Mini Project

PREDICTION OF HEART DISEASE

EEX7244

Data Mining

By

Sachintha Heshan Kasthuriarachchi

516955583

Submitted to

Department of Electrical and Computer Engineering

Faculty of Engineering Technology

The Open University of Sri Lanka

At

Colombo Regional Center

On

28th December 2024

Table of Contents

List of Figure

Introduction

Heart disease is one of the leading causes of death globally, emphasizing the importance of early detection and diagnosis to improve patient outcomes. In this project, we employ Data Mining techniques to analyze patient health data and develop a classification system that predicts the likelihood of heart disease. Data Mining is a systematic process of discovering meaningful patterns, correlations, and insights from large datasets, often using machine learning algorithms for prediction and decision-making.

The dataset used for this study contains 918 records and 12 attributes, representing key health indicators such as Age, Sex, ChestPainType, RestingBP, Cholesterol, MaxHR, and a target variable, HeartDisease, which indicates whether the patient has heart disease (1) or not (0). Additional features like FastingBS, RestingECG, ExerciseAngina, and ST_Slope provide further insights into patients' clinical and physical conditions.

Dataset Attributes

- **Age** : age of the patient [years]
- **Sex** : sex of the patient [M: Male, F: Female]
- **ChestPainType** : chest pain type [TA: Typical Angina, ATA: Atypical Angina, NAP: Non-Anginal Pain, ASY: Asymptomatic]
- **RestingBP** : resting blood pressure [mm Hg]
- **Cholesterol** : serum cholesterol [mm/dl]
- **FastingBS** : fasting blood sugar [1: if FastingBS > 120 mg/dl, 0: otherwise]
- **RestingECG** : resting electrocardiogram results [Normal: Normal, ST: having ST-T wave abnormality (T wave inversions and/or ST elevation or depression of > 0.05 mV), LVH: showing probable or definite left ventricular hypertrophy by Estes' criteria]
- **MaxHR** : maximum heart rate achieved [Numeric value between 60 and 202]
- **ExerciseAngina** : exercise-induced angina [Y: Yes, N: No]
- **Oldpeak** : oldpeak = ST [Numeric value measured in depression]
- **ST_Slope** : the slope of the peak exercise ST segment [Up: upsloping, Flat: flat, Down: downsloping]
- **HeartDisease** : output class [1: heart disease, 0: Normal]

Link: https://www.kaggle.com/datasets/johnsmith88/heart-disease-dataset/data

Problem Background

Heart disease is a major public health challenge and a leading cause of death worldwide. Early detection and timely intervention are crucial to managing heart disease effectively, as delayed diagnosis often leads to severe health complications or fatal outcomes. The diagnosis of heart disease typically relies on a combination of clinical tests, patient history, and physician expertise. However, this traditional approach can be time-consuming, subjective, and prone to inaccuracies.

In recent years, advancements in **Data Mining** and **Machine Learning** have provided new opportunities to analyze large volumes of health data systematically and accurately. These techniques allow the discovery of hidden patterns and relationships within datasets that are not immediately apparent to human analysts. By leveraging machine learning algorithms, we can automate the process of predicting heart disease, enhancing the precision and efficiency of the diagnosis process.

The motivation for this study arises from the increasing availability of patient health data and the urgent need for tools that can support healthcare professionals in diagnosing heart disease. The goal is to develop a classification system capable of predicting whether an individual is at risk of heart disease based on their health attributes, such as age, cholesterol levels, blood pressure, and exercise tolerance. Such a system can serve as an initial screening tool, flagging high-risk patients for further medical evaluation and enabling early intervention.

This project aligns with the broader objectives of predictive healthcare, where the focus is on prevention and proactive management of diseases through data-driven insights. By addressing this problem using Data Mining techniques, this study contributes to improving healthcare outcomes and reducing the burden of heart disease on patients and healthcare systems.

Literature review

The prediction of heart disease through data mining techniques has garnered significant attention in recent years due to the increasing prevalence of cardiovascular diseases. Various studies have employed different data mining methodologies to enhance prediction accuracy, utilizing a range of algorithms and datasets. This review synthesizes key findings from recent literature, highlighting the strengths and limitations of various approaches.

Overview of Data Mining in Healthcare

Data mining has emerged as a critical tool in healthcare, particularly for predicting heart disease. The ability to analyze large datasets allows for the identification of patterns and relationships that can inform clinical decision-making. Supervised learning techniques, such as classification algorithms, are predominantly used for this purpose, enabling models to predict outcomes based on historical data.

Commonly Used Algorithms

Several classification algorithms have been extensively studied for heart disease prediction. Support Vector Machines (SVM), This algorithm has been shown to effectively classify heart disease cases with high accuracy. It works well in high-dimensional spaces and is effective in cases where the number of dimensions exceeds the number of samples [1].

K-Nearest Neighbors (KNN), KNN is favored for its simplicity and effectiveness in non-linear data distributions. Studies indicate that it performs comparably to more complex algorithms when tuned properly [1][6].

Decision Trees, these models are intuitive and provide clear visualization of decision-making processes. They have been successfully used in various studies, demonstrating their effectiveness in handling categorical data [4][8].

Neural Networks, advanced neural network architectures, including deep learning models, have also been employed. These models can capture complex relationships within the data but require substantial computational resources [1][2].

Feature Selection and Data Preprocessing

Effective feature selection is crucial for enhancing model performance. Studies emphasize the importance of identifying significant features that contribute to heart disease risk, such as cholesterol levels, blood pressure, age, and other clinical parameters. Techniques like Principal Component Analysis (PCA) and Recursive Feature Elimination (RFE) are commonly employed to reduce dimensionality and improve model interpretability [2][4].

Data preprocessing steps such as normalization, handling missing values, and encoding categorical variables are also critical for ensuring the robustness of predictive models.

Challenges and Future Directions

Despite advancements in data mining techniques for heart disease prediction, several challenges remain.

Data Quality: Inconsistent or incomplete data can significantly affect model accuracy. Ensuring high-quality datasets is essential for reliable predictions.

Interpretability: Many advanced models (e.g., neural networks) operate as "black boxes," making it difficult for clinicians to understand how predictions are made.

Generalization: Models trained on specific datasets may not perform well across different populations or settings. There is a need for more generalized models that can adapt to diverse clinical environments.

Future research should focus on developing hybrid models that leverage the strengths of various algorithms while addressing these challenges. Additionally, incorporating real-time data from wearable devices could enhance predictive capabilities further.

As a summary, the application of data mining techniques in predicting heart disease shows promising potential but requires ongoing research to optimize methods and improve clinical applicability.

Methodology

The methodology outlines the systematic approach employed to develop a machine learning-based system for predicting heart disease. It includes steps for data preprocessing, feature selection, model training, and evaluation. Three models—Support Vector Machines (SVM), K-Nearest Neighbors (KNN), and Random Forest—are implemented and compared using a rigorous evaluation strategy.

Data preprocessing is a critical step to ensure the quality and reliability of the dataset. Missing or inconsistent values are removed to avoid errors during model training. Numerical features like Age and Cholesterol are normalized to bring them onto a common scale, enhancing the model's learning process. Categorical variables, such as ChestPainType and ST_Slope, are encoded into numerical formats to make them suitable for machine learning algorithms.

Feature selection is performed to identify and retain only the most relevant attributes that significantly impact heart disease prediction. Attributes like Age, Cholesterol, and MaxHR are prioritized due to their strong correlation with heart disease. Techniques such as Recursive Feature Elimination (RFE) and correlation analysis help reduce dimensionality and improve model performance.

Three machine learning models are developed to predict heart disease:

- **Support Vector Machines (SVM):** SVM is used for its ability to classify data points by finding an optimal hyperplane that separates classes effectively. It is particularly useful for handling both linear and non-linear datasets.
- **K-Nearest Neighbors (KNN):** KNN is chosen for its simplicity and effectiveness. It classifies a data point based on the majority class of its 'k' nearest neighbors, making it intuitive and easy to implement.
- **Random Forest (RF):** RF, an ensemble learning method, is employed for its robustness and ability to model complex relationships. It builds multiple decision trees and combines their predictions to enhance accuracy and minimize overfitting.

The models are evaluated using two robust strategies:

- **10-fold Cross-Validation:** The dataset is divided into 10 subsets. Each subset serves as test data once while the remaining nine subsets are used for training. This ensures all data points are tested and helps gauge the model's consistency.
- **3% Train-Test Split:** A small portion (3%) of the dataset is reserved for testing, while the remaining 97% is used for training. This strategy offers a quick assessment of model performance.

Evaluation metrics such as accuracy, precision, recall, and F1-score are calculated to compare the models and determine the best-performing one.

The performance of SVM, KNN, and Random Forest is analyzed to identify patterns and insights. Random Forest typically excels due to its ensemble approach, which handles complex feature interactions and reduces overfitting. SVM provides balanced results but may struggle with large datasets. KNN, while intuitive, is sensitive to noise and works best with smaller datasets.

Design

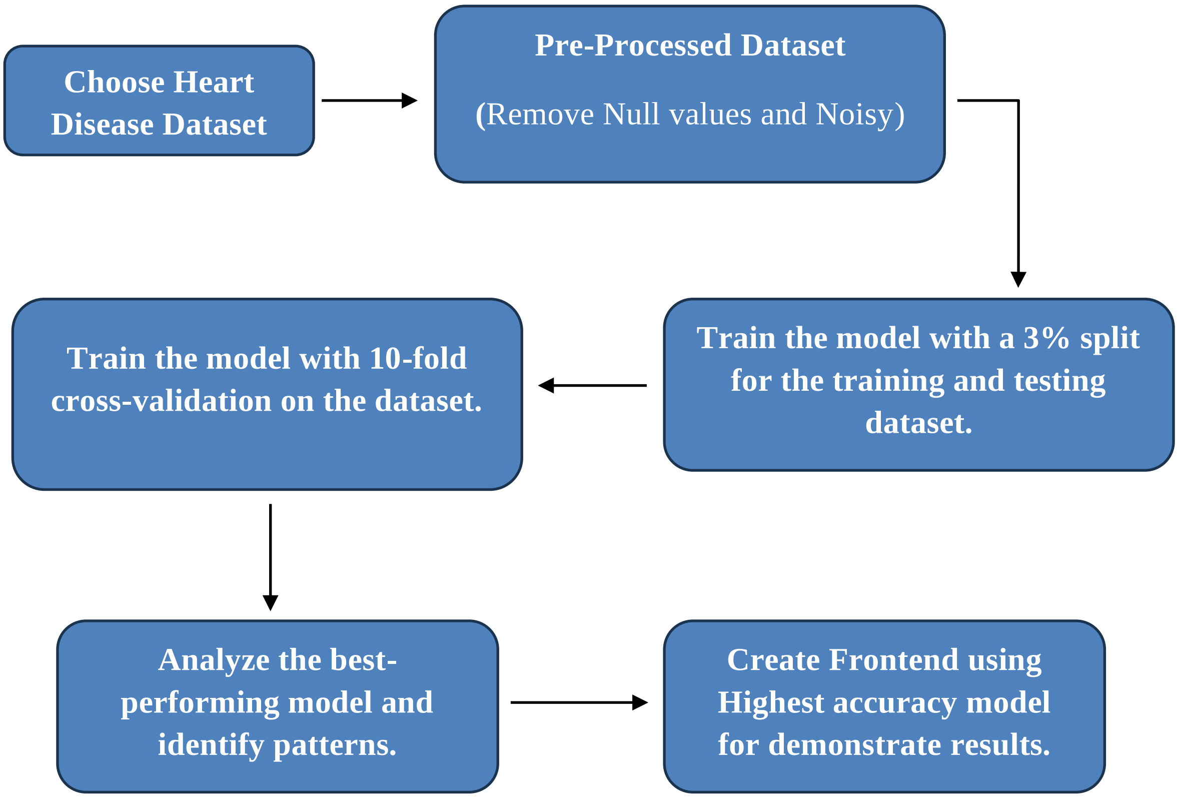


Figure 1 Block Diagram of the Methodology

Development

Using Python code

Libraries

```
[131] import pandas as pd
      import seaborn as sns
      import matplotlib.pyplot as plt
      from sklearn.model_selection import train_test_split
      from sklearn.neighbors import KNeighborsClassifier
      from sklearn.svm import SVC
      from sklearn.ensemble import RandomForestClassifier
      from sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score, confusion_matrix
      from sklearn.model_selection import cross_val_score, StratifiedKFold
      import numpy as np
```

Figure 2 Imported Libraries

Dataset Import

01). Import dataset and Explore dataset type and Quntity

```
[132] data = pd.read_csv('/content/heart_disease.csv')
      data.head()
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	FastingBS	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40.0	M	ATA	140	289.0	0.0	Normal	172.0	N	0.0	Up	0.0
1	49.0	F	NAP	160	180.0	0.0	Normal	156.0	N	1.0	Flat	1.0
2	37.0	M	ATA	130	283.0	0.0	ST	98.0	N	0.0	Up	0.0
3	48.0	F	NaN	138	214.0	0.0	Normal	108.0	Y	1.5	Flat	1.0
4	54.0	M	NAP	150	195.0	0.0	Normal	122.0	N	0.0	Up	0.0

Next steps: Generate code with data | View recommended plots | New interactive sheet

