Heart Disease Risk Assessment Model: Analysis of Seven Algorithms

Xiaofeng Chen
UBIT: xchen326
UBID: 50560458

Li Lu
UBIT: lli93
UBID: 50560532

Xueyi Yang
UBIT: xueyiyan
UBID: 50565976

I. Introduction

Heart disease remains the leading cause of mortality globally. However, many people miss the best opportunity for preventive treatment due to neglect of early symptoms of this disease. Based on this common phenomenon, this project aims to address this critical issue by leveraging reliable data sources (we chose the dataset from a general health survey conducted in 2022, as mentioned in the Phase 1 report, rather than specialized cardiac examination data), to establish a model for people's self-assessment of their own risk of heart disease without requiring a hospital visit.

A. Summary of Phase 1 Achievements

In Phase 1, we focused on data preprocessing and exploratory data analysis (EDA) to understand the characteristics and patterns within the dataset. This involved tasks such as data cleaning, handling missing values, and visualizing various features to gain insights into the underlying trends and distributions of our dataset, laying a solid foundation for subsequent phases.

B. Introduction to Phase 2: Algorithmic Modeling

Building upon the insights gained from Phase 1, Phase 2 aims to apply machine learning and statistical modeling algorithms to develop a classification model for heart disease risk assessment. In this phase, we will implement and evaluate algorithms to determine their effectiveness in classifying heart disease risk into "high risk" and "low risk" based on the dataset. With a specific emphasis on algorithmic selection and coding, we endeavor to apply seven different algorithms, each tailored to address the complexities of our problem domain.

II. Seven Algorithms and Coding Implementation

A. Algorithm Selection and Justification

For this phase, we choose seven algorithms to conduct classification on the dataset, including three algorithms from the lecture-Logistic Regression, K-NN, Naive Bayes, and four algorithms out of curriculum-Decision Tree [1], Random Forest [2], XGBoost [3], and Multilayer perceptron(MLP) [4]. Below we will elaborate the reasons for our selection.

a) Logistic Regression: Logistic regression is a classical algorithm for classification problems. By modeling binary outcomes using a logistic function, it offers a straightforward approach to deal with problems involving linearly separable or approximately linearly separable data.

b) K-NN: K-NN is a simple and intuitive algorithm that classifies samples based on the distances between input and samples in the training set. By capturing local patterns and exhibiting good performance in showing irregular decision boundaries, K-NN provides a straightforward approach to classification that can effectively handle non-linear relationships within the data.

c) Naive Bayes: Assuming feature conditional independence and applying Bayes' theorem, Naive Bayes is chosen for its computational efficiency and ability to adapt well to small sample data, making it suitable for handling high-dimensional data and large-scale datasets.

d) Decision Tree: Decision tree is a classification algorithm based on tree-like structures that make decisions by following a series of simple decision rules. This algorithm is selected for its ease of interpretation and understanding, making it ideal for providing transparent decision-making processes that are easily interpretable by users.

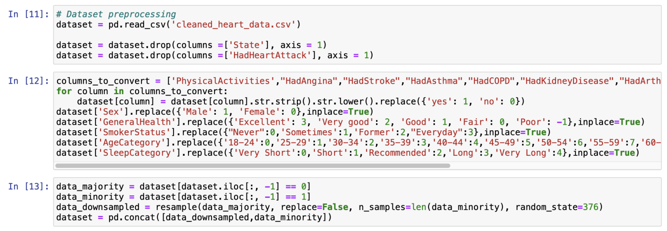
e) Random Forest: Random Forest combines multiple decision trees through voting or averaging. The choice of this algorithm stems from its ability to reduce overfitting and while handling large numbers of features and high-dimensional data effectively.

f) XGBoost: XGBoost integrates multiple weak learners through gradient boosting. It is selected for its ability to handle complex data in large-scale datasets, and to enhance classification performance by iteratively optimizing a loss function, providing robust and scalable solutions across diverse domains.

g) MLP: MLP is chosen for its capacity to learn high-level features and abstract representations from data. By leveraging multiple neural network layers, MLP captures intricate patterns within the data, offering powerful classification capabilities that can adapt to a wide range of data distributions and structures.

B. Implementation Details

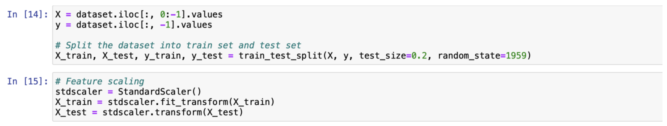
As we apply our dataset processed in Phase 1 to these seven algorithms, we find it necessary to conduct further preprocessing on it. To facilitate algorithmic processing by the computer, we removed the ’state’ column (due to its excessive number of distinct values). Additionally, we converted attributes with a field type of ’string’ to ’integer’, with specific changes detailed in the code. Furthermore, to address the issue of sample imbalance, we utilized undersampling [5], randomly removing negative instance samples to achieve a 1:1 ratio of positive to negative instances. We will provide a more detailed explanation of this operation in the Results and Analysis section.



```
In [11]: # Dataset preprocessing
         dataset = pd.read_csv('cleaned_heart_data.csv')

         dataset = dataset.drop(columns =['State'], axis = 1)
         dataset = dataset.drop(columns =['HadHeartAttack'], axis = 1)

In [12]: columns_to_convert = ['PhysicalActivities',"HadAngina","HadStroke","HadAsthma","HadCOPD","HadKidneyDisease","HadArth
         for column in columns_to_convert:
             dataset[column] = dataset[column].str.strip().str.lower().replace({'yes': 1, 'no': 0})
         dataset['Sex'].replace({'Male': 1, 'Female': 0},inplace=True)
         dataset['GeneralHealth'].replace({'Excellent': 3, 'Very good': 2, 'Good': 1, 'Fair': 0, 'Poor': -1},inplace=True)
         dataset['SmokerStatus'].replace({"Never":0,'Sometimes':1,'Former':2,"Everyday":3},inplace=True)
         dataset['AgeCategory'].replace({'18-24':0,'25-29':1,'30-34':2,'35-39':3,'40-44':4,'45-49':5,'50-54':6,'55-59':7,'60-
         dataset['SleepCategory'].replace({'Very Short':0,'Short':1,'Recommended':2,'Long':3,'Very Long':4},inplace=True)

In [13]: data_majority = dataset[dataset.iloc[:, -1] == 0]
         data_minority = dataset[dataset.iloc[:, -1] == 1]
         data_downsampled = resample(data_majority, replace=False, n_samples=len(data_minority), random_state=376)
         dataset = pd.concat([data_downsampled,data_minority])
```

