CIS 550-7102: Advanced Machine Learning - On-Line
(2024 - Spring CE1)

Final Project Report:
Predictive Modelling for Heart Disease Diagnosis

Group – 10

SUBMITTED BY:

Name: Tiyasa Saha
Student ID: 02113195
Roll No.: 47

Name: Nikhil Premachandrarao
Student ID: 02105149
Roll No.: 42

Name: Jeevan Kumar Banoth
Student ID: 02105145
Roll No.: 04

Project Overview:

In this project, we are analyzing data on heart disease to create models that can predict whether a patient has the condition or not. The dataset includes information such as the patient's age, sex, type of chest pain, blood pressure, cholesterol levels, and other medical details. Predictive modeling is crucial in healthcare for diagnosing diseases and helping doctors make informed decisions. Since heart disease is a major global cause of death, early detection and risk stratification are important.

The project proposes developing predictive models using machine learning algorithms to estimate the likely heart disease that would be present based on patient details and diagnostic test results.

Why this Dataset?

The Heart Disease Dataset is commonly used for creating and assessing machine learning models that can predict the presence or absence of heart disease. It consists of 303 instances and 76 attributes, but many studies concentrate on a subset of 14 key attributes such as age, sex, chest pain type, blood pressure, cholesterol levels, and other clinical features. The main objective is to develop precise predictive models that can determine whether a patient has heart disease by analyzing these medical attributes. This dataset is an important tool for researchers and professionals to experiment and validate different machine learning algorithms like decision trees and random forests for diagnosing heart disease.

There are several common uses of the given dataset:

First, it is medical relevance. Since heart disease is one of the most severe health conditions, its detection at an early stage is vital for the patient’s survival. For this reason, healthcare professionals can opt for creating an accurate predictive model to identify potential risk factors and assess the situation.

Second, the Heart Disease Data Set is a benchmark dataset. It is widely applied in the medical domain and is a well-known type of benchmark dataset. Therefore, researchers can compare the accuracy of its models with the existing approaches.

Finally, there are different types of features that can be compared through the implementation of various feature engineering and preprocessing. Due to its relatively small size, cross-validation can be arranged to determine the model’s performance and avoid overfitting.

Heart disease data prediction Implementation:

Data Preprocessing: -

The data was loaded from a CSV file, and exploratory data analysis was performed to understand the distributions of numerical and categorical variables. Missing values were identified and handled by removing the corresponding rows. Categorical variables were encoded using label encoding and one-hot encoding techniques.

Modelling approaches:

Three different machine learning models were implemented and evaluated: Decision Tree Classifier, XGBoost Classifier, and Support Vector Machine (SVM).

1. Decision Tree Classifier

A decision tree classifier was trained on the encoded dataset, achieving the following performance metrics:

- Accuracy: 0.817
- Precision: 0.724
- Recall: 0.875
- F1-score: 0.792

The confusion matrix, precision-recall curve, and ROC curve were plotted for further analysis. The decision tree model performed reasonably well, with a good balance of precision and recall.

```
python
# Decision Tree Classifier code snippet
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

The above is the code snippet for Decision Tree Classifier.

2. XGBoost Classifier

An XGBoost classifier was trained on the encoded dataset, achieving the following performance metrics:

- Accuracy: 0.833

- Precision: 0.792
- Recall: 0.792
- F1-score: 0.792

The ROC curve was plotted to visualize the model's performance. The XGBoost model showed improved accuracy compared to the decision tree model.

```
python
# XGBoost Classifier code snippet
model = XGBClassifier()
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

