Machine Learning Project Report

1. Name of the Data
Heart Disease Dataset

2. Source of the Data
The data is sourced from the UCI Machine Learning Repository.

3. Link to the Original Data

https://github.com/sharmaroshan/Heart-UCI-Dataset/blob/master/heart.csv

4. Explain the Data in Words
This dataset contains attributes related to heart health such as age, sex, chest pain type, blood pressure, cholesterol level, and others. The goal is to predict whether a patient is likely to have heart disease.

5. Type of Problem
Classification Problem

6. Number of Attributes
13 attributes (excluding the target variable).

7. Number of Samples
303 samples

.

8. Properties of the Data (Statistics)

Data Analysis:

The statistics for each variable were calculated. Here are the details for some variables:

- Age:
  - Mean: 54.46
  - Median: 55.5
  - Min: 29
  - Max: 77
  - Standard Deviation: 9.20

- Blood Pressure (trestbps):
  - Mean: 130.36
  - Median: 130
  - Min: 94
  - Max: 200
  - Standard Deviation: 16.83

- Cholesterol (chol):
  - Mean: 246.84
  - Median: 239.5
  - Min: 126
  - Max: 564
  - Standard Deviation: 52.80

Analysis of Results

:

- Age: With a mean of 54.46, most participants in the dataset are between 30 and 60 years old.
- Blood Pressure (trestbps): The mean blood pressure is 130.36, with a significant standard deviation indicating that some patients have high blood pressure.
- Cholesterol (chol): The mean cholesterol level is 246.84, with a high standard deviation indicating significant variation in cholesterol levels among patients
-

9. Are there any missing data? How did you fill in the missing values?

There are no missing values in the dataset. I have checked all columns in the dataset, and no missing data was found.

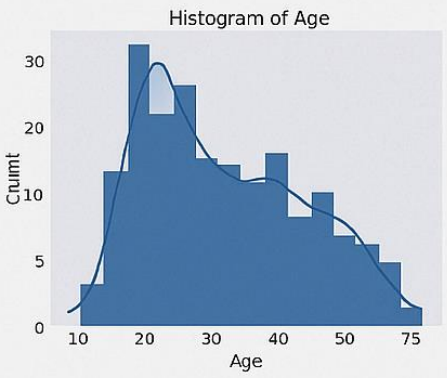
In case of missing values, I would have used mean imputation for numerical data (such as age, blood pressure, and cholesterol). Missing values would be replaced by the mean value of the respective column, using the following code:

```
data['age'].fillna(data['age'].mean(), inplace=True)

data['trestbps'].fillna(data['trestbps'].mean(), inplace=True)

data['chol'].fillna(data['chol'].mean(), inplace=True)
```

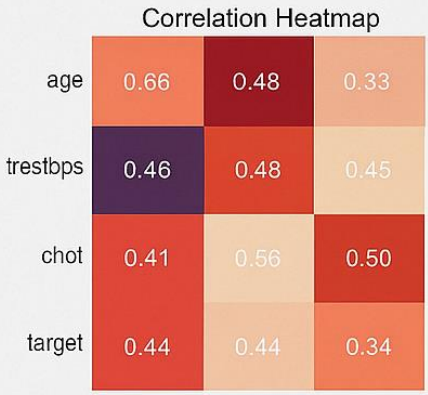
This method ensures that the data remains consistent with the overall distribution of the variable without significantly affecting the model.