Fig. 1. Further Dataset Preprocessing

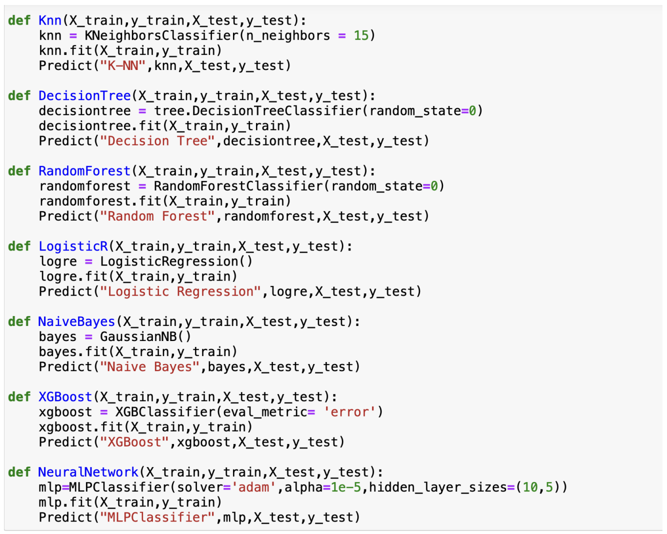
After addressing these issues, we proceeded to split the data into training and testing sets, followed by standardization.



```
In [14]: X = dataset.iloc[:, 0:-1].values
         y = dataset.iloc[:, -1].values

         # Split the dataset into train set and test set
         X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1959)

In [15]: # Feature scaling
         stdscaler = StandardScaler()
         X_train = stdscaler.fit_transform(X_train)
         X_test = stdscaler.transform(X_test)
```

Fig. 2. Dataset Split and Standardization

Now we can define these algorithms, with the Python code as shown in “Fig. 3”.



```
def Knn(X_train,y_train,X_test,y_test):
    knn = KNeighborsClassifier(n_neighbors = 15)
    knn.fit(X_train,y_train)
    Predict("K-NN",knn,X_test,y_test)

def DecisionTree(X_train,y_train,X_test,y_test):
    decisiontree = tree.DecisionTreeClassifier(random_state=0)
    decisiontree.fit(X_train,y_train)
    Predict("Decision Tree",decisiontree,X_test,y_test)

def RandomForest(X_train,y_train,X_test,y_test):
    randomforest = RandomForestClassifier(random_state=0)
    randomforest.fit(X_train,y_train)
    Predict("Random Forest",randomforest,X_test,y_test)

def LogisticR(X_train,y_train,X_test,y_test):
    logre = LogisticRegression()
    logre.fit(X_train,y_train)
    Predict("Logistic Regression",logre,X_test,y_test)

def NaiveBayes(X_train,y_train,X_test,y_test):
    bayes = GaussianNB()
    bayes.fit(X_train,y_train)
    Predict("Naive Bayes",bayes,X_test,y_test)

def XGBoost(X_train,y_train,X_test,y_test):
    xgboost = XGBClassifier(eval_metric= 'error')
    xgboost.fit(X_train,y_train)
    Predict("XGBoost",xgboost,X_test,y_test)

def NeuralNetwork(X_train,y_train,X_test,y_test):
    mlp=MLPClassifier(solver='adam',alpha=1e-5,hidden_layer_sizes=(10,5))
    mlp.fit(X_train,y_train)
    Predict("MLPClassifier",mlp,X_test,y_test)
```

Fig. 3. Definition Code of the Algorithms

Due to the lengthy code required for generating confusion matrix and other evaluation metric plots, we will not display them in the report. Instead, we will present the code results directly in the next section.

III. Results and Analysis

The performance of each algorithm will be rigorously evaluated using a comprehensive suite of metrics including confusion matrix, accuracy, precision, recall, F1-score, ROC curves, and PR curves, providing a holistic assessment of their efficacy in discerning heart disease risk.

A. Performance Metrics and Evaluation

Firstly, let’s display the confusion matrices and common evaluation metrics for the output results of the seven algorithms. Since they are within the scope of the lecture, we will not explain the meanings of these evaluation metrics further.