3. Support Vector Machine (SVM)

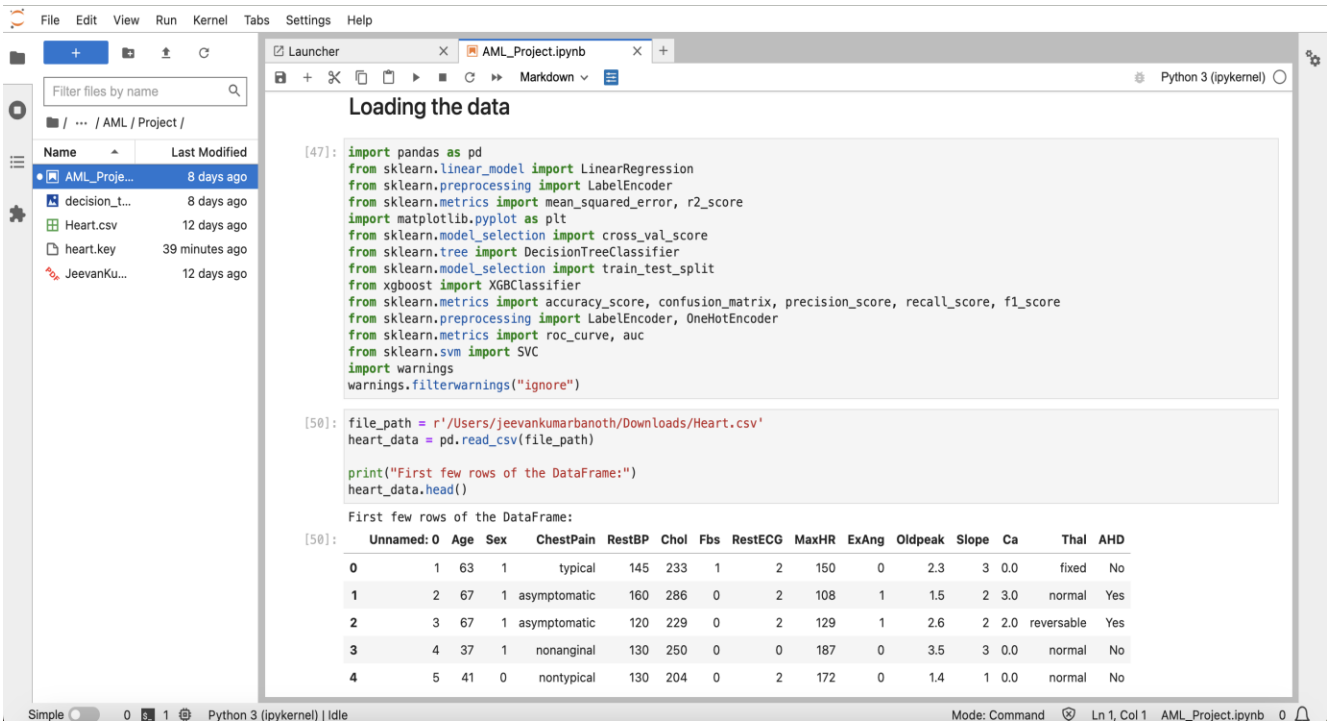
A linear SVM model was trained on the encoded dataset, achieving the following performance metrics:

- Accuracy: 0.850
- Precision: 0.826
- Recall: 0.792
- F1-score: 0.809

The ROC curve was plotted to visualize the model's performance. The SVM model emerged as the best performer, with the highest accuracy and a good balance of precision and recall

```
python
# SVM code snippet
model = SVC(kernel='linear', C=1.0)
model.fit(X_train, y_train)
y_pred = model.predict(X_test)
```

Loading the dataset:



Imports and Setup:

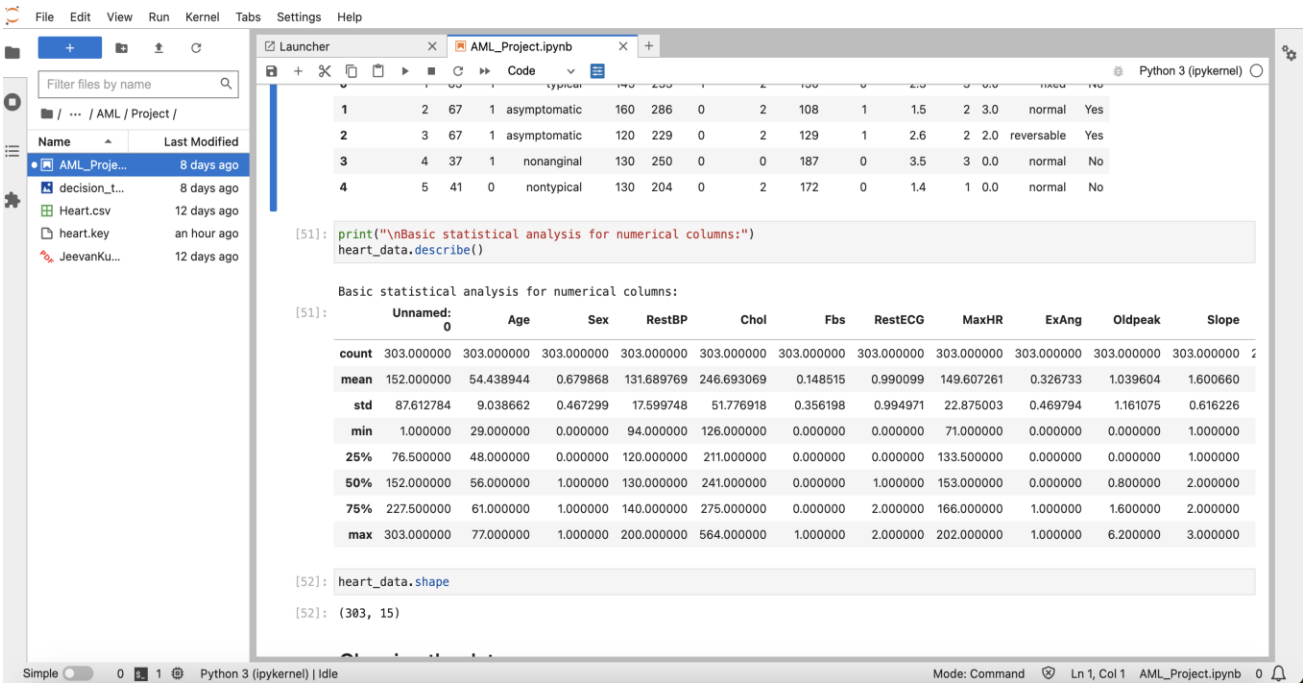
Importing libraries such as pandas, sklearn, and matplotlib for manipulating data, modeling machine learning, and visualization purposes.
We suppress warnings to maintain a clean output. Data Loading: We load the Heart disease dataset from a CSV file into a Pandas DataFrame.