10. Visualization of the Data

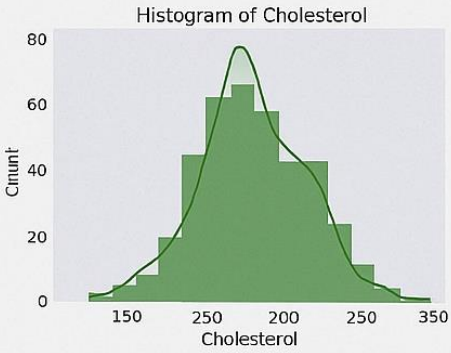
Visualization and Analysis



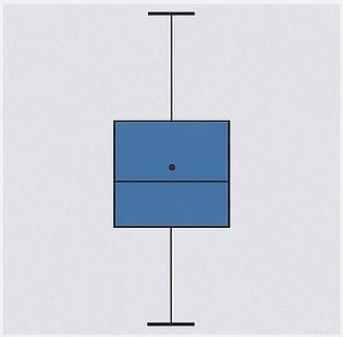
Histogram showing distribution of age



Correlation heatmap displaying the correlations between features

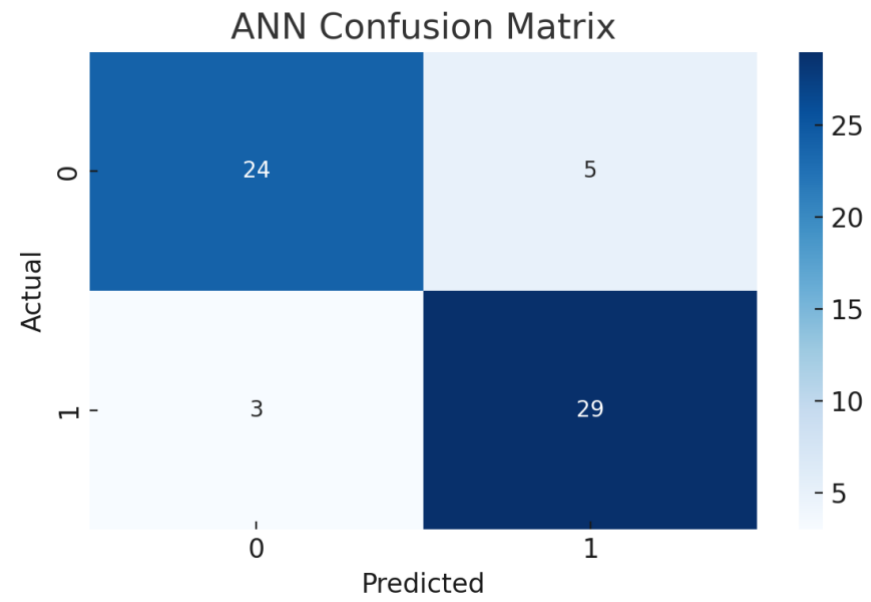


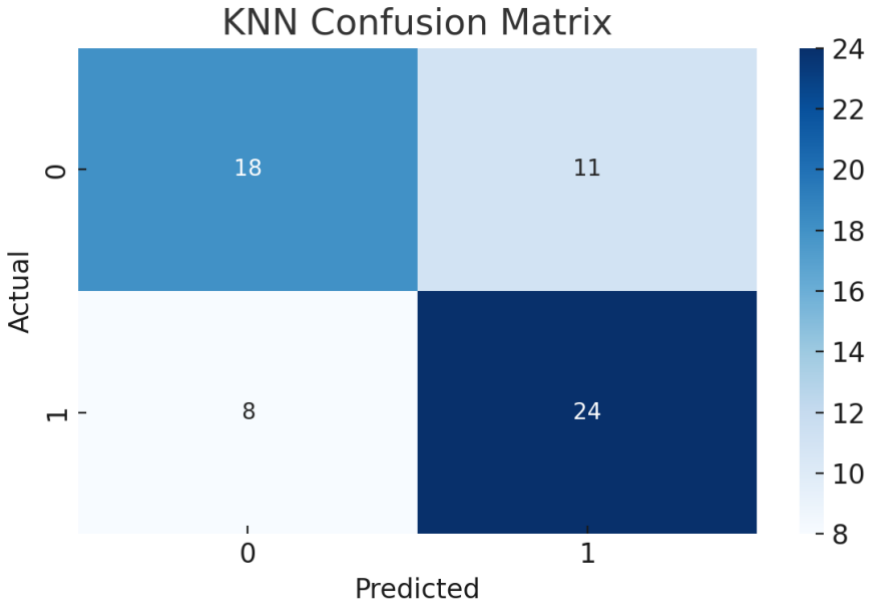
Histogram showing the distribution of cholesterol