```
0s [133] print('Shape of the data is ', data.shape)
```

```
Shape of the data is  (918, 12)
```

Figure 3 Dataset Import

Data Types

```
0s [134] data.dtypes
```

	0
Age	float64
Sex	object
ChestPainType	object
RestingBP	int64
Cholesterol	float64
FastingBS	float64
RestingECG	object
MaxHR	float64
ExerciseAngina	object
Oldpeak	float64
ST_Slope	object
HeartDisease	float64

dtype: object

Figure 4 Data Types

Data Pre-Processing

02). Dataset Preprocessing

```
[135] # Check for null values
      print(data.isnull().sum())
```

```
Age                5
Sex                6
ChestPainType      5
RestingBP          0
Cholesterol        6
FastingBS          1
RestingECG        13
MaxHR              2
ExerciseAngina     2
Oldpeak            0
ST_Slope           5
HeartDisease       2
dtype: int64
```

```
[136] # Drop All null Values
      data = data.dropna()
```

```
[137] print(data.isnull().sum())
      print(data.shape)
```


```
[137] print(data.isnull().sum())
      print(data.shape)
```


```
Age               0
Sex               0
ChestPainType     0
RestingBP         0
Cholesterol       0
FastingBS         0
RestingECG        0
MaxHR             0
ExerciseAngina    0
Oldpeak           0
ST_Slope          0
HeartDisease      0
dtype: int64
(875, 12)
```

```
[138] #print characters colum values
      print(data['Sex'].unique())
      print(data['ChestPainType'].unique())
      print(data['RestingECG'].unique())
      print(data['ExerciseAngina'].unique())
      print(data['ST_Slope'].unique())
```

```
['M' 'F']
['ATA' 'NAP' 'ASY' 'TA']
['Normal' 'ST' 'LVH']
['N' 'Y']
['Up' 'Flat' 'Down']
```

```python
# Sample columns that need encoding
binary_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']

# Encode binary columns with 1 and 0
binary_mapping = {
    'M': 1, 'F': 0,
    'ATA': 0, 'NAP': 1, 'ASY': 2, 'TA': 3,
    'Normal': 0, 'ST': 1, 'LVH': 2,
    'N': 0, 'Y': 1,
    'Up': 0, 'Flat': 1, 'Down': 2

}

for col in binary_columns:
    data[col] = data[col].map(binary_mapping)

# Check encoding results
print(data.head())
```

```
    Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0  40.0    1              0        140        289.0        0.0           0
1  49.0    0              1        160        180.0        0.0           0
2  37.0    1              0        130        283.0        0.0           1
4  54.0    1              1        150        195.0        0.0           0
5  39.0    1              1        120        339.0        0.0           0

   MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
0  172.0               0      0.0         0           0.0
1  156.0               0      1.0         1           1.0
2   98.0               0      0.0         0           0.0
```

```python
[140] data.dtypes
```

	0
Age	float64
Sex	int64
ChestPainType	int64
RestingBP	int64
Cholesterol	float64
FastingBS	float64
RestingECG	int64
MaxHR	float64
ExerciseAngina	int64
Oldpeak	float64
ST_Slope	int64
HeartDisease	float64

dtype: object

```
[141] print(data)
```

```
      Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0    40.0    1              0        140        289.0        0.0           0
1    49.0    0              1        160        180.0        0.0           0
2    37.0    1              0        130        283.0        0.0           1
4    54.0    1              1        150        195.0        0.0           0
5    39.0    1              1        120        339.0        0.0           0
..    ...  ...            ...        ...          ...        ...         ...
913  45.0    1              3        110        264.0        0.0           0
914  68.0    1              2        144        193.0        1.0           0
915  57.0    1              2        130        131.0        0.0           0
916  57.0    0              0        130        236.0        0.0           2
917  38.0    1              1        138        175.0        0.0           0

     MaxHR  ExerciseAngina  Oldpeak  ST_Slope  HeartDisease
0    172.0               0      0.0         0           0.0
1    156.0               0      1.0         1           1.0
2     98.0               0      0.0         0           0.0
4    122.0               0      0.0         0           0.0
5    170.0               0      0.0         0           0.0
..     ...             ...      ...       ...           ...
913  132.0               0      1.2         1           1.0
914  141.0               0      3.4         1           1.0
915  115.0               1      1.2         1           1.0
916  174.0               0      0.0         1           1.0
917  173.0               0      0.0         0           0.0

[875 rows x 12 columns]
```

Figure 5 Data Pre-processing

Dataset Training

03). Dataset Training

```
[142] # Separate features and target variable
      X = data.drop('HeartDisease', axis=1)  # Features
      y = data['HeartDisease']               # Target
```

```
[143] print(X)
```

```
      Age  Sex  ChestPainType  RestingBP  Cholesterol  FastingBS  RestingECG  \
0    40.0    1              0        140        289.0        0.0           0
1    49.0    0              1        160        180.0        0.0           0
2    37.0    1              0        130        283.0        0.0           1
4    54.0    1              1        150        195.0        0.0           0
5    39.0    1              1        120        339.0        0.0           0
..    ...  ...            ...        ...          ...        ...         ...
913  45.0    1              3        110        264.0        0.0           0
914  68.0    1              2        144        193.0        1.0           0
915  57.0    1              2        130        131.0        0.0           0
916  57.0    0              0        130        236.0        0.0           2
917  38.0    1              1        138        175.0        0.0           0

     MaxHR  ExerciseAngina  Oldpeak  ST_Slope
0    172.0               0      0.0         0
1    156.0               0      1.0         1
2     98.0               0      0.0         0
4    122.0               0      0.0         0
5    170.0               0      0.0         0
```

```
[144] print(y)
```

```
0      0.0
1      1.0
2      0.0
4      0.0
5      0.0
      ...
913    1.0
914    1.0
915    1.0
916    1.0
917    0.0
Name: HeartDisease, Length: 875, dtype: float64
```

```
[145] # Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.3, random_state=42)
```

```
[148] # Train the SVM model
svm_model = SVC(kernel='rbf', C=1, gamma='scale', random_state=42)
svm_model.fit(X_train, y_train)
```

```
SVC
SVC(C=1, random_state=42)
```

```
[149] knn_model = KNeighborsClassifier(n_neighbors=5)
knn_model.fit(X_train, y_train)
```

```
KNeighborsClassifier
KNeighborsClassifier()
```

```
[150] RandamForest_model = RandomForestClassifier(n_estimators=100, random_state=42)
RandamForest_model.fit(X_train, y_train)
```

```
RandomForestClassifier
RandomForestClassifier(random_state=42)
```

Figure 6 Dataset Training

10-fold cross validation SVM

```
[159] # Define the cross-validation strategy (e.g., 10-fold) for SVM
cv_svm = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Perform cross-validation and get the accuracy scores
cv_scores_svm = cross_val_score(svm_model, X, y, cv=cv_svm, scoring='accuracy')

# Output the results
print("Cross-Validation Scores:", cv_scores_svm)
print("Mean Accuracy:", np.mean(cv_scores_svm,))
print("Standard Deviation:", np.std(cv_scores_svm))
```

```
Cross-Validation Scores: [0.75       0.75       0.70454545 0.78409091 0.71590909 0.68965517
 0.70114943 0.59770115 0.79310345 0.68965517]
Mean Accuracy: 0.7175809822361547
Standard Deviation: 0.05348157716659597
```

Figure 7 Cross Fold validation SVM

10-fold cross validation KNN

```
[161] # Define the cross-validation strategy (e.g., 10-fold) for KNN
      cv_knn = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

      # Perform cross-validation and get the accuracy scores
      cv_scores_knn = cross_val_score(knn_model, X, y, cv=cv_knn, scoring='accuracy')

      # Output the results
      print("Cross-Validation Scores:", cv_scores_knn)
      print("Mean Accuracy:", np.mean(cv_scores_knn))
      print("Standard Deviation:", np.std(cv_scores_knn))
```

```
Cross-Validation Scores: [0.73863636 0.75       0.70454545 0.68181818 0.71590909 0.68965517
 0.64367816 0.6091954  0.74712644 0.70114943]
Mean Accuracy: 0.698171368861024
Standard Deviation: 0.0427908059291661
```

Figure 8 Cross fold validation KNN

10-fold cross validation RandamForest

```
# Define the cross-validation strategy (e.g., 10-fold) for Random Forest
cv_rf = StratifiedKFold(n_splits=10, shuffle=True, random_state=42)

# Perform cross-validation and get the accuracy scores
cv_scores_rf = cross_val_score(RandamForest_model, X, y, cv=cv_rf, scoring='accuracy')

# Output the results
print("Cross-Validation Scores:", cv_scores_rf)
print("Mean Accuracy:", np.mean(cv_scores_rf))
print("Standard Deviation:", np.std(cv_scores_rf))
```

```
Cross-Validation Scores: [0.88636364 0.86363636 0.85227273 0.88636364 0.82954545 0.88505747
 0.85057471 0.83908046 0.88505747 0.86206897]
Mean Accuracy: 0.8640020898641589
Standard Deviation: 0.020042021332142022
```

Figure 9 Cross Fold validation Random Forest

Dataset Preference analysis

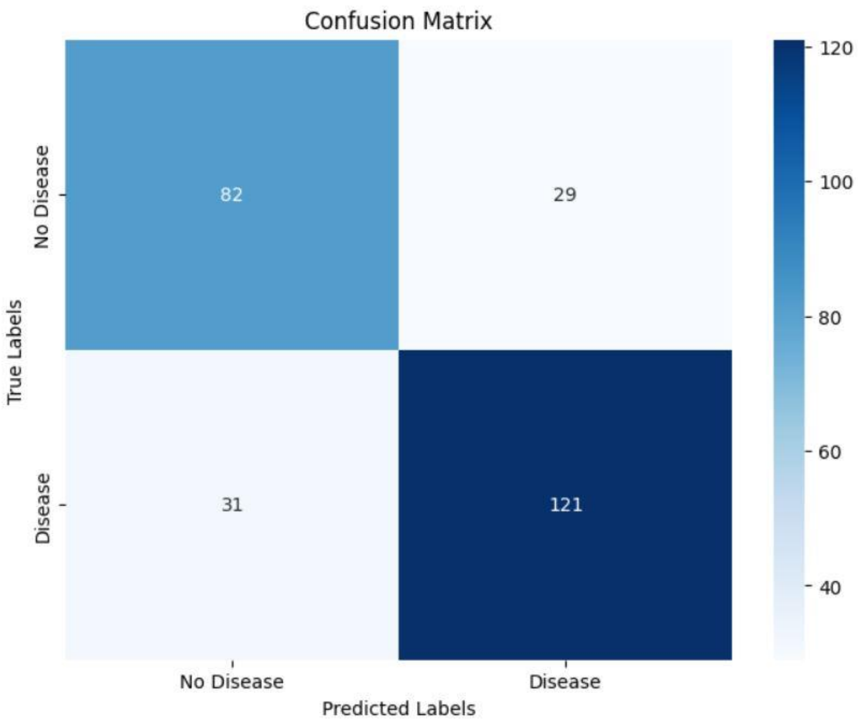
04). Dataset performance

```
[151] #Perfomences of the SVM
      y_pred_svm = svm_model.predict(X_test)
      accuracy_svm = accuracy_score(y_test, y_pred_svm)
      print("SVM Accuracy:", round(accuracy_svm, 3) )
      precision_svm = precision_score(y_test, y_pred_svm)
      print("SVM Precision:", round(precision_svm, 3))
      recall_svm = recall_score(y_test, y_pred_svm)
      print("SVM recall:", round(recall_svm, 3))
      f1_svm = f1_score(y_test, y_pred_svm)
      print("SVM F1 Score:", round(f1_svm, 3))
```

```
SVM Accuracy: 0.772
SVM Precision: 0.807
SVM recall: 0.796
SVM F1 Score: 0.801
```

[152]
```python
#svm confusion matrix using seaboan
confusion_matrix_svm = confusion_matrix(y_test, y_pred_svm)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_svm, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

[152]