Initial Data Display:

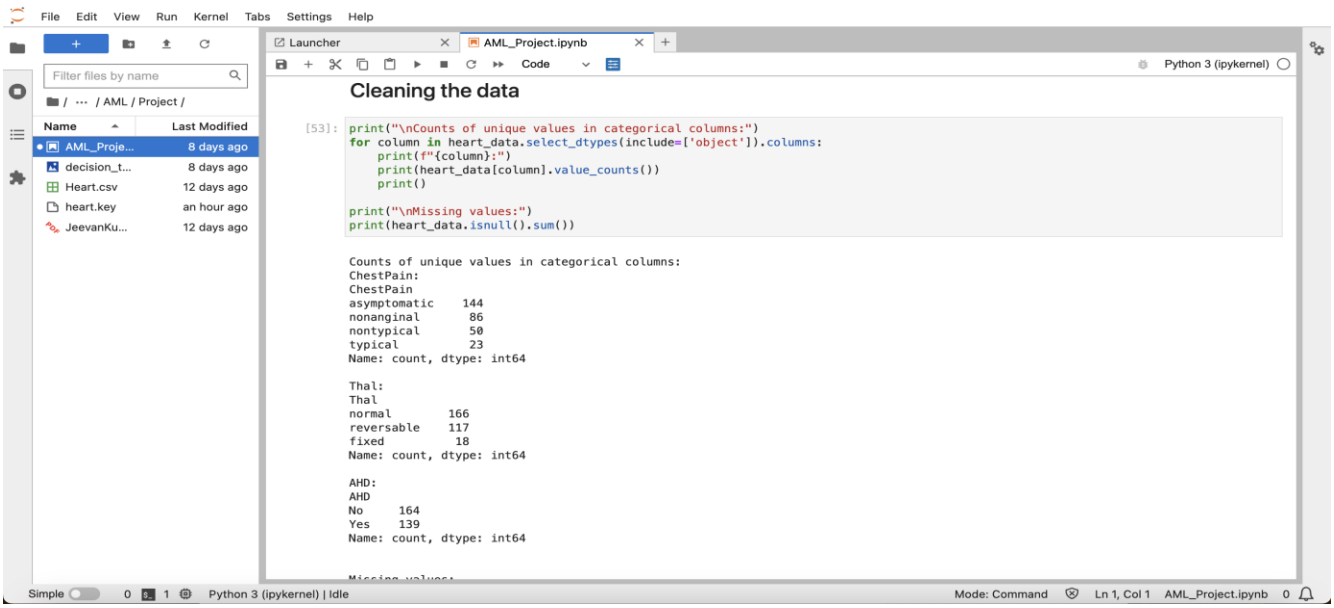
The first few rows of the DataFrame are displayed to provide an overview of the dataset, highlighting features like age, sex, chest pain type, blood pressure, cholesterol levels, etc.

Primary Functions of Libraries Used: - pandas for data manipulation. - sklearn for data preprocessing (label encoding, one-hot encoding), data splitting, and building machine learning models like (Linear Regression, Decision Tree, XGBClassifier, and SVM), and evaluating model performance (using metrics such as mean squared error, accuracy, precision, recall, F1 score, ROC curve).

The code snippet provided indicates the use of Matplotlib for plotting, but it is not actually implemented. XGBoost is utilized for more advanced ensemble machine learning techniques, which offer greater stability compared to simple models. The implied preparation for the machine learning model involves preprocessing the data (including handling categorical variables), dividing the data into training and testing sets, training multiple machine learning models for predicting heart disease (binary classification), and assessing their effectiveness.



As you can see here, we are printing the basic statistical analysis for numerical columns like count, std, mean, min, max etc., for keywords in dataset like age, sex, RestBP, Chol, Fbs, RestECG etc., as shown above.