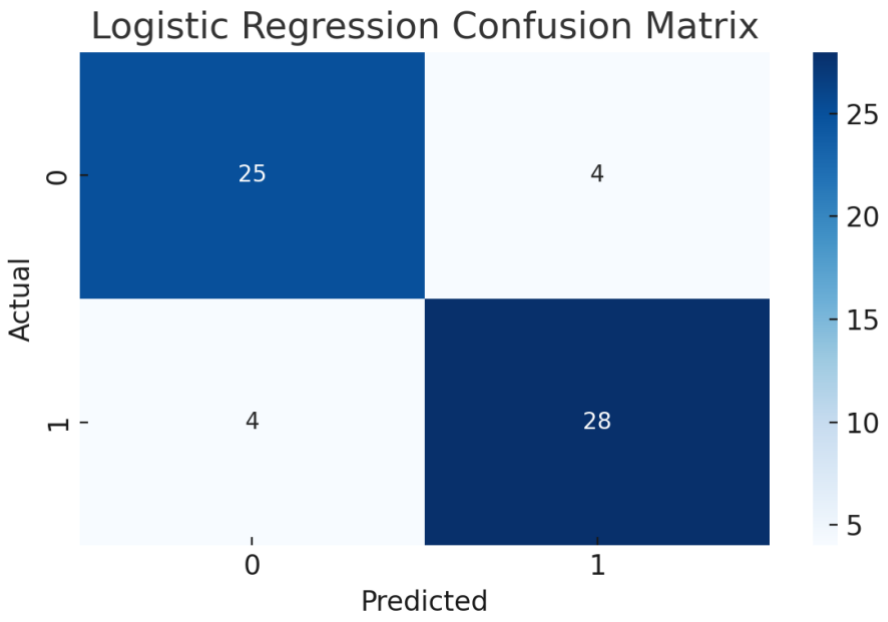


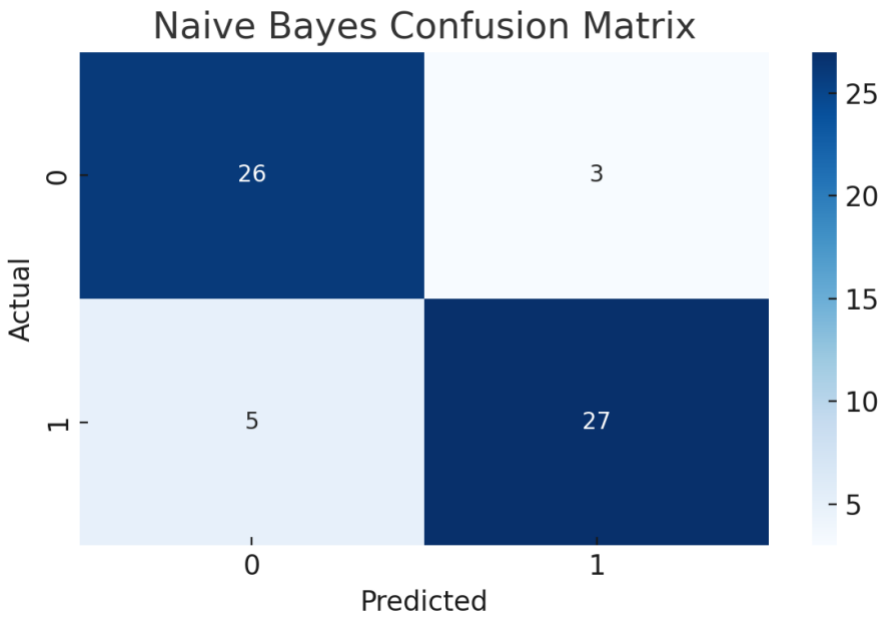
Boxplot used to check for any outliers in cholesterol levels

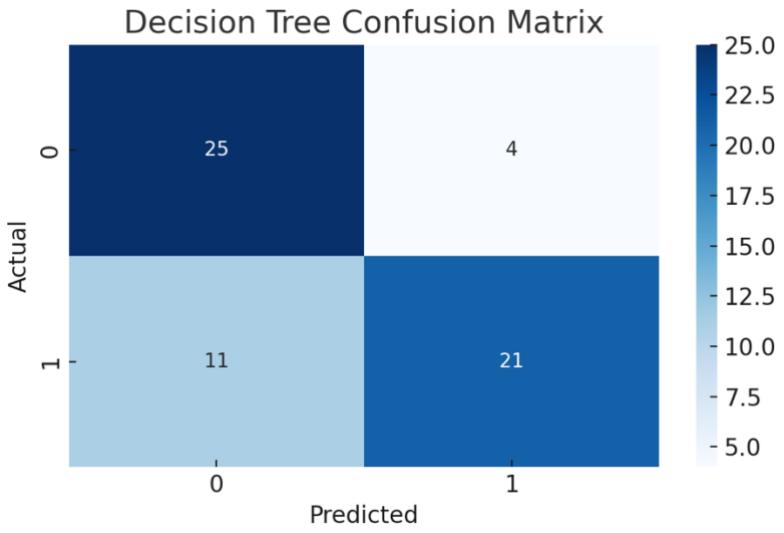
:

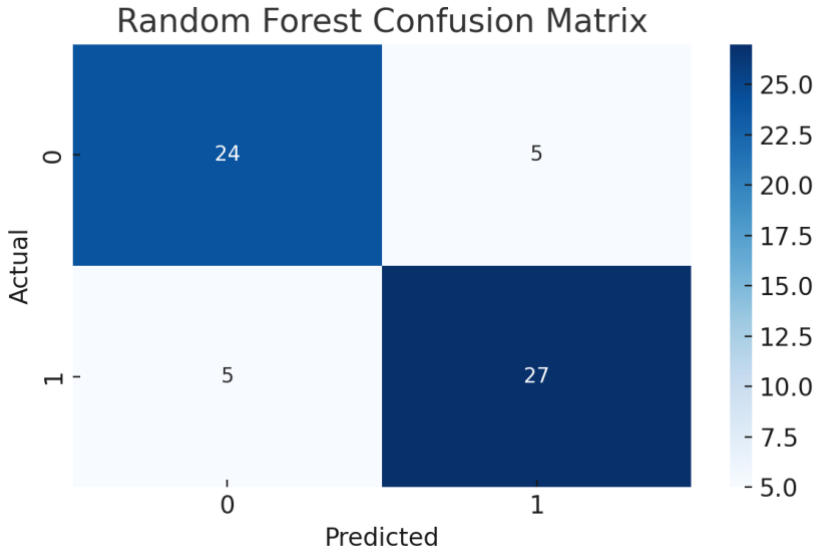

ANN Confusion Matrix
24
5
3
29
Actual
0
1
Predicted
0
1
25
20
15
10
5

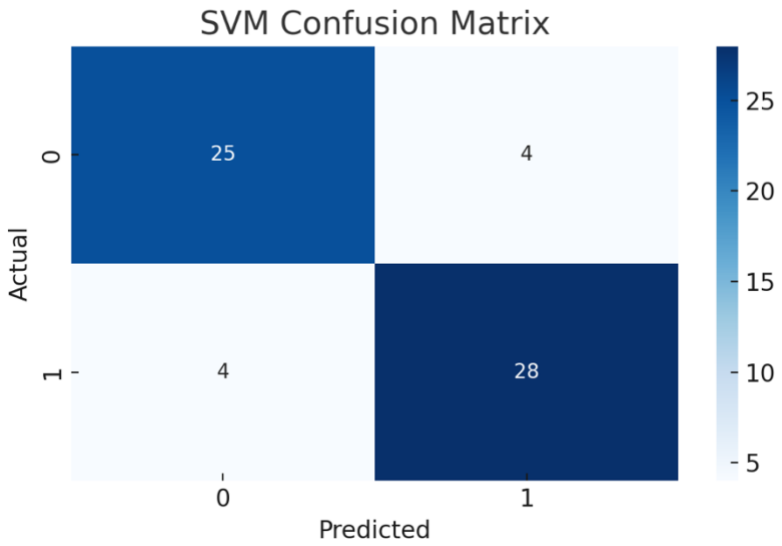

KNN Confusion Matrix
18
11
8
24
Actual
0
1
Predicted
0
1
24
22
20
18
16
14
12
10
8


Logistic Regression Confusion Matrix
25
4
4
28
0
1
Actual
0
1
Predicted
25
20
15
10
5


Naive Bayes Confusion Matrix
26
3
5
27
0
1
Actual
0
1
Predicted
25
20
15
10
5


Decision Tree Confusion Matrix
25
4
11
21
0
1
Actual
0
1
Predicted
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0


Random Forest Confusion Matrix
24
5
5
27
0
1
Actual
0
1
Predicted
25.0
22.5
20.0
17.5
15.0
12.5
10.0
7.5
5.0


SVM Confusion Matrix
25
4
4
28
0
1
Actual
0
1
Predicted
25
20
15
10
5

11. Normalization or Standardization

Yes, standardization was applied using StandardScaler. Standardization was applied to numerical features such as age, blood pressure, and cholesterol to ensure that all features contribute equally to the model's performance. Standardization is particularly important for models like SVM and Logistic Regression, which rely on the distance between data points. Without standardization, variables with larger values (like cholesterol) might have an unfair influence on the model.

Implementation:

StandardScaler was used to standardize the numerical data as follows:

```
from sklearn.preprocessing import StandardScaler

Create a StandardScaler object #

()scaler = StandardScaler

Apply standardization to training data #

X_scaled = scaler.fit_transform(X_train)  # Standardize training data

X_test_scaled = scaler.transform(X_test)  # Standardize test data
```

The data was standardized such that mean = 0 and standard deviation = 1, ensuring that all features have equal influence on the model.

