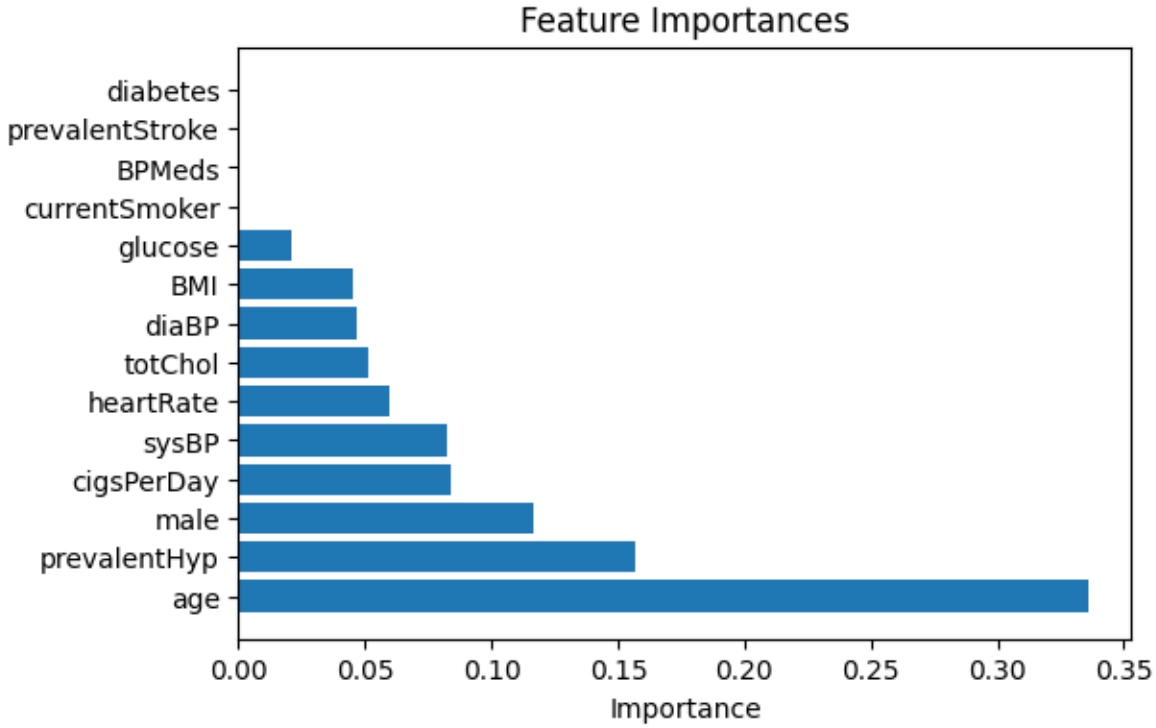
The accuracy rate of the model obtained is **64%,** precision of **61%,** and recall of **70%**.

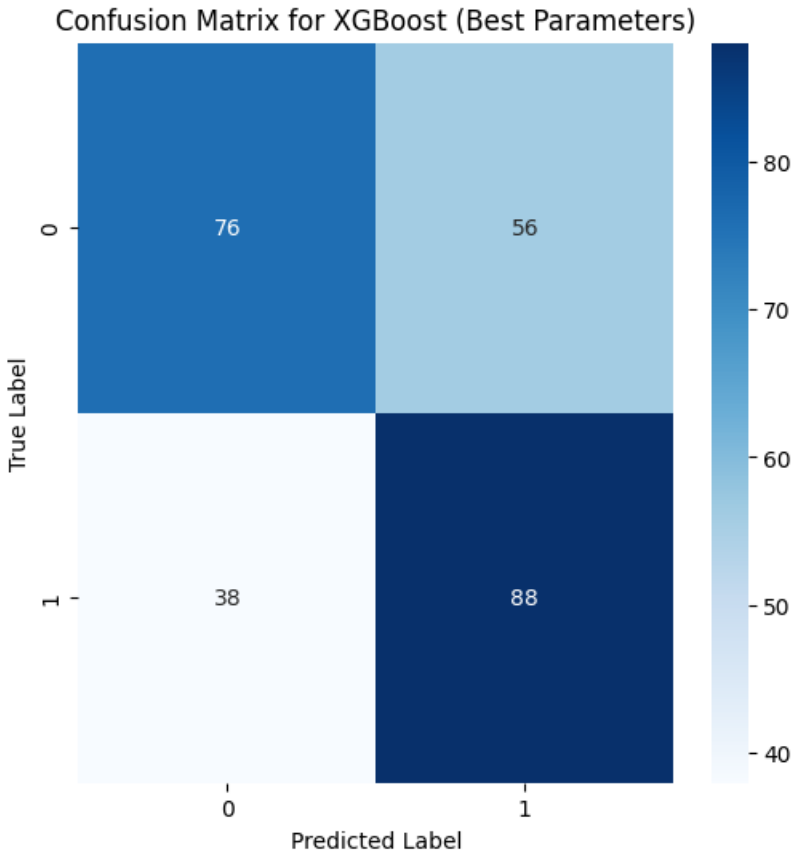
The model has the best hyperparameters of learning_rate = 0.01, max_depth = 4, and n_estimators = 50.





The model has the feature importances that shows the relative importance of each feature for the prediction. The confusion matrix is derived to show that the true negative value is higher than the false prediction and similarly for class 1 true positive is higher than the false positive.





Conclusions after Comparing 3 Algorithms:

- KNN accuracy obtained in this experiment is 66.67 which is slightly better than the accuracy rate of the remaining algorithms as KNN classifiers are faster to train and as there is no training required. Overall all three algorithms' performance accuracy is almost similar.
- KNN is slow with a very large dimensional dataset. Whereas Logistic Regression model works well on high-dimensional datasets.
- When there is a roughly linear connection between the characteristics and the target, Logistic Regression provides a fair compromise between speed and performance.

Source Code :

https://colab.research.google.com/drive/143v91iE1Ez2Wsac9LaiO7D23KjlVCJDn?usp=sharing

References:

https://scikit-learn.org/stable/

https://en.wikipedia.org/wiki/XGBoost

https://en.wikipedia.org/wiki/Logistic_regression

https://realpython.com/logistic-regression-python/

https://www.geeksforgeeks.org/k-nearest-neighbours/