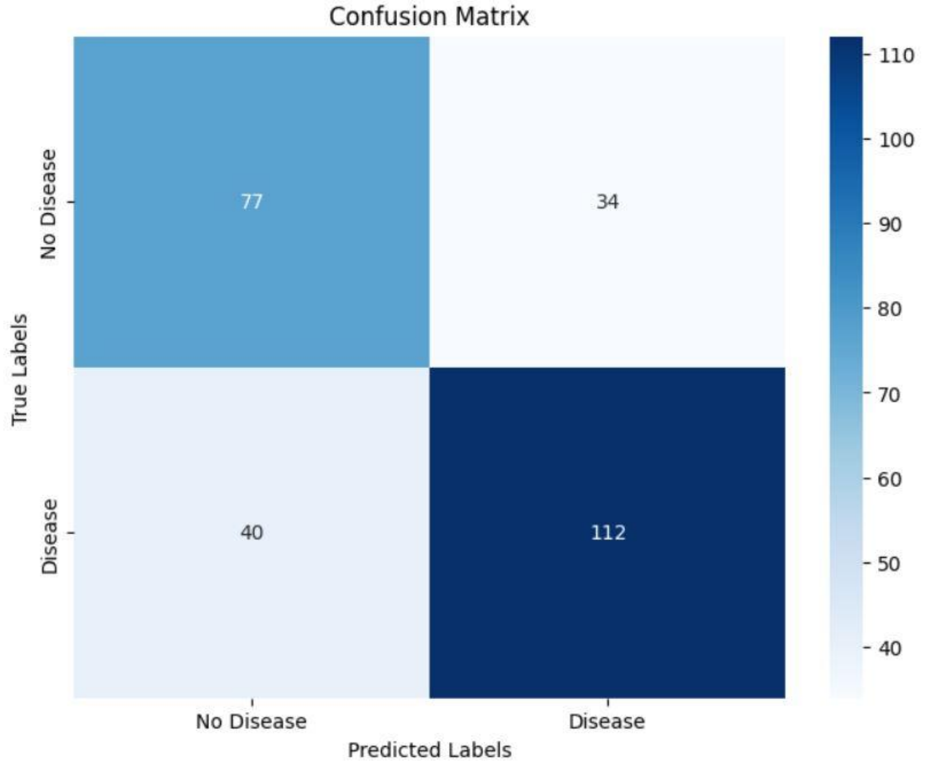


[153]
```python
#Perfomences of the knn
y_pred_knn = knn_model.predict(X_test)
accuracy_knn = accuracy_score(y_test, y_pred_knn)
print("KNN Accuracy:", round(accuracy_knn,3))
precision_knn = precision_score(y_test, y_pred_knn)
print("KNN Precision:", round(precision_knn, 3))
recall_knn = recall_score(y_test, y_pred_knn)
print("KNN recall:", round(recall_knn, 3))
f1_knn = f1_score(y_test, y_pred_knn)
print("KNN F1 Score:", round(f1_svm, 3))
```

```
KNN Accuracy: 0.719
KNN Precision: 0.767
KNN recall: 0.737
KNN F1 Score: 0.801
```

[154]
```python
#knn confusion matrix using seaboan
confusion_matrix_knn = confusion_matrix(y_test, y_pred_knn)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_knn, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

[154]




```
[155] #Perfomences of the RF
y_pred_rf = RandamForest_model.predict(X_test)
accuracy_rf = accuracy_score(y_test, y_pred_rf)
print("RandamForest Accuracy:", round(accuracy_rf, 3))
precision_rf = precision_score(y_test, y_pred_rf)
print("RandamForest Precision:", round(precision_rf, 3))
recall_rf = recall_score(y_test, y_pred_rf)
print("RandamForest recall:", round(recall_rf,3))
f1_rf = f1_score(y_test, y_pred_rf)
print("RandamForest F1 Score:", round(f1_rf, 3))
```

```
RandamForest Accuracy: 0.886
RandamForest Precision: 0.886
RandamForest recall: 0.921
RandamForest F1 Score: 0.903
```