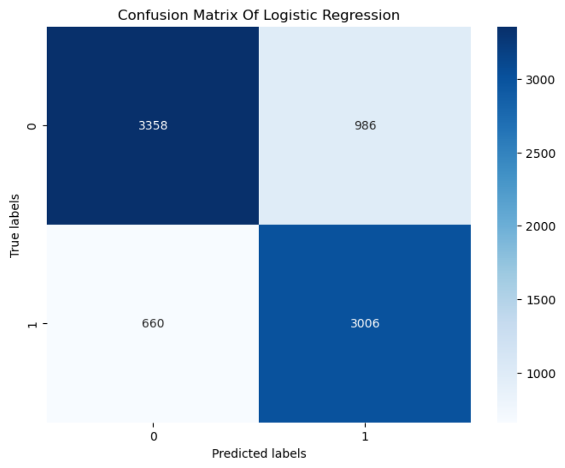
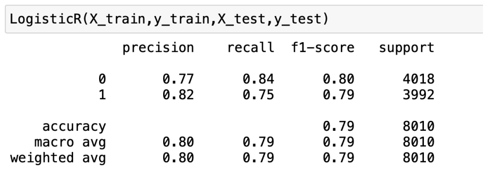


Fig. 4. Logistic Regression



```
LogisticR(X_train,y_train,X_test,y_test)

              precision    recall  f1-score   support

           0       0.77      0.84      0.80      4018
           1       0.82      0.75      0.79      3992

    accuracy                           0.79      8010
   macro avg       0.80      0.79      0.79      8010
weighted avg       0.80      0.79      0.79      8010
```

Fig. 5. Logistic Regression

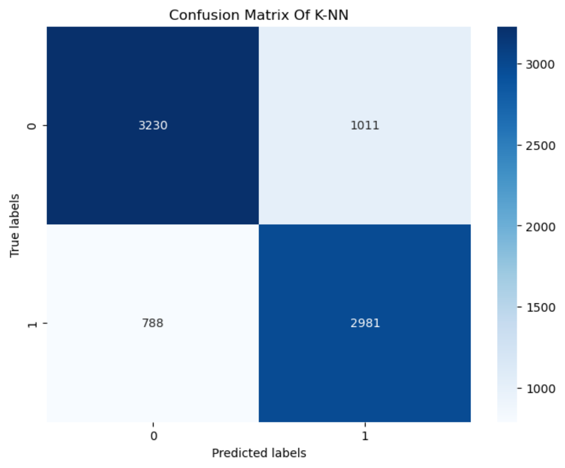
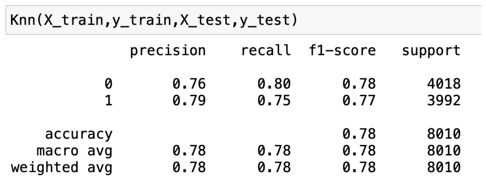


Fig. 6. K-NN



```
Knn(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.76      0.80      0.78      4018
           1       0.79      0.75      0.77      3992

    accuracy                           0.78      8010
   macro avg       0.78      0.78      0.78      8010
weighted avg       0.78      0.78      0.78      8010
```

Fig. 7. K-NN

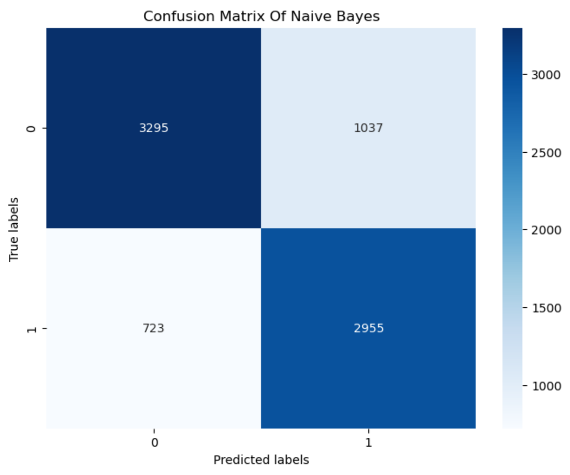
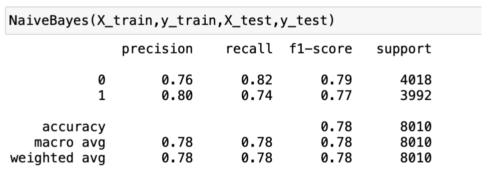


Fig. 8. Naive Bayes



```
NaiveBayes(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.76      0.82      0.79      4018
           1       0.80      0.74      0.77      3992

    accuracy                           0.78      8010
   macro avg       0.78      0.78      0.78      8010
weighted avg       0.78      0.78      0.78      8010
```

Fig. 9. Naive Bayes

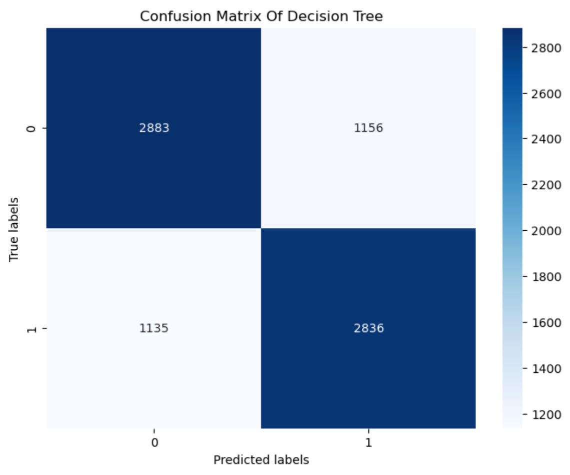
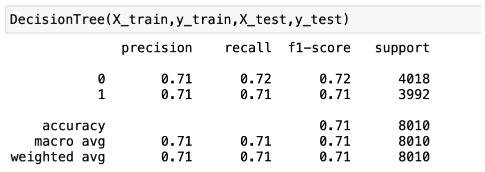