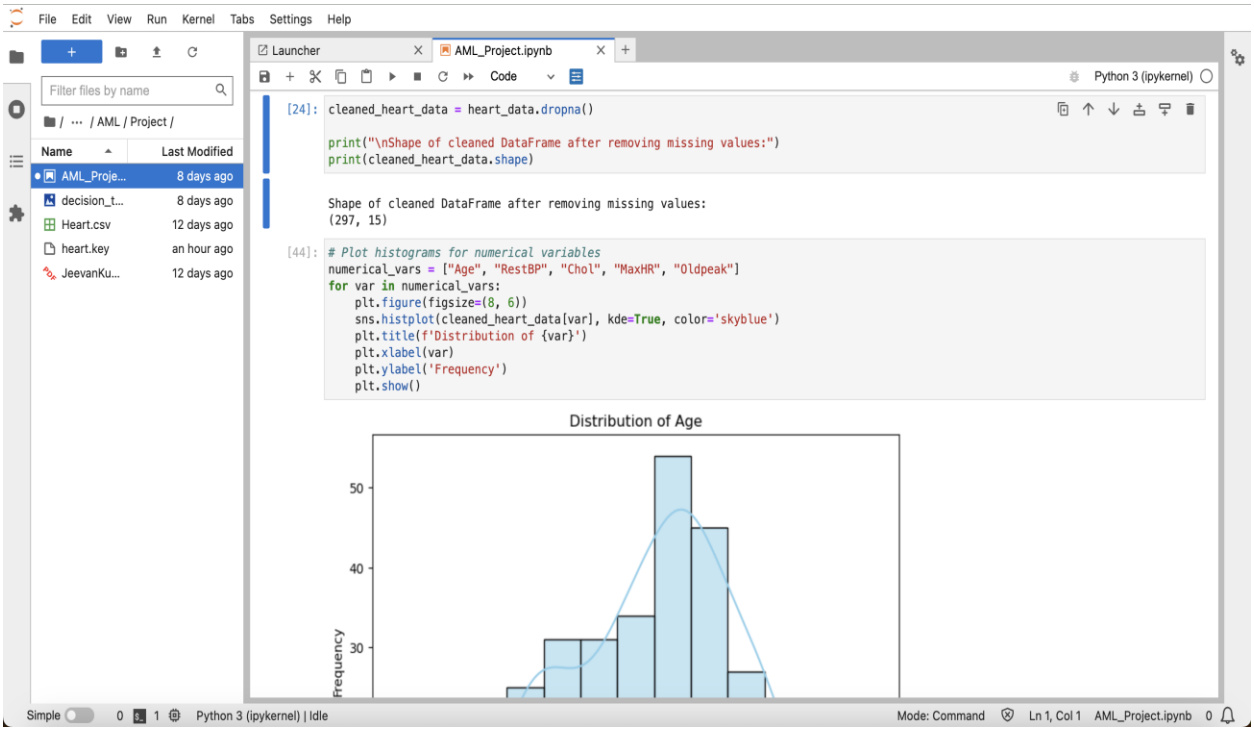
Cleaning the data:

Analysis of Categorical Columns: This code goes through columns that are of type 'object' (usually string or categorical) in the dataset. For every categorical column, it shows the number of unique values.

Specifically, it focuses on 'ChestPain', 'Thal', and 'AHD' columns to present the frequency of each category.

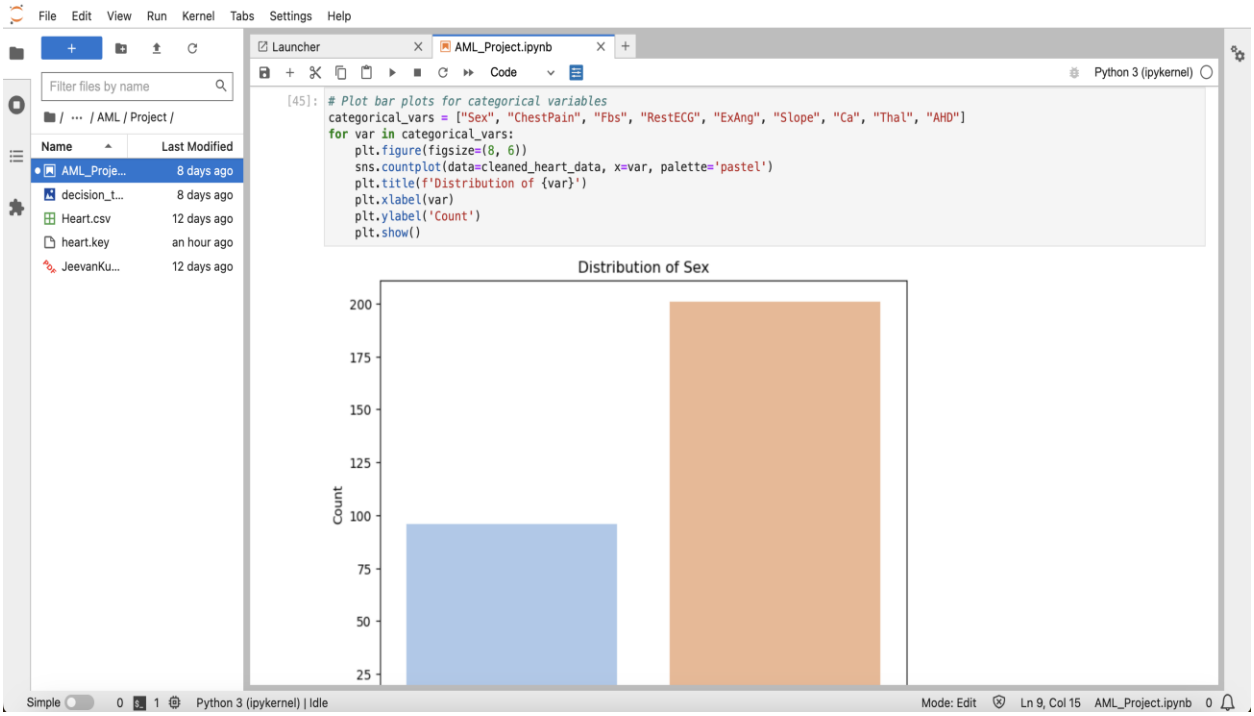
Analysis of Missing Values: The code looks for any missing values in all columns of the dataset. It displays the count of missing values for each column, noting that the 'Ca' column has 4 missing values and the 'Thal' column has 2 missing values.