```
[156] #Randamforest confusion matrix using seaboan
confusion_matrix_rf = confusion_matrix(y_test, y_pred_rf)
plt.figure(figsize=(8, 6))
sns.heatmap(confusion_matrix_rf, annot=True, fmt='d', cmap='Blues', xticklabels=['No Disease', 'Disease'], yticklabels=['No Disease', 'Disease'])
plt.title("Confusion Matrix")
plt.xlabel("Predicted Labels")
plt.ylabel("True Labels")
plt.show()
```

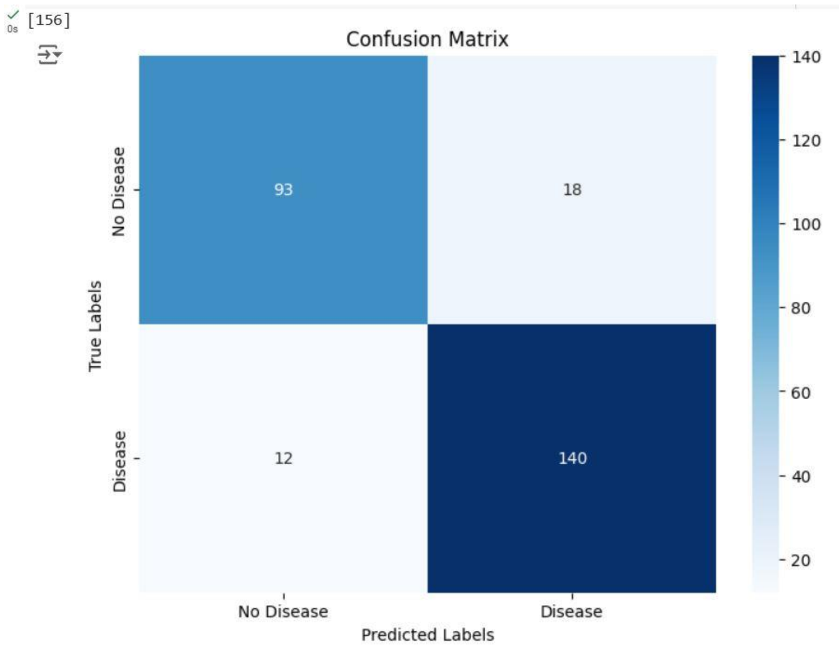
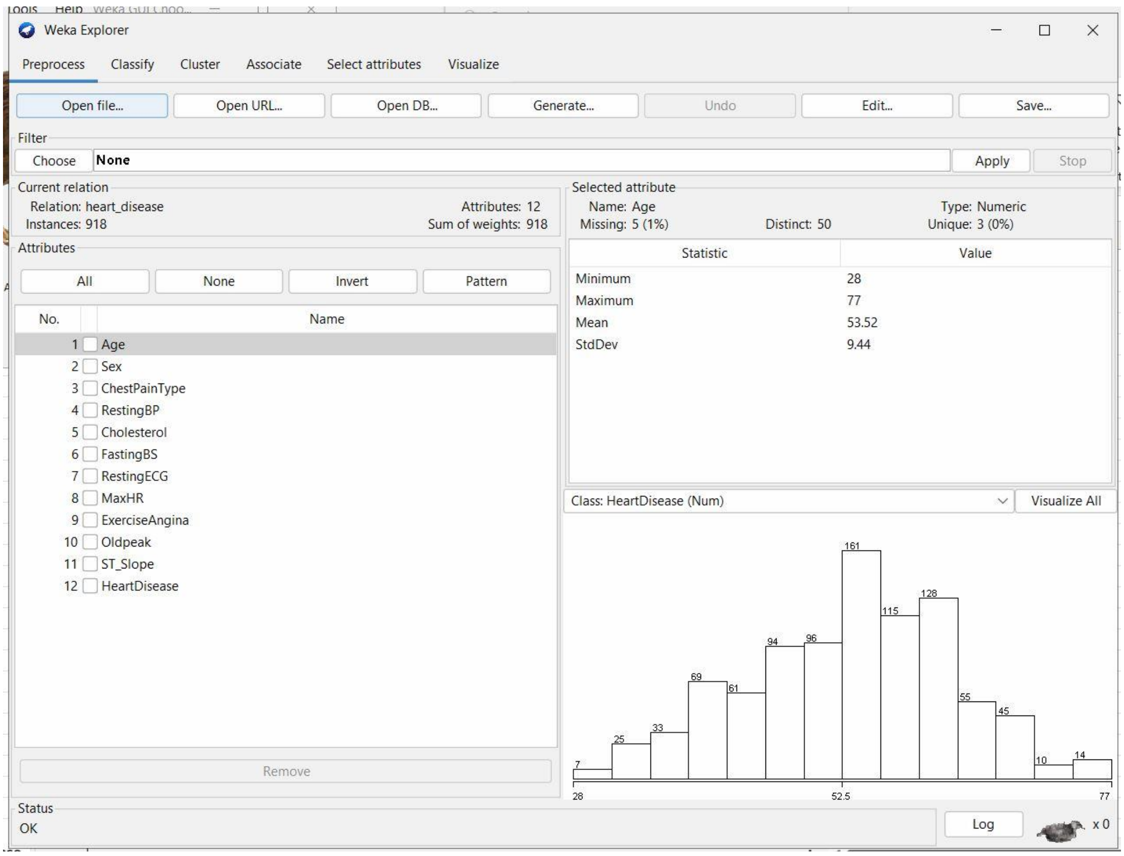


Figure 10 Model Performance analysis

Weka Tool

Import dataset



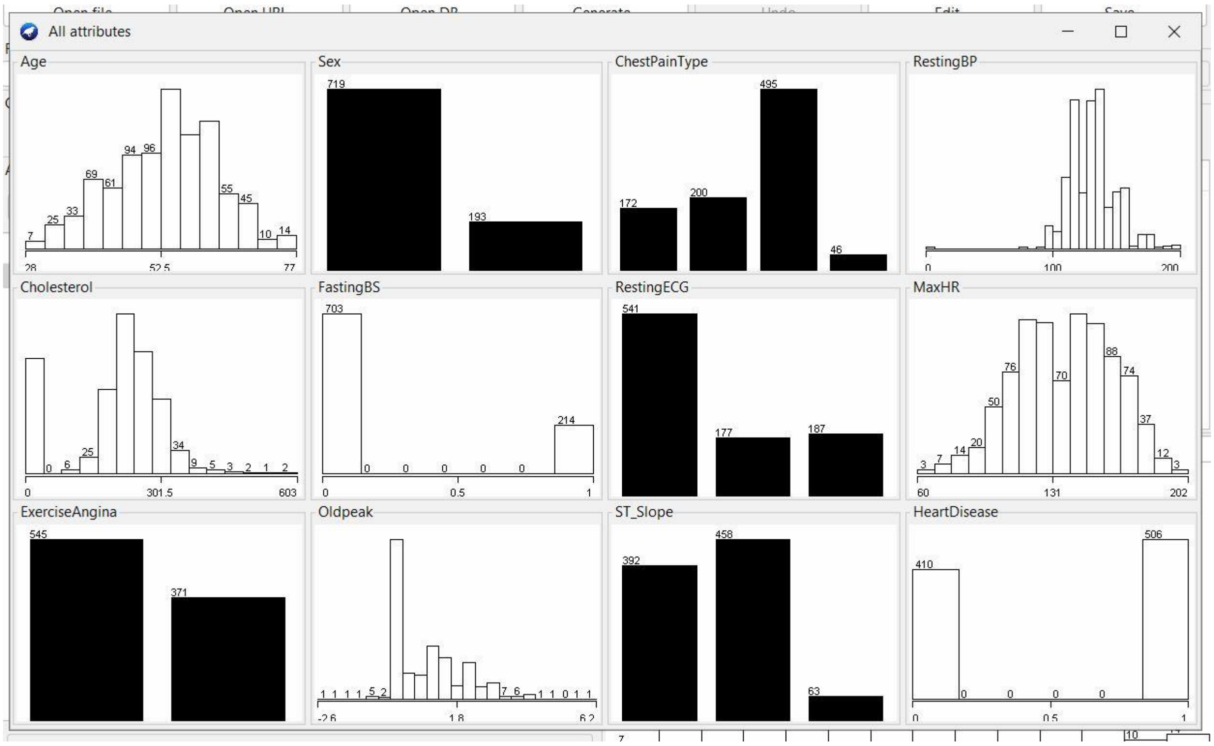


Figure 11 Import Dataset using Weka

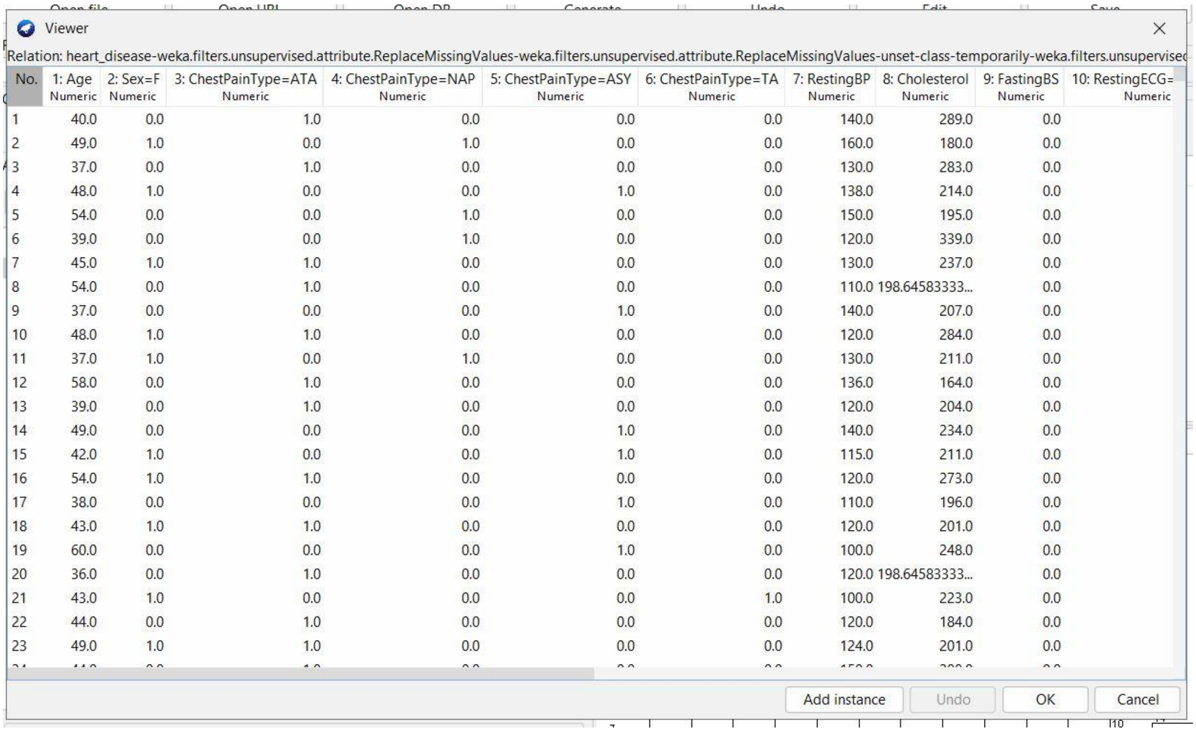
Remove Null values

Viewer

Relation: heart_disease

No.	1: Age Numeric	2: Sex Nominal	3: ChestPainType Nominal	4: RestingBP Numeric	5: Cholesterol Numeric	6: FastingBS Numeric	7: RestingECG Nominal	8: MaxHR Numeric	9: ExerciseAngina Nominal	10: Oldpeak Numeric	11: ST_Slope Nominal	12: **HeartDisease** Numeric
1	40.0	M	ATA	140.0	289.0	0.0	Normal	172.0	N	0.0	Up	0.0
2	49.0	F	NAP	160.0	180.0	0.0	Normal	156.0	N	1.0	Flat	1.0
3	37.0	M	ATA	130.0	283.0	0.0	ST	98.0	N	0.0	Up	0.0
4	48.0	F		138.0	214.0	0.0	Normal	108.0	Y	1.5	Flat	1.0
5	54.0	M	NAP	150.0	195.0	0.0	Normal	122.0	N	0.0	Up	0.0
6	39.0	M	NAP	120.0	339.0	0.0	Normal	170.0	N	0.0	Up	0.0
7	45.0	F	ATA	130.0	237.0	0.0	Normal	170.0	N	0.0	Up	0.0
8	54.0	M	ATA	110.0		0.0	Normal	142.0	N	0.0	Up	0.0
9	37.0	M	ASY	140.0	207.0	0.0	Normal	130.0	Y	1.5	Flat	1.0
10	48.0	F	ATA	120.0	284.0	0.0	Normal	120.0	N	0.0	Up	0.0
11	37.0	F	NAP	130.0	211.0	0.0	Normal	142.0	N	0.0	Up	0.0
12	58.0	M	ATA	136.0	164.0	0.0	ST	99.0	Y	2.0	Flat	1.0
13	39.0	M	ATA	120.0	204.0	0.0	Normal	145.0	N	0.0	Up	0.0
14	49.0	M	ASY	140.0	234.0	0.0	Normal	140.0	Y	1.0	Flat	1.0
15	42.0	F		115.0	211.0	0.0	ST	137.0	N	0.0	Up	0.0
16	54.0	F	ATA	120.0	273.0	0.0		150.0		1.5	Flat	0.0
17	38.0	M	ASY	110.0	196.0	0.0	Normal	166.0	N	0.0	Flat	1.0
18	43.0	F	ATA	120.0	201.0	0.0	Normal	165.0	N	0.0	Up	0.0
19	60.0	M	ASY	100.0	248.0	0.0	Normal	125.0	N	1.0	Flat	1.0
20	36.0	M	ATA	120.0		0.0	Normal	160.0	N	3.0	Flat	1.0
21	43.0	F	TA	100.0	223.0	0.0	Normal	142.0	N	0.0	Up	0.0
22	44.0	M	ATA	120.0	184.0	0.0	Normal	142.0	N	1.0	Flat	0.0
23	49.0	F	ATA	124.0	201.0	0.0	Normal	164.0	N	0.0	Up	0.0
24	44.0	M	ATA	150.0	288.0	0.0	Normal	150.0	Y	3.0	Flat	1.0

Add instance | Undo | OK | Cancel

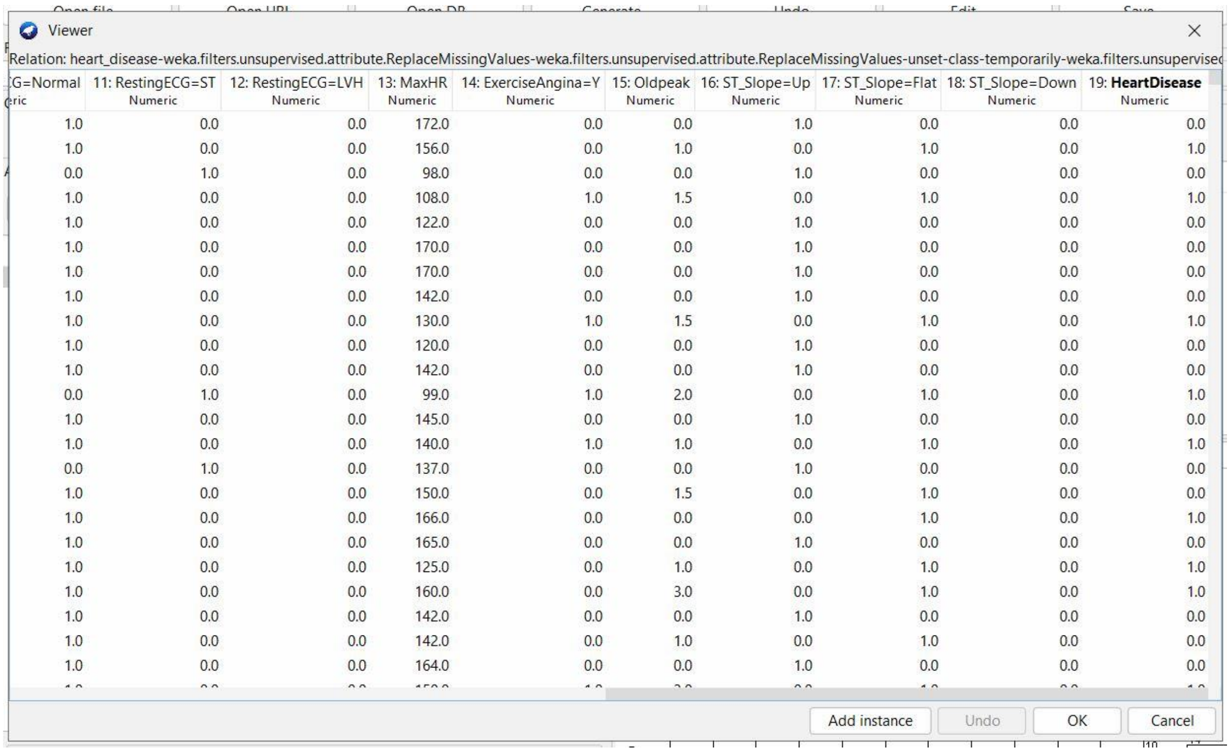


Viewer

Relation: heart_disease-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.ReplaceMissingValues-unset-class-temporarily-weka.filters.unsupervised

No.	1: Age Numeric	2: Sex=F Numeric	3: ChestPainType=ATA Numeric	4: ChestPainType=NAP Numeric	5: ChestPainType=ASY Numeric	6: ChestPainType=TA Numeric	7: RestingBP Numeric	8: Cholesterol Numeric	9: FastingBS Numeric	10: RestingECG= Numeric
1	40.0	0.0	1.0	0.0	0.0	0.0	140.0	289.0	0.0	
2	49.0	1.0	0.0	1.0	0.0	0.0	160.0	180.0	0.0	
3	37.0	0.0	1.0	0.0	0.0	0.0	130.0	283.0	0.0	
4	48.0	1.0	0.0	0.0	1.0	0.0	138.0	214.0	0.0	
5	54.0	0.0	0.0	1.0	0.0	0.0	150.0	195.0	0.0	
6	39.0	0.0	0.0	1.0	0.0	0.0	120.0	339.0	0.0	
7	45.0	1.0	1.0	0.0	0.0	0.0	130.0	237.0	0.0	