Fig. 10. Decision Tree



```
DecisionTree(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.71      0.72      0.72      4018
           1       0.71      0.71      0.71      3992

    accuracy                           0.71      8010
   macro avg       0.71      0.71      0.71      8010
weighted avg       0.71      0.71      0.71      8010
```

Fig. 11. Decision Tree

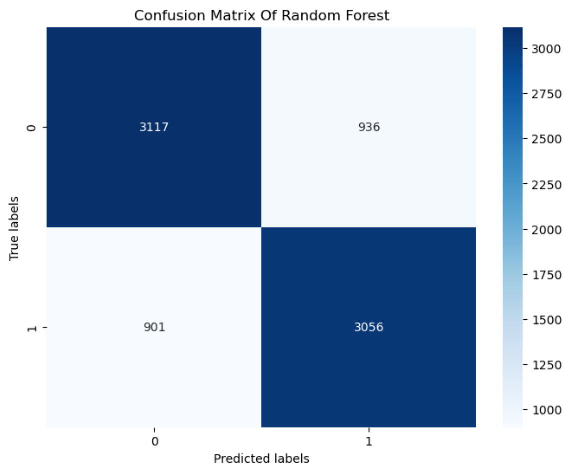
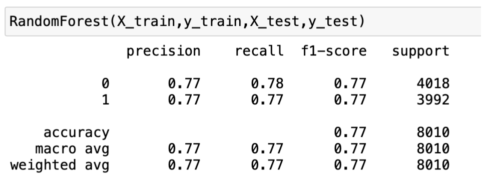


Fig. 12. Random Forest



```
RandomForest(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.77      0.78      0.77      4018
           1       0.77      0.77      0.77      3992

    accuracy                           0.77      8010
   macro avg       0.77      0.77      0.77      8010
weighted avg       0.77      0.77      0.77      8010
```

Fig. 13. Random Forest

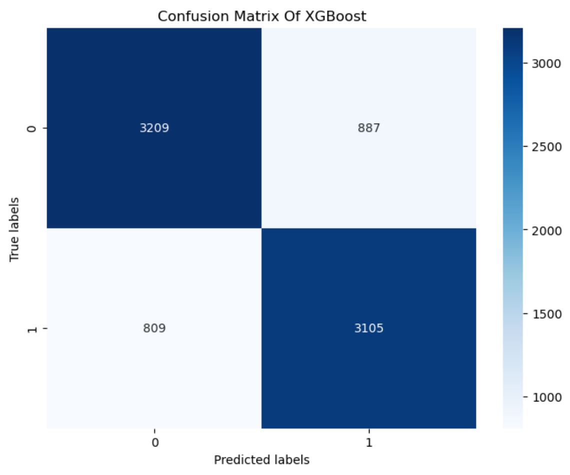
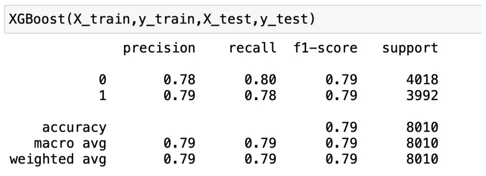


Fig. 14. XGBoost



```
XGBoost(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.78      0.80      0.79      4018
           1       0.79      0.78      0.79      3992

    accuracy                           0.79      8010
   macro avg       0.79      0.79      0.79      8010
weighted avg       0.79      0.79      0.79      8010
```

Fig. 15. XGBoost

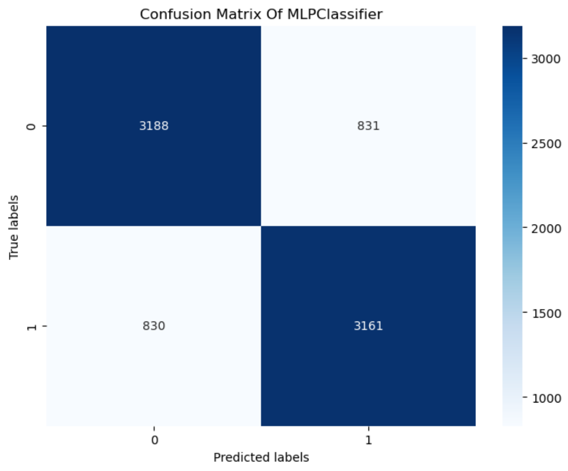
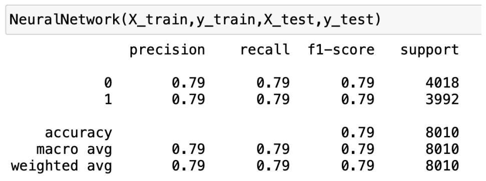


Fig. 16. MLP



```
NeuralNetwork(X_train,y_train,X_test,y_test)
              precision    recall  f1-score   support

           0       0.79      0.79      0.79      4018
           1       0.79      0.79      0.79      3992

    accuracy                           0.79      8010
   macro avg       0.79      0.79      0.79      8010
weighted avg       0.79      0.79      0.79      8010
```

Fig. 17. MLP

From the results, our algorithms achieve accuracy rates of over 70% on the test set. Except for decision tree, the accuracy of the remaining algorithms reaches around 80%, and they also perform well on the other three common evaluation metrics.

Therefore, we focus on analyzing the underperformance of the decision tree, which may stem from its tendency to generate complex models when dealing with numerous features, leading to overfitting on the training data. Overfitting results in decreased performance of the model on new data, lacking generalization ability.