This snippet is helpful for gaining insights into the distribution of categorical variables and assessing the level of data cleanliness (in terms of missing values) in the heart disease dataset.



When dealing with missing values in the heart disease dataset, the code efficiently removes rows with any missing data using the dropna() method. After this process, the dataset now contains 297 entries and 15 features, indicating that the removal of rows with missing data was successful.

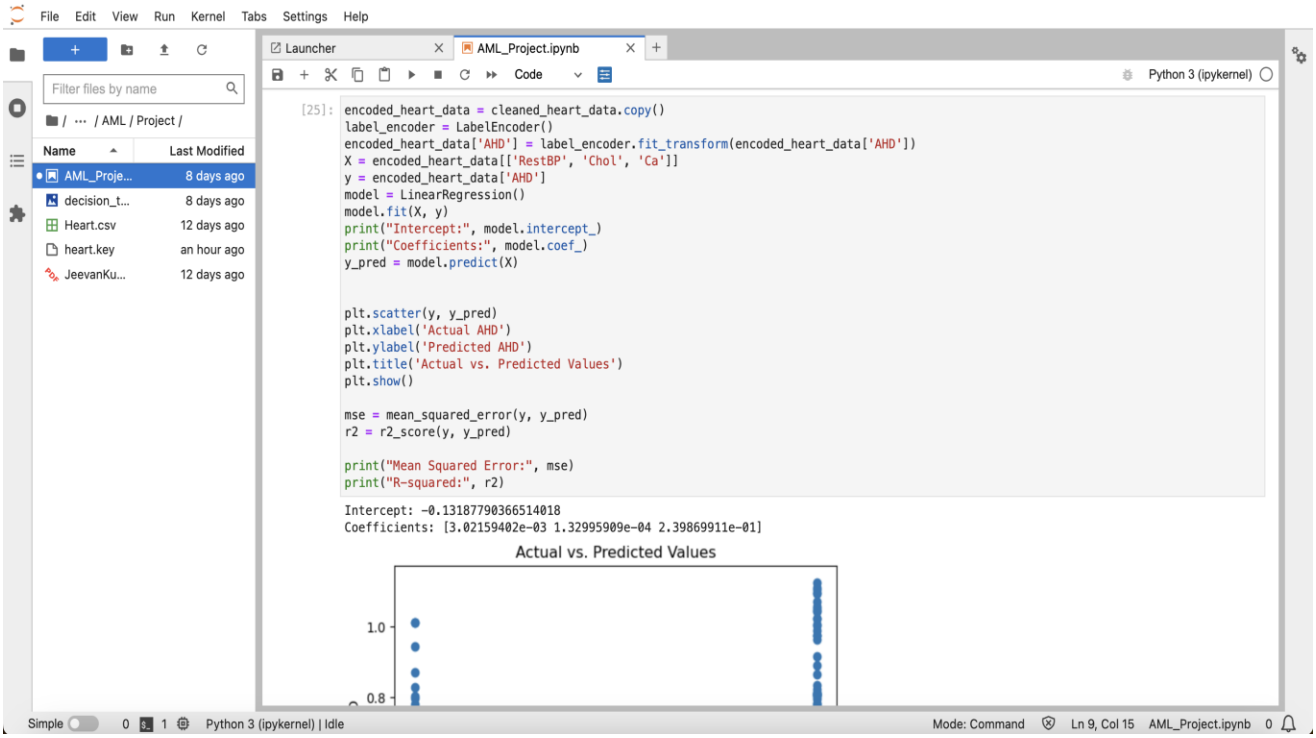
In terms of visualization, the code focuses on numerical variables such as Age, RestBP, Chol, MaxHR, and Oldpeak. It creates histograms for each of these variables using seaborn's histplot, with kernel density estimates (KDE) overlaid to provide a smoother representation of the distribution. Each histogram is carefully labeled and titled for clarity, and adjustments are made to the plot sizes to enhance readability.



Identifying Categorical Variables: This section includes the categorical variables such as Sex, ChestPain, Fbs, RestECG, ExAng, Slope, Ca, Thal, and AHD for visualization purposes.

Visualization Process:

For each variable mentioned above, a bar plot is created using seaborn's countplot function. The data used for each plot is sourced from the cleaned_heart_data DataFrame. The plots are designed with a 'pastel' color palette for a visually appealing display. Each plot has dimensions of 8x6 inches and features a title indicating the variable distribution, along with x and y-axis labels Count and the variable name respectively. Displaying Plots: Each plot is shown sequentially as the loop progresses through the categorical variables.