8	54.0	0.0	1.0	0.0	0.0	0.0	110.0	198.64583333...	0.0	
9	37.0	0.0	0.0	0.0	1.0	0.0	140.0	207.0	0.0	
10	48.0	1.0	1.0	0.0	0.0	0.0	120.0	284.0	0.0	
11	37.0	1.0	0.0	1.0	0.0	0.0	130.0	211.0	0.0	
12	58.0	0.0	1.0	0.0	0.0	0.0	136.0	164.0	0.0	
13	39.0	0.0	1.0	0.0	0.0	0.0	120.0	204.0	0.0	
14	49.0	0.0	0.0	0.0	1.0	0.0	140.0	234.0	0.0	
15	42.0	1.0	0.0	0.0	1.0	0.0	115.0	211.0	0.0	
16	54.0	1.0	1.0	0.0	0.0	0.0	120.0	273.0	0.0	
17	38.0	0.0	0.0	0.0	1.0	0.0	110.0	196.0	0.0	
18	43.0	1.0	1.0	0.0	0.0	0.0	120.0	201.0	0.0	
19	60.0	0.0	0.0	0.0	1.0	0.0	100.0	248.0	0.0	
20	36.0	0.0	1.0	0.0	0.0	0.0	120.0	198.64583333...	0.0	
21	43.0	1.0	0.0	0.0	0.0	1.0	100.0	223.0	0.0	
22	44.0	0.0	1.0	0.0	0.0	0.0	120.0	184.0	0.0	
23	49.0	1.0	1.0	0.0	0.0	0.0	124.0	201.0	0.0	

Add instance Undo OK Cancel



Viewer

Relation: heart_disease-weka.filters.unsupervised.attribute.ReplaceMissingValues-weka.filters.unsupervised.attribute.ReplaceMissingValues-unset-class-temporarily-weka.filters.unsupervised

G=Normal eric	11: RestingECG=ST Numeric	12: RestingECG=LVH Numeric	13: MaxHR Numeric	14: ExerciseAngina=Y Numeric	15: Oldpeak Numeric	16: ST_Slope=Up Numeric	17: ST_Slope=Flat Numeric	18: ST_Slope=Down Numeric	19: **HeartDisease** Numeric
1.0	0.0	0.0	172.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	156.0	0.0	1.0	0.0	1.0	0.0	1.0
0.0	1.0	0.0	98.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	108.0	1.0	1.5	0.0	1.0	0.0	1.0
1.0	0.0	0.0	122.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	170.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	170.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	142.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	130.0	1.0	1.5	0.0	1.0	0.0	1.0
1.0	0.0	0.0	120.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	142.0	0.0	0.0	1.0	0.0	0.0	0.0
0.0	1.0	0.0	99.0	1.0	2.0	0.0	1.0	0.0	1.0
1.0	0.0	0.0	145.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	140.0	1.0	1.0	0.0	1.0	0.0	1.0
0.0	1.0	0.0	137.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	150.0	0.0	1.5	0.0	1.0	0.0	0.0
1.0	0.0	0.0	166.0	0.0	0.0	0.0	1.0	0.0	1.0
1.0	0.0	0.0	165.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	125.0	0.0	1.0	0.0	1.0	0.0	1.0
1.0	0.0	0.0	160.0	0.0	3.0	0.0	1.0	0.0	1.0
1.0	0.0	0.0	142.0	0.0	0.0	1.0	0.0	0.0	0.0
1.0	0.0	0.0	142.0	0.0	1.0	0.0	1.0	0.0	0.0
1.0	0.0	0.0	164.0	0.0	0.0	1.0	0.0	0.0	0.0

Add instance Undo OK Cancel

Figure 12 Remove Null Values

SVM 10-cross fold

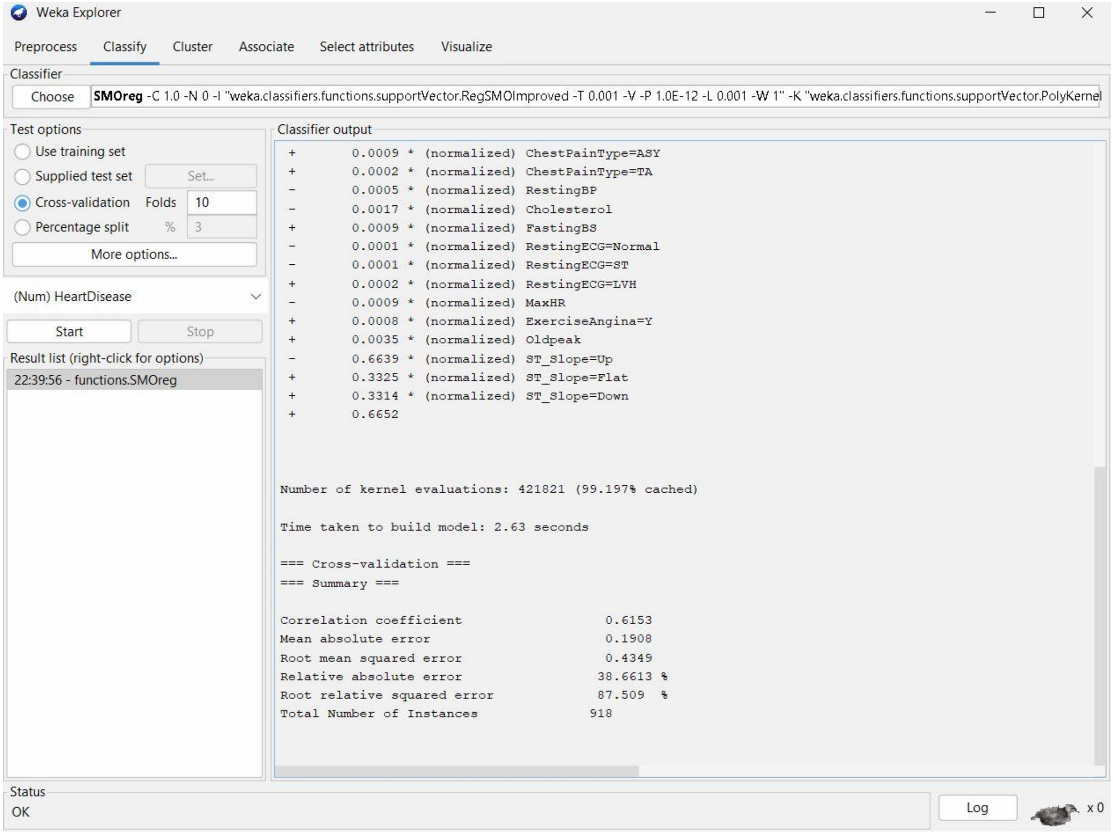


Figure 13 SVM Cross Fold

SVM 3% split data

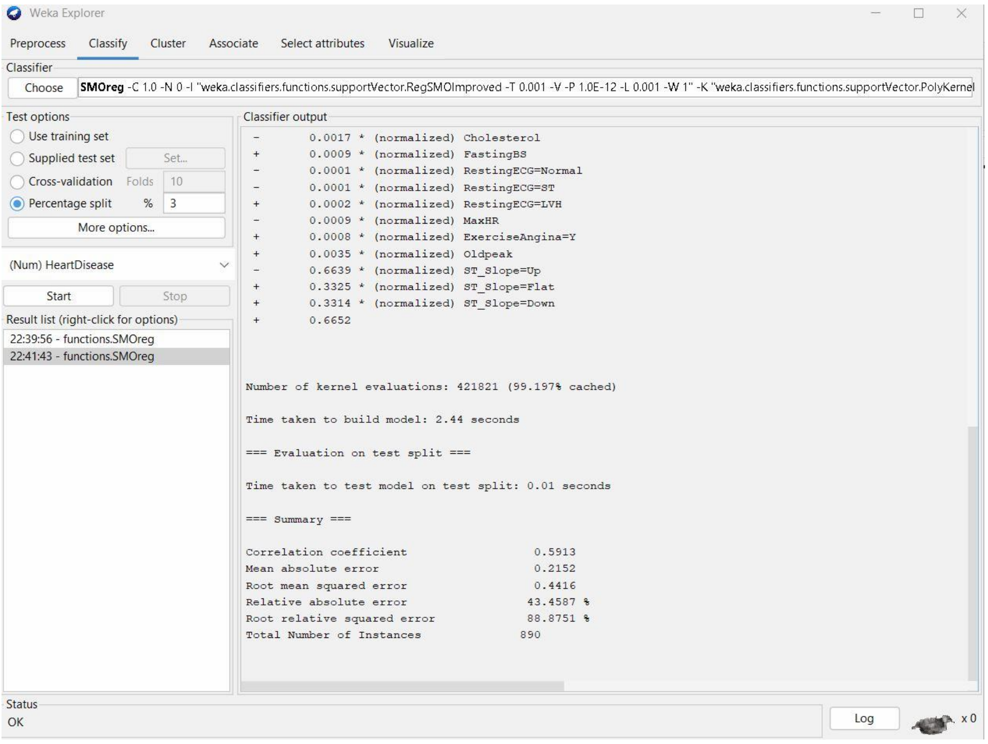


Figure 14 SVM 3% Split data

KNN 3% split data

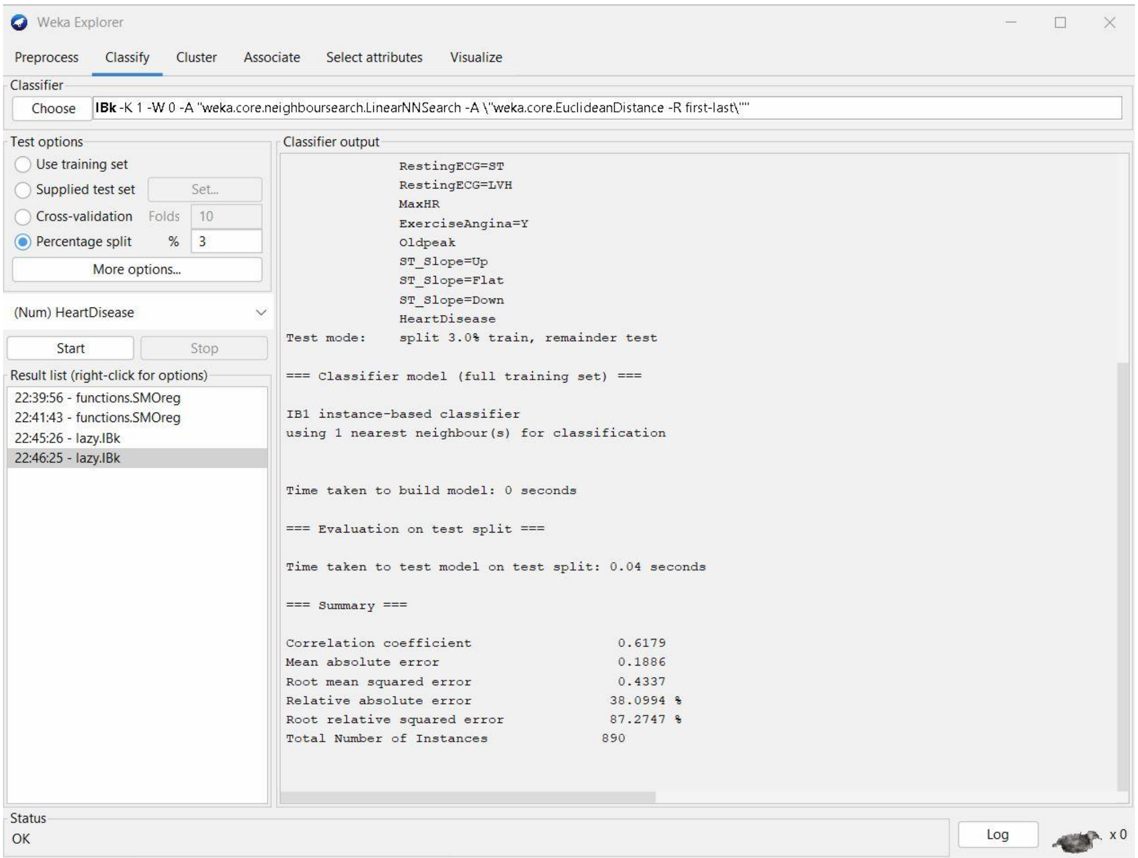


Figure 15 KNN 3% Split Data

KNN 10-cross fold

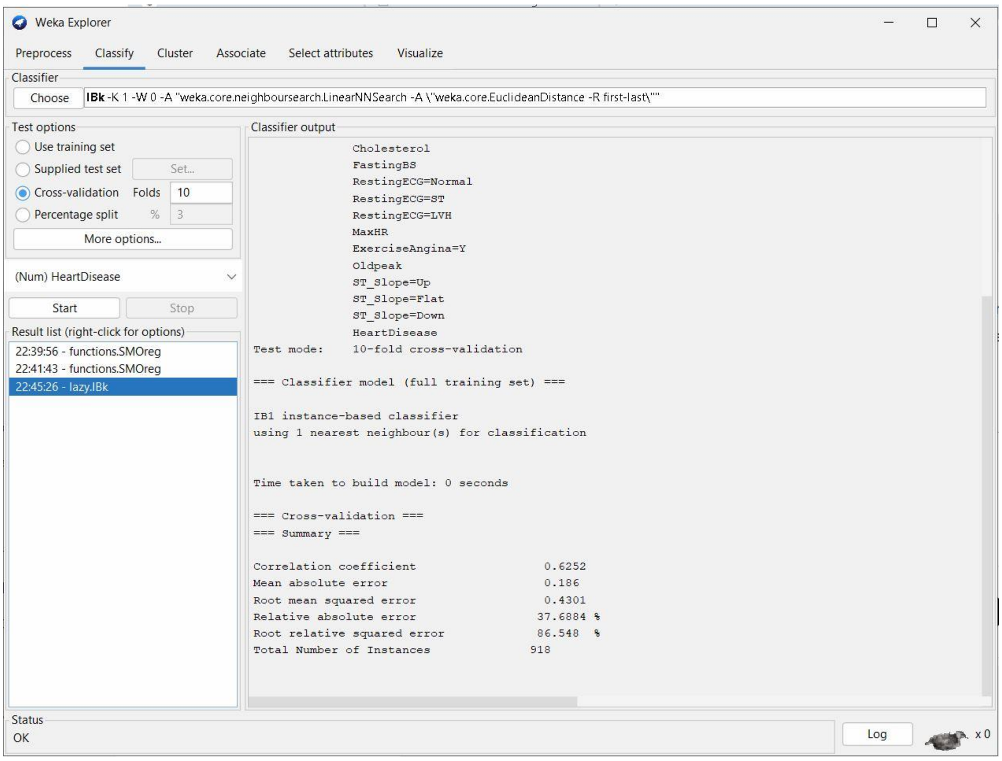


Figure 16 KNN 10-Cross fold

RandamForest 10-cross fold

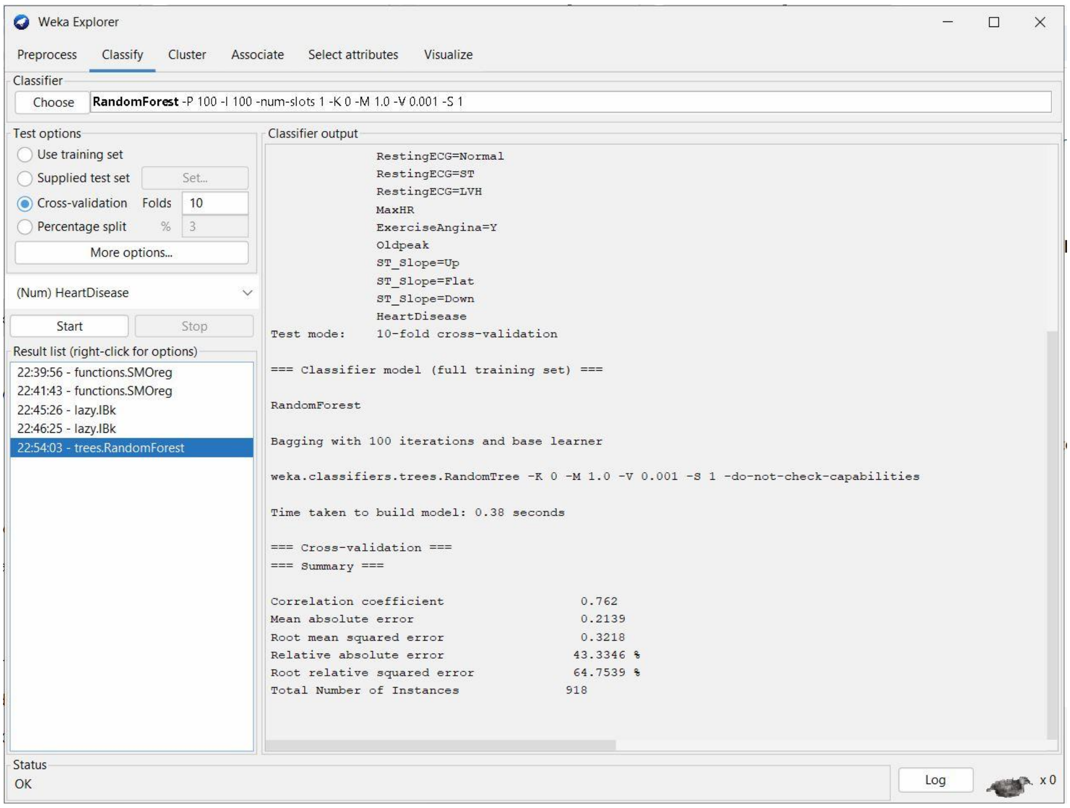


Figure 17 Randa Forest 10-cross fold

RandamForest 3% split data

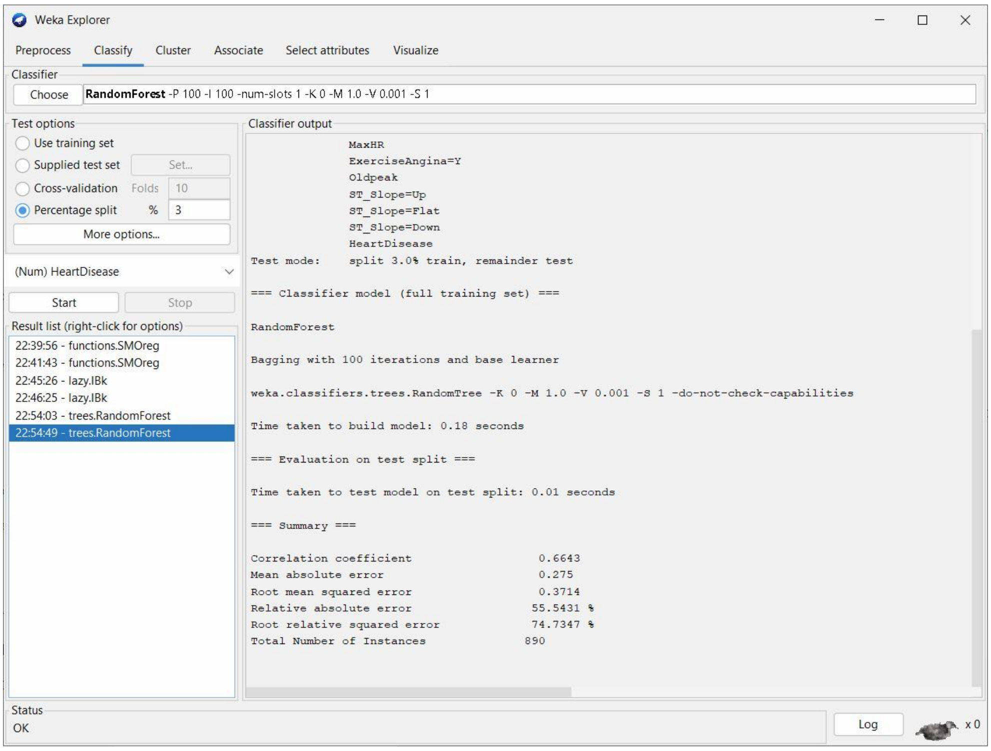
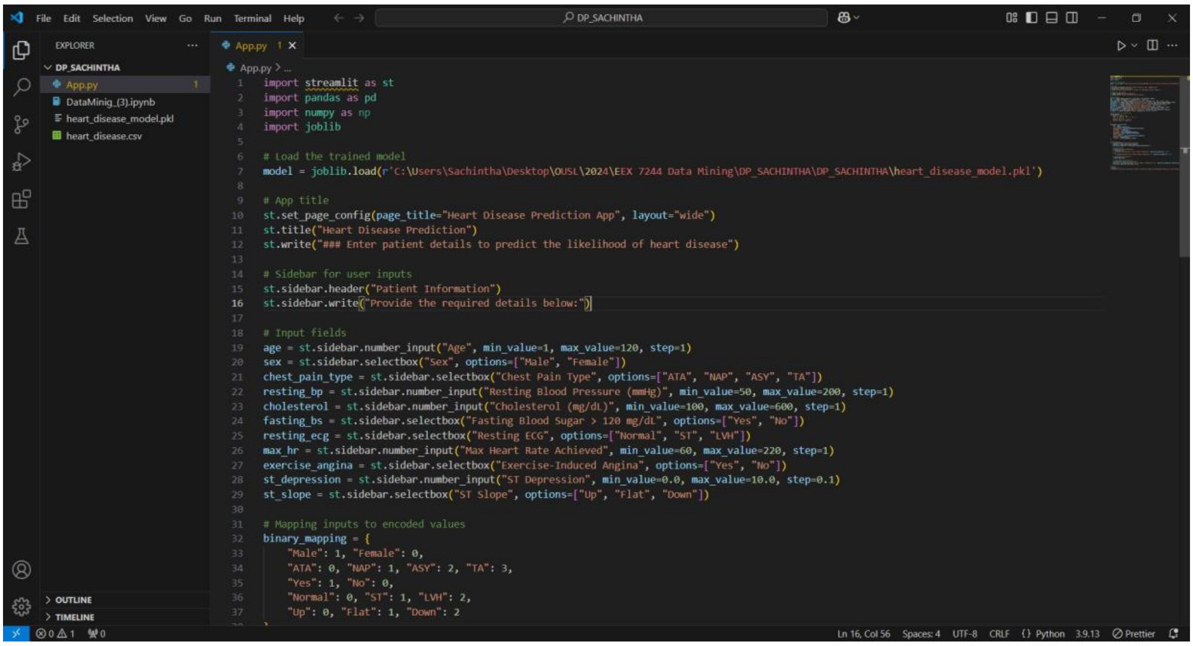
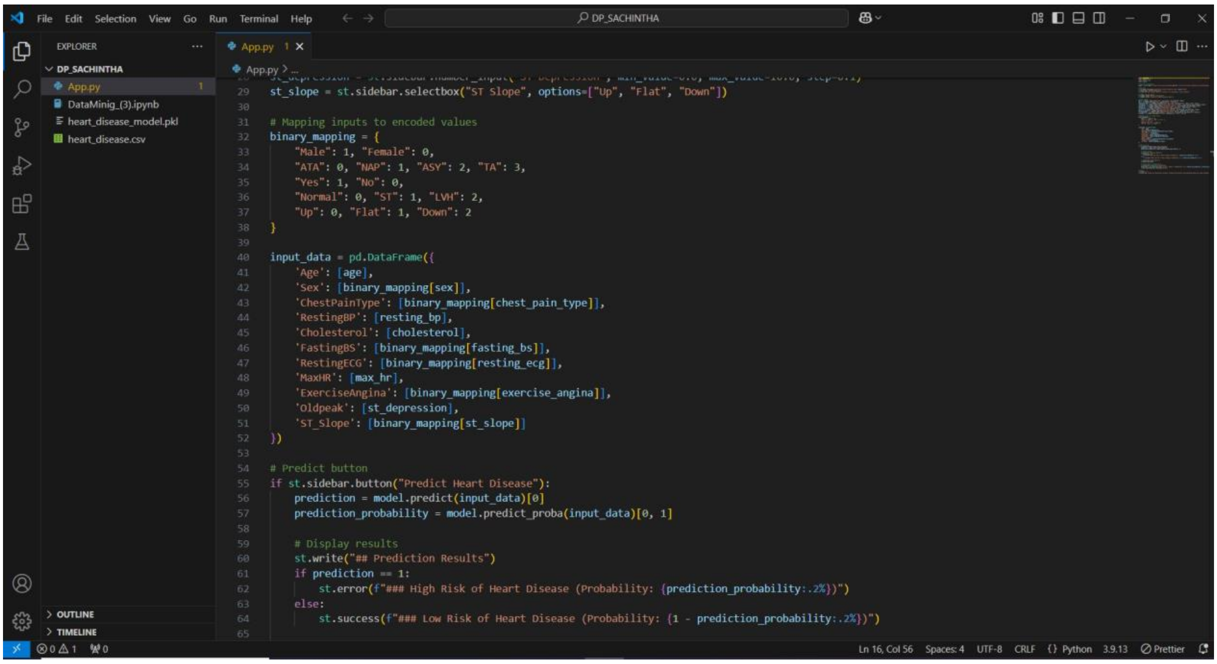


Figure 18 Random Forest 3% split data

Frontend