In contrast, both Random Forest and XGBoost algorithms, despite being based on decision trees, have made notable improvements. Random Forest utilizes multiple decision trees and aggregates their predictions through voting, enabling it to handle high-dimensional data efficiently. Similarly, XGBoost integrates weak learners into a strong learner and updates predictions based on the differences between the target values and predictions of previous trees. Additionally, XGBoost introduces regularization terms such as L1 and L2 regularization to control model complexity and reduce overfitting.

B. *Other Visualizations of Algorithms*

Now let's show the results for ROC curves and PR curves.

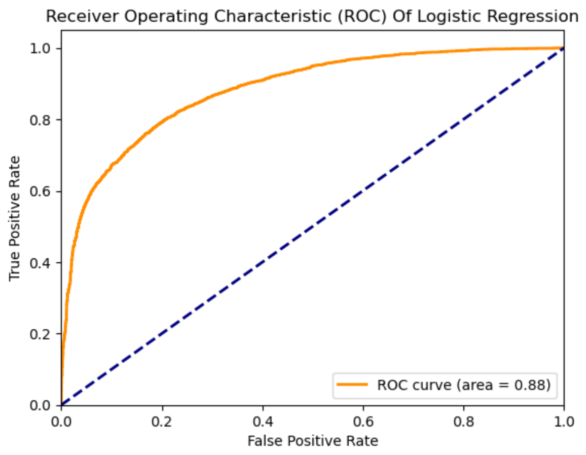


Fig. 18. Logistic Regression

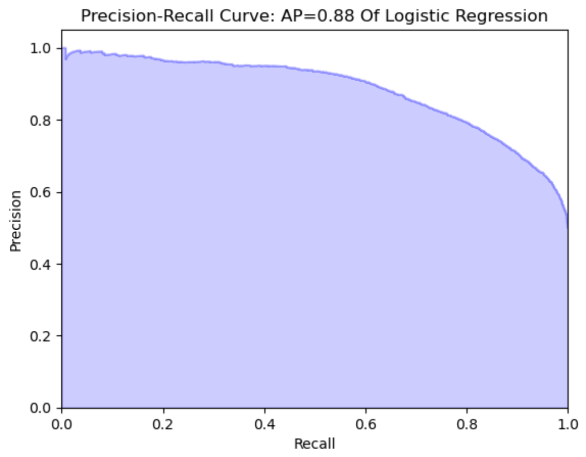


Fig. 19. Logistic Regression

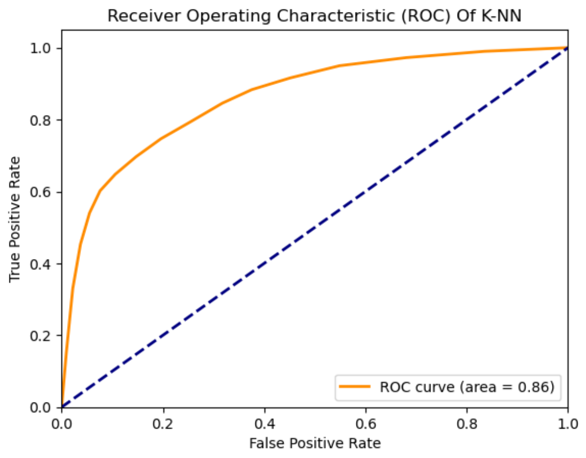


Fig. 20. K-NN

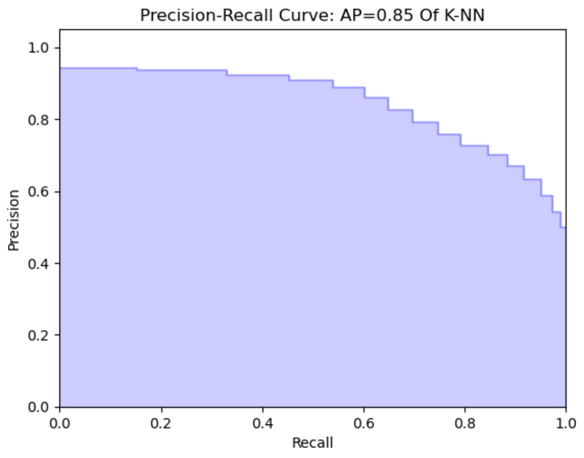


Fig. 21. K-NN

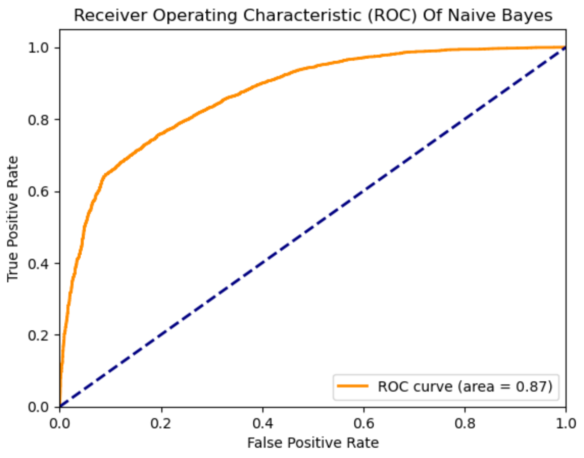


Fig. 22. Naive Bayes

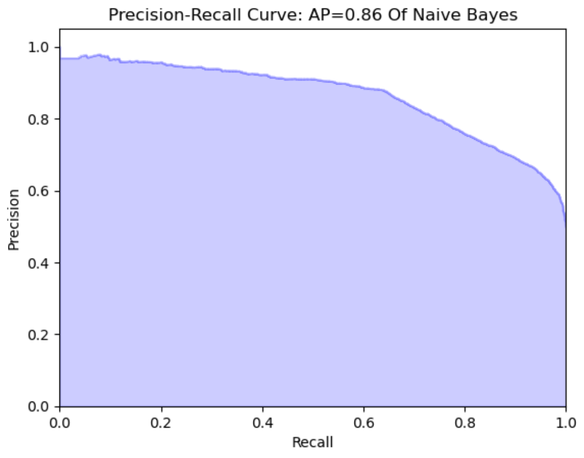


Fig. 23. Naive Bayes

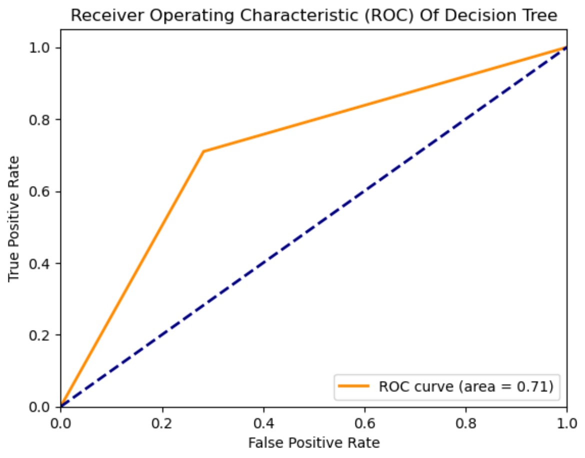


Fig. 24. Decision Tree

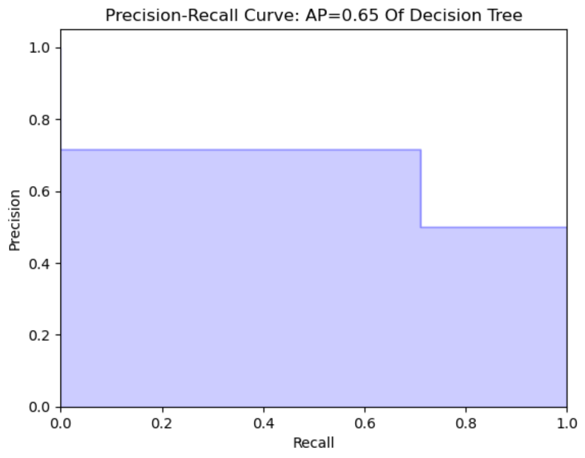


Fig. 25. Decision Tree

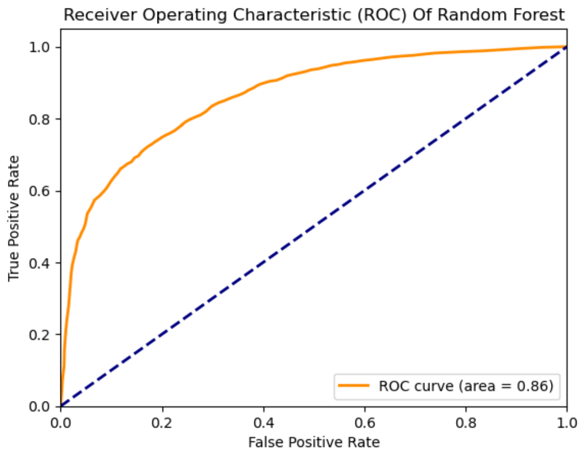


Fig. 26. Random Forest

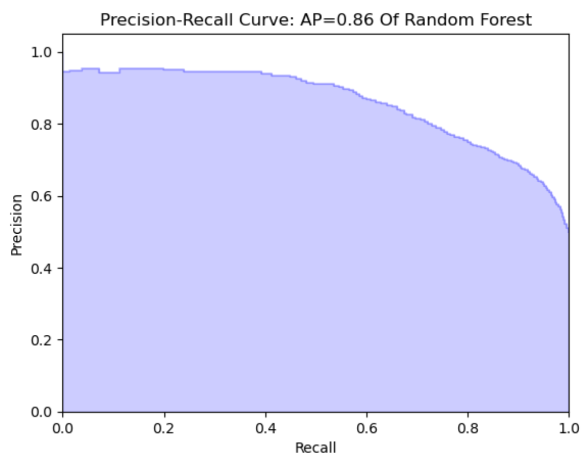


Fig. 27. Random Forest