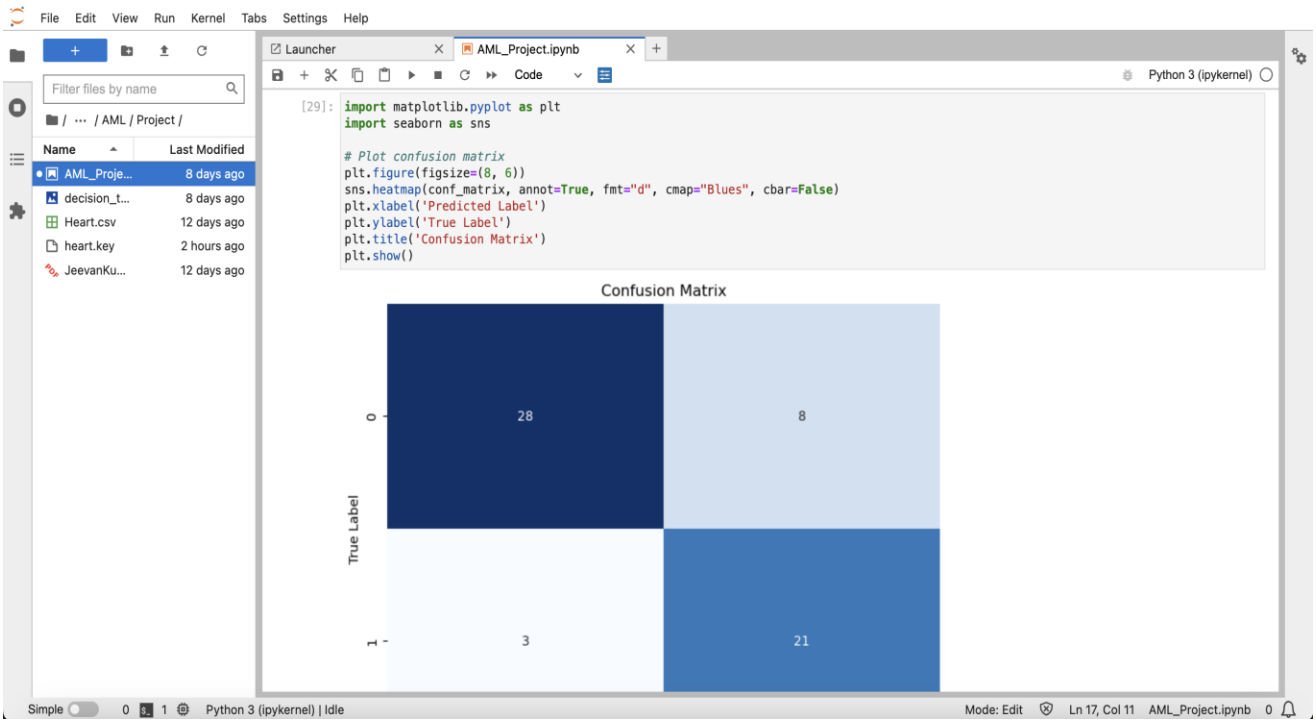
Data Preparation:

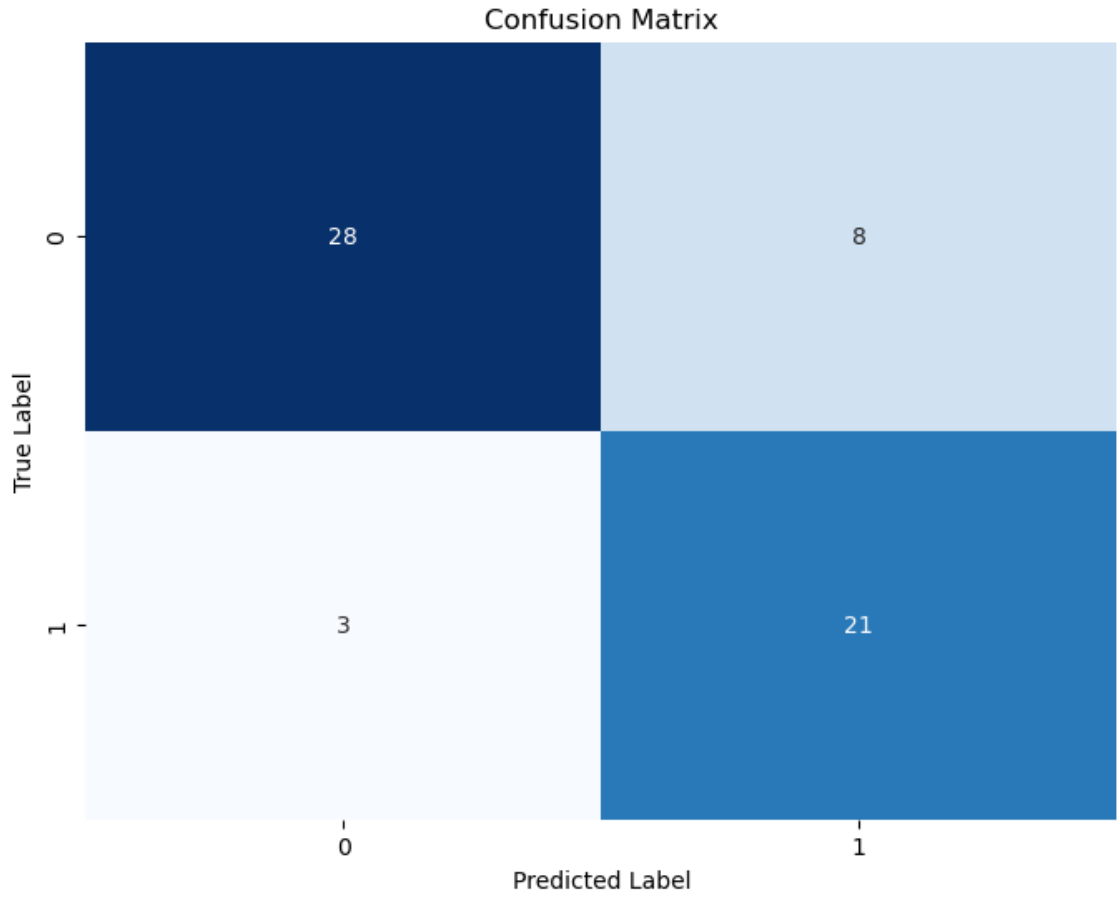
The first step involves duplicating the clean dataset and then utilizing a label encoder to convert the AHD column into a numeric format. This process involves converting categorical labels ('Yes', 'No') to numeric labels (1, 0).