```python
import streamlit as st
import pandas as pd
import numpy as np
import joblib

# Load the trained model
model = joblib.load(r'C:\Users\Sachintha\Desktop\OUSL\2024\EEX 7244 Data Mining\DP_SACHINTHA\DP_SACHINTHA\heart_disease_model.pkl')

# App title
st.set_page_config(page_title="Heart Disease Prediction App", layout="wide")
st.title("Heart Disease Prediction")
st.write("### Enter patient details to predict the likelihood of heart disease")

# Sidebar for user inputs
st.sidebar.header("Patient Information")
st.sidebar.write("Provide the required details below:")

# Input fields
age = st.sidebar.number_input("Age", min_value=1, max_value=120, step=1)
sex = st.sidebar.selectbox("Sex", options=["Male", "Female"])
chest_pain_type = st.sidebar.selectbox("Chest Pain Type", options=["ATA", "NAP", "ASY", "TA"])
resting_bp = st.sidebar.number_input("Resting Blood Pressure (mmHg)", min_value=50, max_value=200, step=1)
cholesterol = st.sidebar.number_input("Cholesterol (mg/dL)", min_value=100, max_value=600, step=1)
fasting_bs = st.sidebar.selectbox("Fasting Blood Sugar > 120 mg/dL", options=["Yes", "No"])
resting_ecg = st.sidebar.selectbox("Resting ECG", options=["Normal", "ST", "LVH"])
max_hr = st.sidebar.number_input("Max Heart Rate Achieved", min_value=60, max_value=220, step=1)
exercise_angina = st.sidebar.selectbox("Exercise-Induced Angina", options=["Yes", "No"])
st_depression = st.sidebar.number_input("ST Depression", min_value=0.0, max_value=10.0, step=0.1)
st_slope = st.sidebar.selectbox("ST Slope", options=["Up", "Flat", "Down"])

# Mapping inputs to encoded values
binary_mapping = {
    "Male": 1, "Female": 0,
    "ATA": 0, "NAP": 1, "ASY": 2, "TA": 3,
    "Yes": 1, "No": 0,
    "Normal": 0, "ST": 1, "LVH": 2,
    "Up": 0, "Flat": 1, "Down": 2
```



```python
st_slope = st.sidebar.selectbox("ST Slope", options=["Up", "Flat", "Down"])

# Mapping inputs to encoded values
binary_mapping = {
    "Male": 1, "Female": 0,
    "ATA": 0, "NAP": 1, "ASY": 2, "TA": 3,
    "Yes": 1, "No": 0,
    "Normal": 0, "ST": 1, "LVH": 2,
    "Up": 0, "Flat": 1, "Down": 2
}

input_data = pd.DataFrame({
    'Age': [age],
    'Sex': [binary_mapping[sex]],
    'ChestPainType': [binary_mapping[chest_pain_type]],
    'RestingBP': [resting_bp],
    'Cholesterol': [cholesterol],
    'FastingBS': [binary_mapping[fasting_bs]],
    'RestingECG': [binary_mapping[resting_ecg]],
    'MaxHR': [max_hr],
    'ExerciseAngina': [binary_mapping[exercise_angina]],
    'Oldpeak': [st_depression],
    'ST_Slope': [binary_mapping[st_slope]]
})

# Predict button
if st.sidebar.button("Predict Heart Disease"):
    prediction = model.predict(input_data)[0]
    prediction_probability = model.predict_proba(input_data)[0, 1]