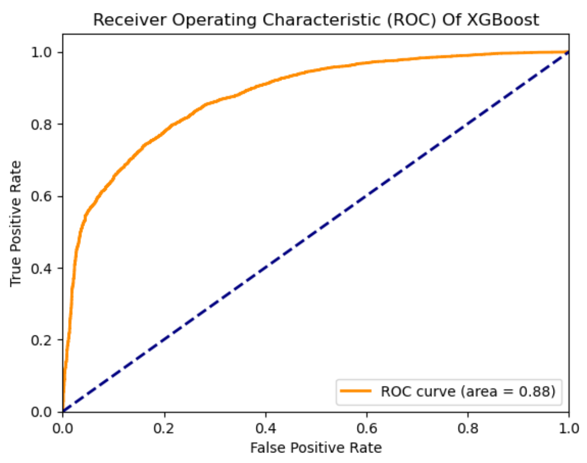


Fig. 28. XGBoost

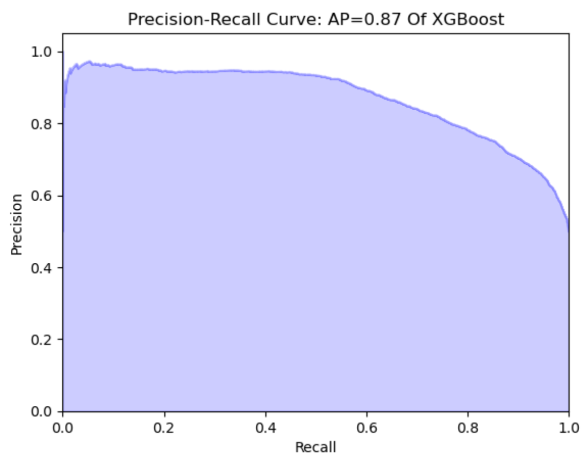


Fig. 29. XGBoost

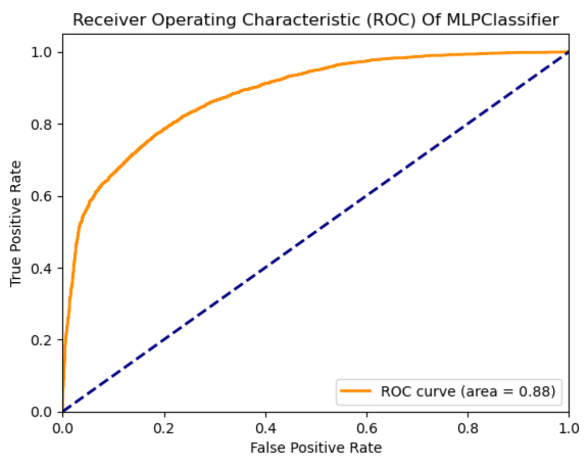


Fig. 30. MLP

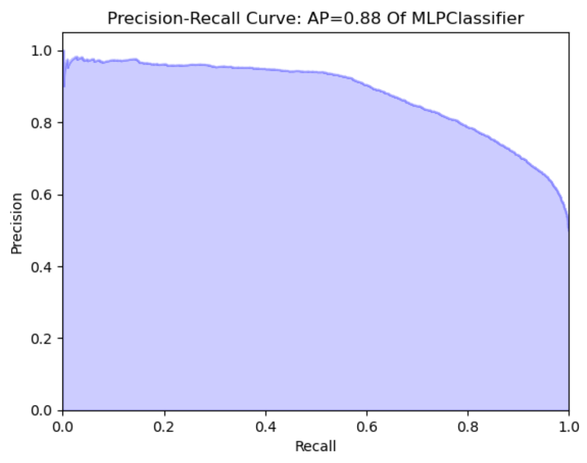


Fig. 31. MLP

In our project, the use of ROC curves and PR curves serves as essential tools for evaluating the generalization performance of our classification models.

The ROC curve assesses the model's ability to distinguish between different classes by plotting the true positive rate against the false positive rate. AUC, the area under the ROC curve, quantifies the model's discriminative power, with values closer to 1 indicating better performance. Conversely, an AUC of 0.5 suggests random guessing.

Similarly, the PR curve evaluates the model's generalization ability but places more emphasis on false positives rather than false negatives. A PR curve with a larger area indicates a better predictive performance, particularly when the area exceeds 0.8 or 0.9, suggesting reliable predictions and precise outcomes.

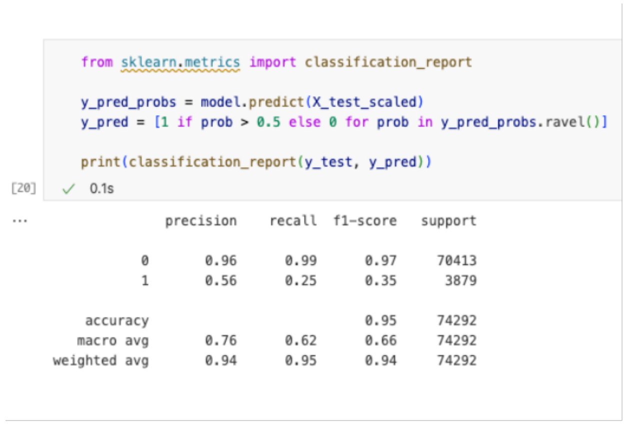
In our heart disease prediction model, while the decision tree model exhibits an AUC value around 0.7, the remaining algorithm models surpass an AUC of 0.85. This signifies that the training models, constructed using various machine learning algorithms on raw data obtained from telephone survey questionnaires and processed through data preprocessing and filtering, possess a certain capability for identifying individuals at risk of heart disease. When the model outputs a positive result for a test sample, it indicates a need for the individual to pay particular attention to heart health issues and seek medical assistance.

C. *Insights Gained through Implementation*

As we have mentioned above, we utilize undersampling to achieve a 1:1 ratio of positive to negative instances. This is the first insight we obtained from our implementation. That is to say, to achieve the balance between positive and negative samples in the training set is essential before applying the dataset to models. let us elaborate the reason below.

After processing in Phase 1, the ratio of data with and without heart disease in our dataset is approximately 1:17.5. In other words, there is a severe imbalance between positive and negative instances. This is due to the dataset being sourced from a sampling survey, where the proportion of individuals with heart disease in the total population is inevitably smaller than those without it. Consequently, due to probabilistic reasons, the number of individuals with heart disease in the dataset is much fewer than those without it. However, this imbalance significantly affects the output results of the model during training.