Model Training:

The predictors (X) are selected as features RestBP, Chol, and Ca, while the response variable (y) is set as AHD. A linear regression model is then created and trained using these selected variables. The intercept and coefficients of the model are then displayed, offering a deeper understanding of the relationship between each predictor and the response variable.

Model Prediction and Evaluation: Once the model is trained, it can make predictions on the training data (X). These predictions are then compared against the actual values in a scatter plot, providing a visual representation of the accuracy of the model's predictions.


File Edit View Run Kernel Tabs Settings Help
Launcher
AML_Project.ipynb
Code
Python 3 (ipykernel)
Filter files by name
/ ··· / AML / Project /
Name
Last Modified
AML_Proje... 8 days ago
decision_t... 8 days ago
Heart.csv 12 days ago
heart.key 2 hours ago
JeevanKu... 12 days ago
[29]:
import matplotlib.pyplot as plt
import seaborn as sns
# Plot confusion matrix
plt.figure(figsize=(8, 6))
sns.heatmap(conf_matrix, annot=True, fmt="d", cmap="Blues", cbar=False)
plt.xlabel('Predicted Label')
plt.ylabel('True Label')
plt.title('Confusion Matrix')
plt.show()
Confusion Matrix
True Label
0
1
28
8
3
21
Simple
Python 3 (ipykernel) | Idle
Mode: Edit
Ln 17, Col 11
AML_Project.ipynb


Confusion Matrix
True Label
0
1
28
8
3
21
0
1
Predicted Label

Plot Setup:

The plot is initialized with a size of 8x6 inches. Creating Heatmap: The sns.heatmap function is utilized to generate the heatmap. The confusion matrix (conf_matrix) is presented with actual counts formatted as "d", and the color palette is set to "Blues". Numeric values within the heatmap squares are annotated.
The color bar (cbar) is disabled for a cleaner appearance. Adding Labels and Title: The x-axis is labeled as "Predicted Label", the y-axis as "True Label", and the plot is titled "Confusion Matrix". Showing the Plot: The plot is displayed using plt.show(). This visual representation aids in comprehending the performance of a classification model by illustrating the true positives, true negatives, and false positives counts.

XGBoost MODEL:

Output:

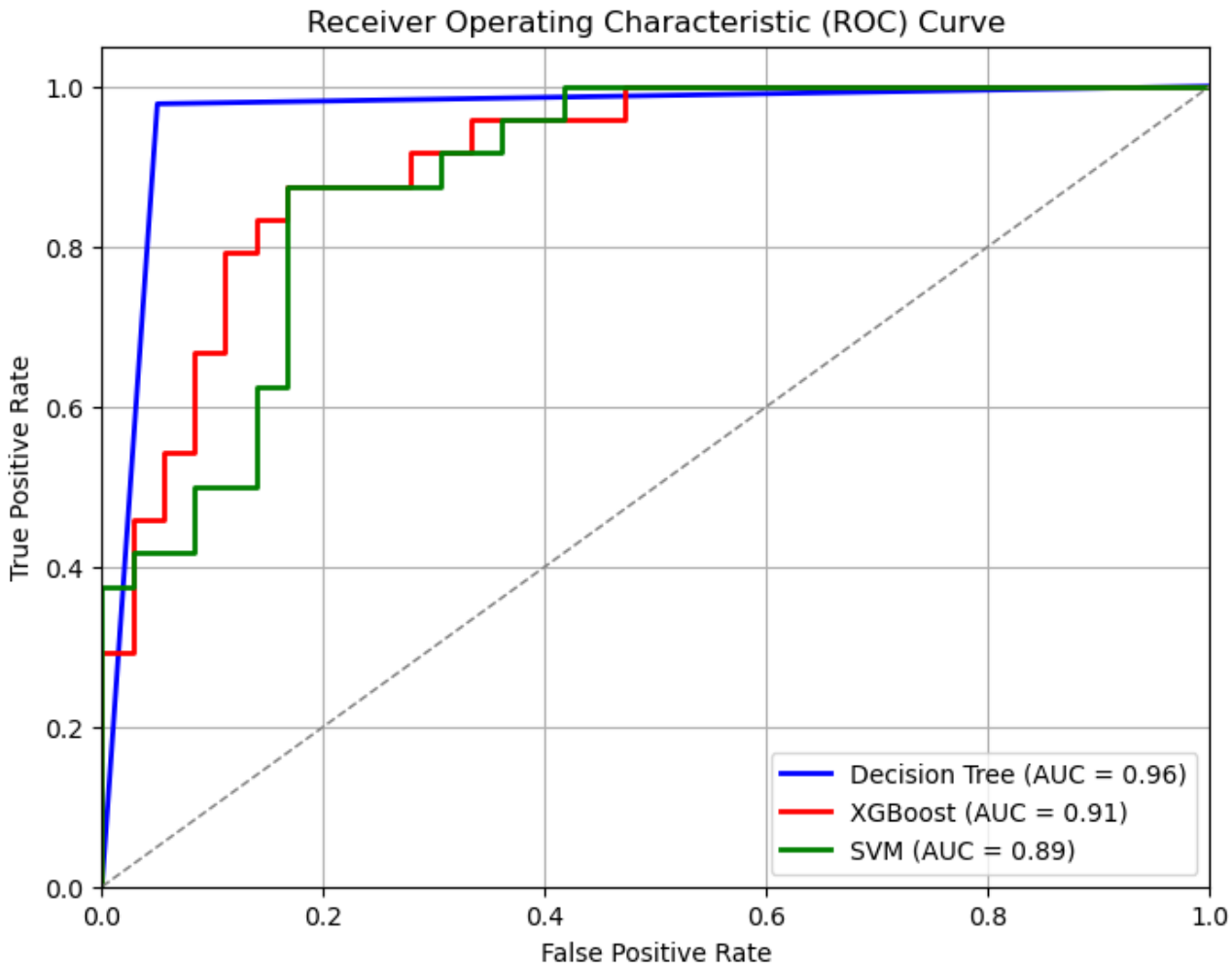
Accuracy: 0.8333333333333334
Confusion Matrix:
[[31 5]
[5 19]]
Precision: 0.7916666666666666
Recall: 0.7916666666666666
F1 Score: 0.7916666666666666

Support Vector Machines (SVM) MODEL:

Output:

Accuracy: 0.85
Confusion Matrix:
[[32 4]
[5 19]]
Precision: 0.8260869565217391
Recall: 0.7916666666666666
F1 Score: 0.8085106382978724

ROC Curve:



The following code snippet demonstrates the visualization of the Receiver Operating Characteristic (ROC) curves for three distinct classification models: Decision Tree, XGBoost, and SVM (Support Vector Machine).

ROC curves are visual representations that illustrate how well a classification model performs at different threshold levels. Additionally, the Area Under the Curve (AUC) is computed for each model, giving an overall indication of the model's ability to differentiate between classes.

This visualization aids in comparing the diagnostic capabilities of the Decision Tree, XGBoost, and SVM models based on their respective AUC values, offering insights into which model may excel in the dataset.

COMPARISON OF MODELS:

Output View

	Model	Precision	Recall	F1 Score	Accuracy
0	Decision Tree	0.724138	0.875000	0.792453	0.816667
1	XGBoost	0.791667	0.791667	0.791667	0.833333
2	SVM	0.826087	0.791667	0.808511	0.850000

The performance metrics of three different machine learning models are presented in the table below. Each model, namely Decision Tree, XGBoost, and SVM (Support Vector Machine), is evaluated based on precision, recall, F1 score, and accuracy: -

Decision Tree: Achieves a precision of 0.724, a recall of 0.875, an F1 score of 0.792, and an accuracy of 0.817.

XGBoost: Displays a precision of 0.792, a recall of 0.792, an F1 score of 0.792, and an accuracy of 0.833.

SVM: Exhibits the highest precision of 0.826, a recall of 0.792, an F1 score of 0.809, and the highest accuracy among the three models at 0.850.

Conclusion:

In this study, we utilized different machine learning models to predict heart disease using medical data. The SVM model stood out as the top performer with an accuracy of 0.850 and balanced precision and recall. However, there is room for improvement through additional tuning and feature engineering.

The decision tree and XGBoost models also performed well and merit further investigation. It is crucial to emphasize that these models should complement medical expertise, rather than replacing it entirely. They can aid healthcare professionals in decision-making and identifying heart disease risk factors.

References:

Dataset - https://data.world/uci/heart-disease/activity

XGBoost - https://machinelearningmastery.com/develop-first-xgboost-model-python-scikit-learn/