    # Display results
    st.write("## Prediction Results")
    if prediction == 1:
        st.error(f"### High Risk of Heart Disease (Probability: {prediction_probability:.2%})")
    else:
        st.success(f"### Low Risk of Heart Disease (Probability: {1 - prediction_probability:.2%})")
```

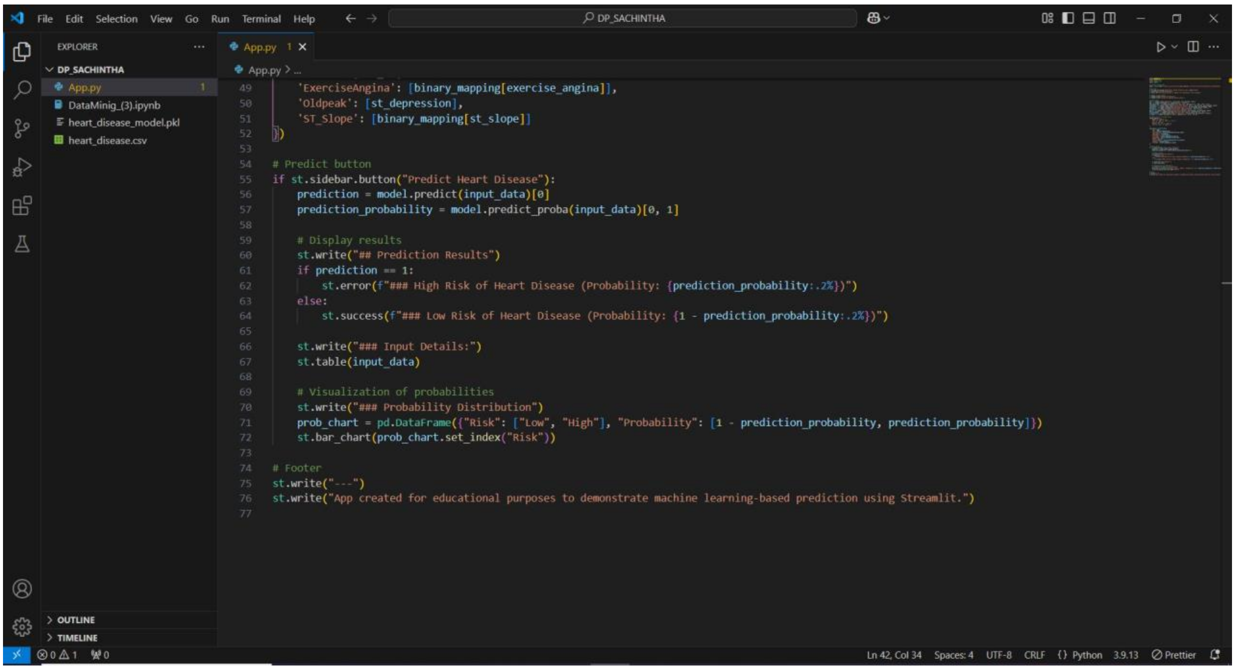


Figure 19 Front end Code

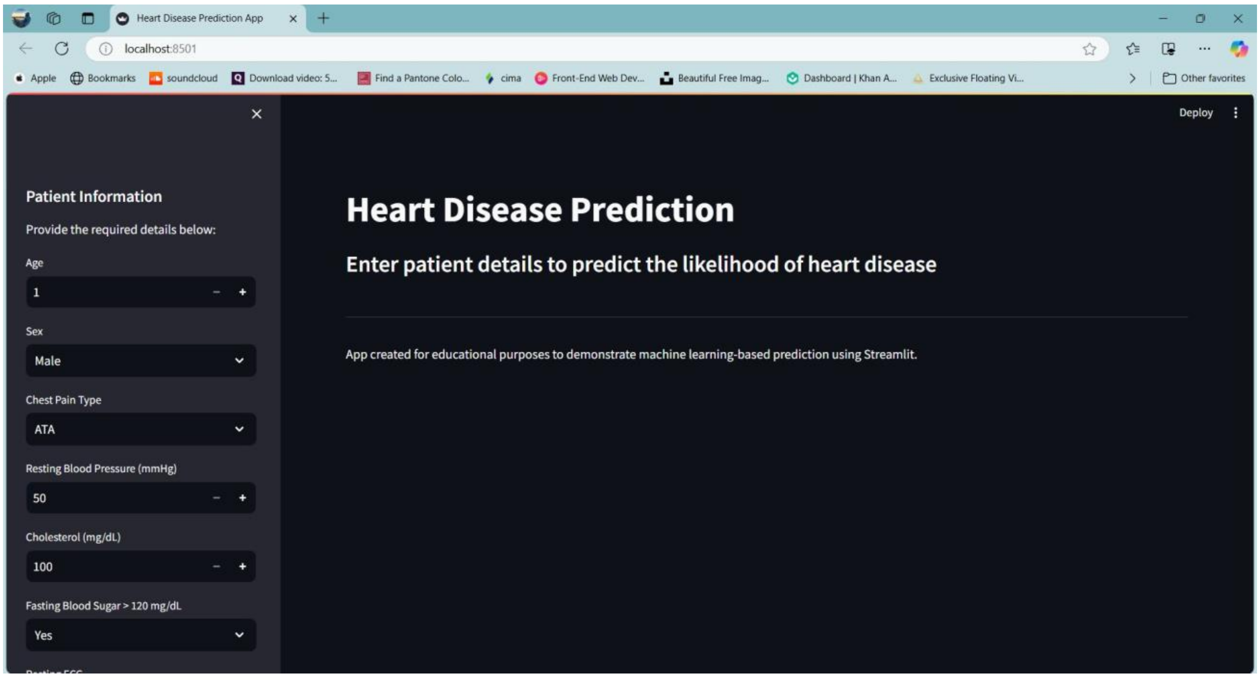


Figure 20 Front end

Test results

Python code

3% split

```
[ ]  # results Table for SVM, KNN, and Random Forest as follows
     results = {
         "Model": ["SVM", "KNN", "Random Forest"],
         "Accuracy": [accuracy_svm , accuracy_knn, accuracy_rf],
         "Precision": [precision_svm , precision_knn, precision_rf],
         "Recall": [recall_svm , recall_knn, recall_rf],
         "F1-Score": [f1_svm, f1_knn, f1_rf],
     }

     # Create a dataframe
     comparison_df = pd.DataFrame(results)

     # Print the table
     print(comparison_df)
```

```
           Model  Accuracy  Precision    Recall  F1-Score
0            SVM  0.771863   0.806667  0.796053  0.801325
1            KNN  0.718631   0.767123  0.736842  0.751678
2  Random Forest  0.885932   0.886076  0.921053  0.903226
```

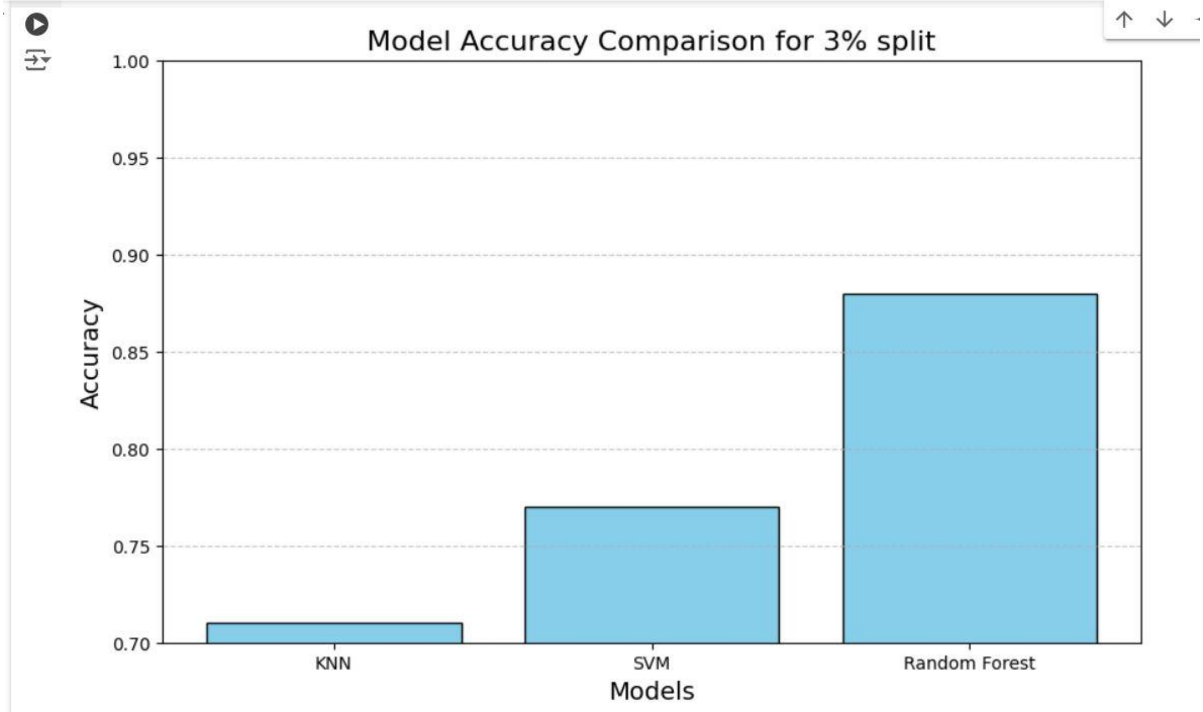
Figure 21 3% split Test Results

10- cross fold

```
[163] # The results for SVM, KNN, and Random Forest in 10-fold cross validation
      results_10fold = {
          "Model": ["SVM", "KNN", "Random Forest"],
          "Mean Accuracy": [np.mean(cv_scores_svm) , np.mean(cv_scores_knn), np.mean(cv_scores_rf)],
          "Standard Deviation": [np.std(cv_scores_svm) , np.std(cv_scores_knn), np.std(cv_scores_rf)],
      }