In our initial attempt, we did not notice this issue and directly provided the training set to the model. The results obtained from the test set are as shown in “Fig. 32”.



```
from sklearn.metrics import classification_report

y_pred_probs = model.predict(X_test_scaled)
y_pred = [1 if prob > 0.5 else 0 for prob in y_pred_probs.ravel()]

print(classification_report(y_test, y_pred))
```

[20] ✓ 0.1s

```
              precision    recall  f1-score   support

           0       0.96      0.99      0.97     70413
           1       0.56      0.25      0.35      3879

    accuracy                           0.95     74292
   macro avg       0.76      0.62      0.66     74292
weighted avg       0.94      0.95      0.94     74292
```

Fig. 32. Initial Result for Logistic Regression

We can observe that the model is relatively good at identifying the negative class. However, it performs poorly on the positive class. This phenomenon is common in imbalanced datasets and is particularly concerning in medical prediction tasks where failing to identify positive cases, such as heart attacks, can have serious consequences.

There is a significant class imbalance, with the ’no heart attack’ class being much larger than the ’heart attack’ class. This can lead to high accuracy being misleading, as the model could achieve high accuracy by mostly predicting the majority class correctly.

We also have output results from other algorithms, which similarly perform poorly on the positive class. These results will not be displayed here.

Upon above analysis, several insights have been gained. Firstly, logistic regression, k-NN, and Naive Bayes demonstrate relatively balanced performance with F1-scores ranging from 0.77 to 0.80 for class 1 and 0.78 to 0.80 for class 0. These algorithms exhibit stable precision and recall rates, indicating their ability to effectively classify both positive and negative instances.

However, the decision tree algorithm shows slightly lower performance compared to other algorithms, with F1-scores of 0.71 for both classes. This may be attributed to its inherent tendency to create complex decision boundaries, leading to suboptimal generalization on unseen data.

In contrast, both Random Forest and XGBoost algorithms exhibit robust performance across all metrics, with F1-scores consistently above 0.77 for both classes. These ensemble methods leverage multiple weak learners to make predictions, effectively reducing overfitting and improving generalization performance.

D. *Insights Gained from Results*

Upon output results, several insights have been gained. Here we present a comparison plot of the algorithm results.

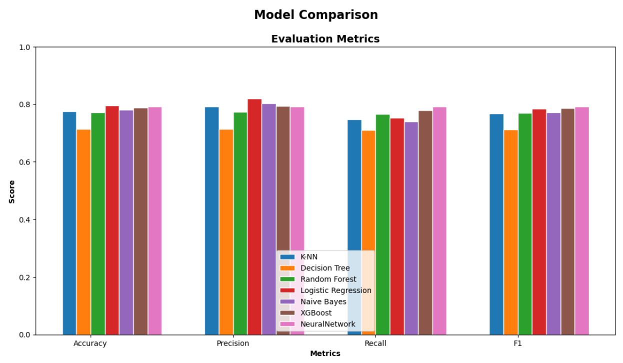


Fig. 33. Comparison between Algorithms

Firstly, logistic regression, k-NN, and Naive Bayes demonstrate relatively balanced performance with F1-scores ranging from 0.77 to 0.80 for class 1 and 0.78 to 0.80 for class 0. These algorithms exhibit stable precision and recall rates, indicating their ability to effectively classify both positive and negative instances.

However, the decision tree algorithm shows slightly lower performance compared to other algorithms, with F1-scores of 0.71 for both classes. This may be attributed to its inherent tendency to create complex decision boundaries, leading to suboptimal generalization on unseen data.

In contrast, both Random Forest and XGBoost algorithms exhibit robust performance across all metrics, with F1-scores consistently above 0.77 for both classes. These ensemble methods leverage multiple weak learners to make predictions, effectively reducing overfitting and improving generalization performance.

Furthermore, the Multilayer Perceptron (MLP) algorithm demonstrates competitive performance, with F1-scores of 0.79

for both classes. The identical precision and recall values for MLP may suggest that the algorithm handles the dataset with more consistency and balance. This could imply that MLP treats positive and negative instances more evenly during the learning process, or it pays more uniform attention to different aspects of the dataset during training.

IV. Summary

In summary, this report presents the analysis of seven algorithms for heart disease risk assessment. Leveraging dataset processed in phase 1, we apply machine learning and statistical modules to develop classification models which can distinguish between high and low-risk individuals. Through algorithm selection, coding implementation, and rigorous evaluation, we demonstrate the effectiveness of various algorithms. While some algorithms exhibited superior performance, others, like the decision tree, showed limitations. Insights gained from the analysis highlighted the importance of ensemble methods and regularization techniques in enhancing model performance. Overall, we have successfully completed the task of establishing a usable classification model for assessment of heart disease risk.

References

[1] Li, B., Friedman, J., Olshen, R., & Stone, C. (1984). Classification and regression trees (CART). Biometrics, 40(3), 358-361.

[2] Breiman, L. (2001). Random forests. Machine learning, 45, 5-32.

[3] Chen, T., & Guestrin, C. (2016, August). Xgboost: A scalable tree boosting system. In Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining (pp. 785-794).

[4] Amato, F., Mazzocca, N., Moscato, F., & Vivenzio, E. (2017, March). Multilayer perceptron: an intelligent model for classification and intrusion detection. In 2017 31st International conference on advanced information networking and applications workshops (WAINA) (pp. 686-691). IEEE.

[5] Wallace, B. C., & Dahabreh, I. J. (2012, December). Class probability estimates are unreliable for imbalanced data (and how to fix them). In 2012 IEEE 12th international conference on data mining (pp. 695-704). IEEE.