      # Create a dataframe
      comparison_df = pd.DataFrame(results_10fold)

      # Print the table
      print(comparison_df)
```

```
           Model  Mean Accuracy  Standard Deviation
0            SVM       0.717581            0.053482
1            KNN       0.698171            0.042791
2  Random Forest       0.864002            0.020042
```

Figure 22 10-Cross fold Test Results



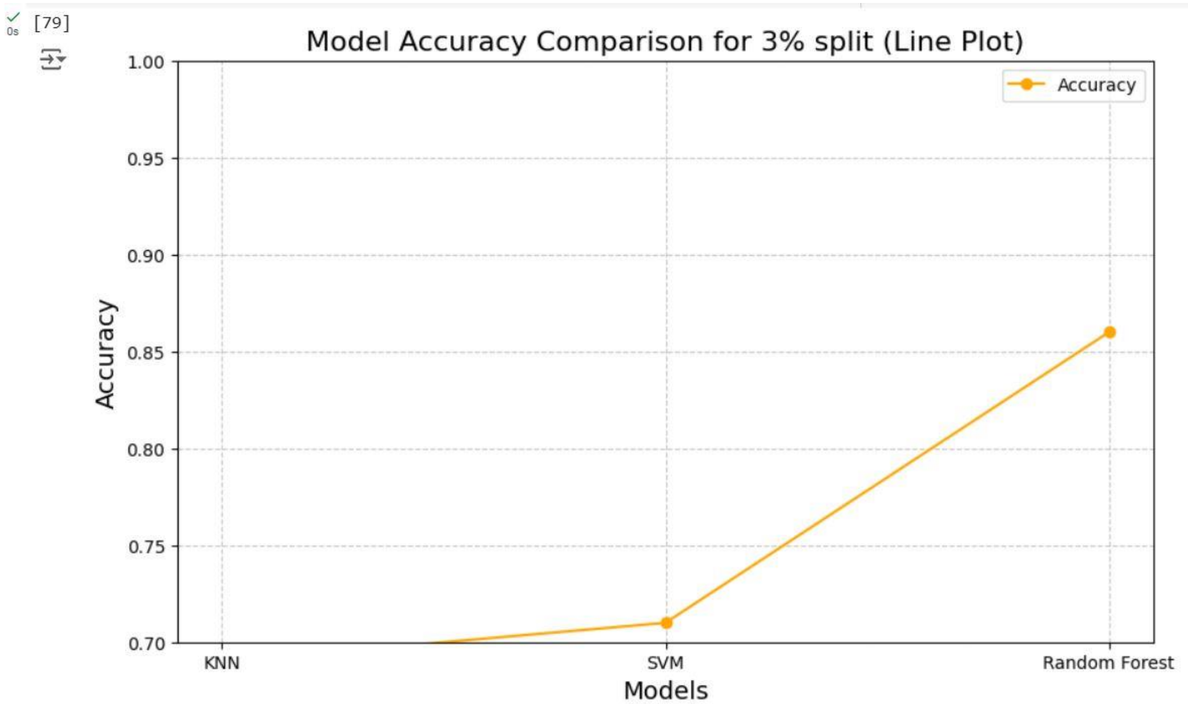
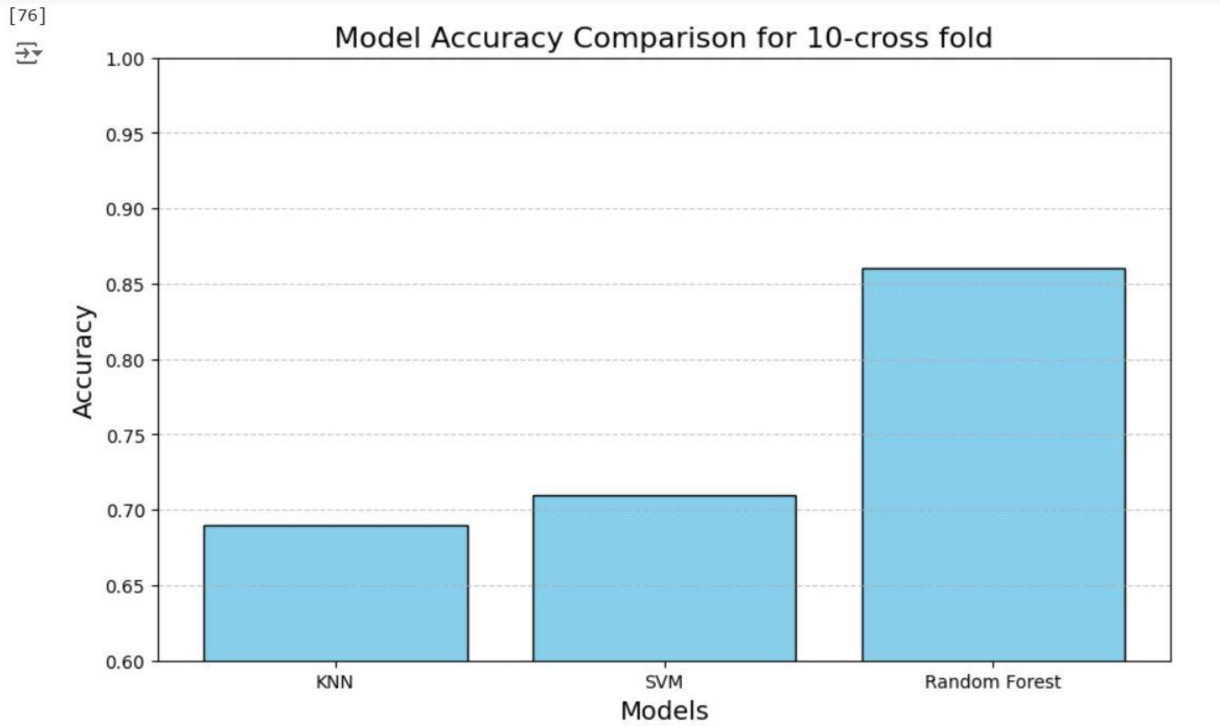


Figure 23 3% Split Accuracy Comparison



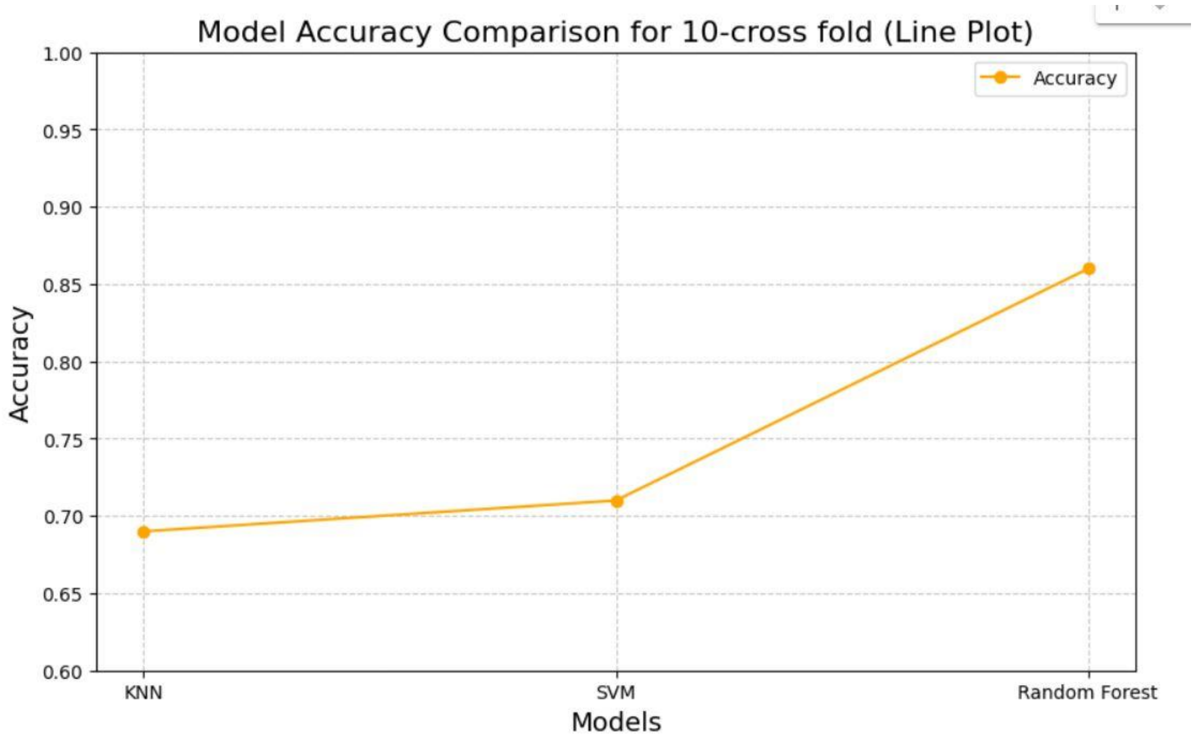


Figure 24 10-Cross fold accuracy Comparison

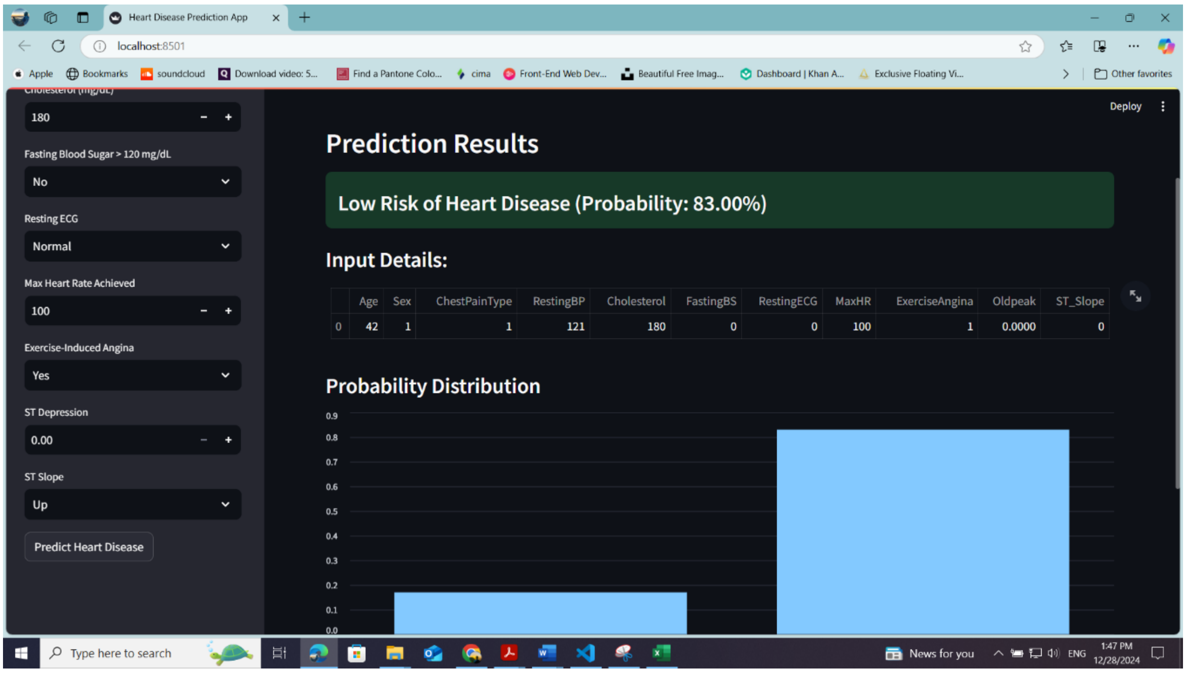


Figure 25 Result at Front end Positive value check

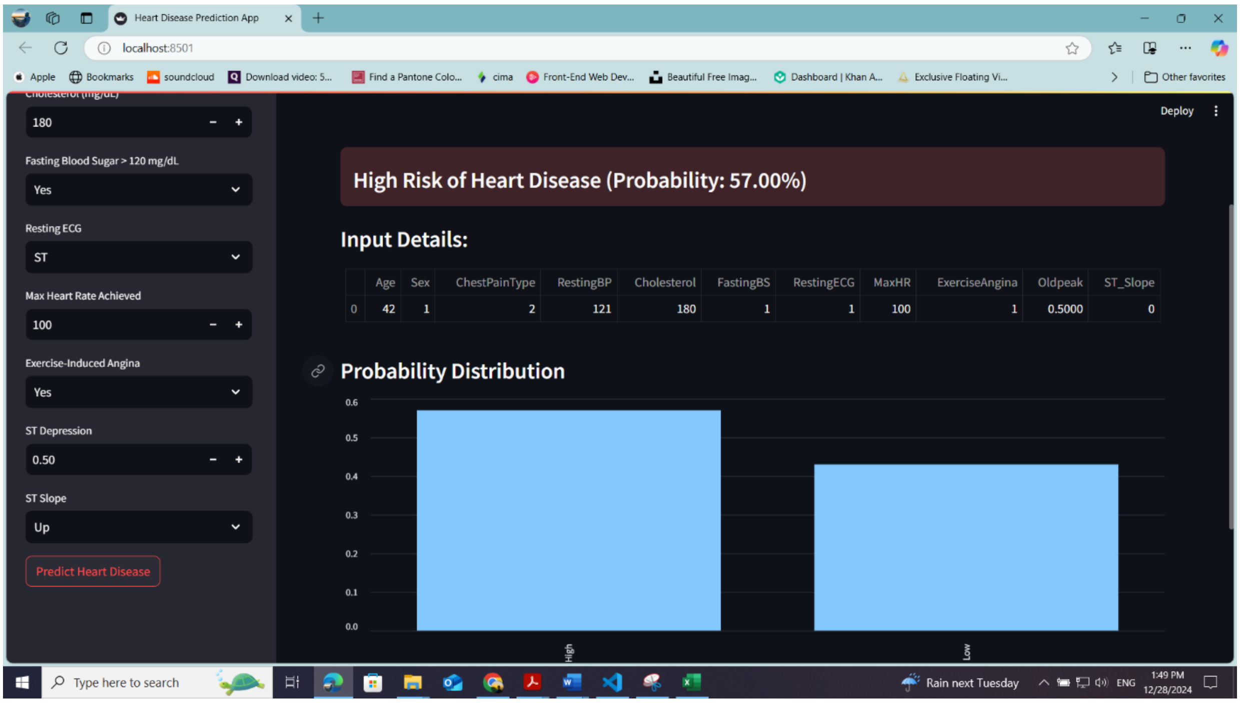


Figure 26 Result at Front end Negative Value check

Discussion

The primary objective of this analysis is to predict heart disease using machine learning models, evaluating their performance across two strategies: a 3% split validation and 10-fold cross-validation. Three models—Support Vector Machine (SVM), K-Nearest Neighbors (KNN), and Random Forest—were assessed based on metrics such as accuracy, precision, recall, and F1-score.

In the 3% split validation, the dataset was divided into 97% training and 3% testing data. Among the models, Random Forest achieved the highest accuracy (88.6%), precision (88.6%), recall (92.1%), and F1-score (90.3%), demonstrating its ability to effectively capture complex relationships in the data. SVM followed with balanced metrics, including an accuracy of 77.2% and an F1-score of 80.1%, indicating consistent but slightly lower performance. KNN, while achieving an accuracy of 71.9% and an F1-score of 75.2%, struggled to match the performance of the other models, likely due to its sensitivity to noise and the small test set.

In the 10-fold cross-validation, which offers a more robust evaluation by splitting the data into ten subsets, Random Forest once again outperformed the other models with a mean accuracy of 86.4% and a low standard deviation of 2.0%, signifying consistent and reliable performance across folds. SVM showed a mean accuracy of 71.8% with a standard deviation of 5.3%, slightly lower than its split validation performance, indicating sensitivity to variations in the dataset. KNN maintained similar performance with a mean accuracy of 69.8% and a standard deviation of 4.3%, confirming its limitations in handling complex patterns in the data.

Overall, Random Forest emerged as the most effective model for predicting heart disease, consistently demonstrating superior performance across both evaluation methods. Its ability to handle complex feature interactions and reduce overfitting makes it the best-suited model for this task.

While SVM provided balanced results, it fell short of Random Forest's accuracy and consistency. KNN, being simple and sensitive to data distribution, underperformed compared to the other models. Based on these findings, Random Forest is recommended as the primary model for heart disease prediction. For future work, hyperparameter tuning, advanced feature engineering, and combining models into an ensemble could further enhance predictive performance.

Finally, I used the Random Forest model with a 3% split to implement the results in the front end, as it provided the best results for the analysis.

References

[1]. Rairikar, V. Kulkarni, V. Sabale, H. Kale, and A. Lamgunde, "Heart disease prediction using data mining techniques," in *2017 International Conference on Intelligent Computing and Control (I2C2)*, June 2017, pp. 1-8.

[2]. M. S. Amin, Y. K. Chiam, and K. D. Varathan, "Identification of significant features and data mining techniques in predicting heart disease," *Telematics and Informatics*, vol. 36, pp. 82-93, 2019.

[3]. S. V. M. Vishwanathan and M. N. Murty, "SSVM: a simple SVM algorithm," in *Proceedings of the 2002 International Joint Conference on Neural Networks. IJCNN'02 (Cat. No. 02CH37290)*, vol. 3, May 2002, pp. 2393-2398.

[4]. B. Saleh, A. Saedi, A. Al-Aqbi, and L. Salman, "Analysis of Weka data mining techniques for heart disease prediction system," *International Journal of Medical Reviews*, vol. 7, no. 1, pp. 15-24, 2020.

[5]. V. Jakkula, "Tutorial on support vector machine (SVM)," *School of EECS, Washington State University*, vol. 37, no. 2.5, p. 3, 2006.

[6]. R. Chitra and V. Seenivasagam, "Review of heart disease prediction system using data mining and hybrid intelligent techniques," *ICTACT Journal on Soft Computing*, vol. 3, no. 4, pp. 605-609, 2013.

[7]. Huang, S., Cai, N., Pacheco, P.P., Narrandes, S., Wang, Y. and Xu, W., 2018. Applications of support vector machine (SVM) learning in cancer genomics. Cancer genomics & proteomics, 15(1), pp.41-51.

[8]. S. H. Shadid, A. Shafkat, M. F. Yasmeen, S. A. Sibli, and M. R. Rafi, "Prediction of heart disease using data mining techniques: A case study," *Prediction of Heart Disease Using Data Mining Techniques: A Case Study*, vol. 41, no. 1, pp. 11-11, 2019.

[9]. B. Kaur and W. Singh, "Review on heart disease prediction system using data mining techniques," *International Journal on Recent and Innovation Trends in Computing and Communication*, vol. 2, no. 10, pp. 3003-3008, 2014.