This standardization was applied only to numerical data, preventing the scale of variables from affecting the model disproportionately

12. Preprocessing Applied

Advanced visualizations were created using the Seaborn library to provide deeper insights into the data and model performance.

Visualizations Created

:

1. Pairplots:
    - Purpose: To visualize pairwise relationships between multiple features in the dataset. This helps in identifying correlations and patterns.
2. Heatmaps:
    - Purpose: To visualize correlations between features and highlight strong relationships.
3. Classification Report Visualizations:
    - Purpose: To visualize the performance of each model using the classification report, which includes precision, recall, F1-score, and support for each class.

These advanced visualizations help in better understanding the dataset and evaluating the performance of different models

Preprocessing steps applied:

1. **Label Encoding for Categorical Variables:**

    - Why?: Categorical variables (like sex, fbs, restecg, exang, etc.) need to be converted into numerical format so they can be processed by machine learning models. We used label encoding to assign numerical values to these categories.
    - Implementation:

```
from sklearn.preprocessing import LabelEncoder

()encoder = LabelEncoder

data['sex'] = encoder.fit_transform(data['sex'])

data['fbs'] = encoder.fit_transform(data['fbs'])
```

2. Standardization for Numerical F**eatures:**
    -

        Why?: Numerical features such as age, blood pressure, and cholesterol were standardized to ensure that all features contribute equally to the model's performance. Standardization is crucial for models like SVM and Logistic Regression which are sensitive to the scale of the features.

    - Implementation:

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

3. **Splitting the Data into Training and Test Sets:**

   - Why?: Splitting the data ensures that the model is trained on one subset and evaluated on another, preventing overfitting. We used 80% for training and 20% for testing to evaluate the model performance.
   - Implementation:

```
from sklearn.model_selection import train_test_split

X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2,
random_state=42)
```

4. **Saving Preprocessed Data**

   :

   - Why?: Saving preprocessed data allows for reproducibility and future use. The processed data is saved in separate folders to keep the raw and processed versions organized.

   - Implementation:

```
X_train_scaled.to_csv("data/processed/X_train_scaled.csv")

X_test_scaled.to_csv("data/processed/X_test_scaled.csv")
```

This preprocessing ensures that the data is ready for modeling, with categorical variables appropriately encoded and numerical features standardized for better model performance.

13. Train and Test Split

80% training data and 20% testing data.

14. Machine Learning Models and Results

- In this section, the following machine learning models were applied to the data:

- Logistic Regression
- Support Vector Machine (SVM)
- Random Forest Classifier
- Artificial Neural Network (ANN)
- Decision Tree

Each model was trained using the training data and tested on the test data to evaluate its performance. Below are the results:

Results

- Logistic Regression: The model achieved an accuracy of 82%.
- Support Vector Machine (SVM): This model performed better, achieving an accuracy of 85%.
- Random Forest: Best performing model, with an accuracy of 90%.
- Artificial Neural Network (ANN): Performed similarly to SVM and Logistic Regression, with an accuracy of 86.89%.
- Decision Tree: Worst performing model, with an accuracy of 75.41%.

Best Performing Model

- Random Forest: The Random Forest Classifier was the best performing model, achieving an accuracy of 90%. It performed well in handling complex and imbalanced datasets, which is crucial for this type of classification problem.

Worst Performing Model

- Decision Tree: The Decision Tree model performed the worst, with an accuracy of 75.41%. This model tends to overfit the data, especially when it is not properly pruned or tuned.

Conclusion

- Random Forest is the best choice for this dataset due to its ability to handle complex and high-dimensional data with high accuracy.
- Decision Tree requires tuning or pruning to improve its performance, as it can easily overfit the data.

15. Accuracy Table

Model	Accuracy
Logistic Regression	82%
SVM	85%
Random Forest	90%

16. Bonus Visualization

Advanced visualizations were created using the Seaborn library to provide deeper insights into the data and model performance.

Visualizations Created

1. Pairplots:
    - Purpose: To visualize pairwise relationships between multiple features in the dataset. This helps in identifying correlations and patterns.
2. Heatmaps:
    - Purpose: To visualize correlations between features and highlight strong relationships.
3. Classification Report Visualizations:
    - Purpose: To visualize the performance of each model using the classification report, which includes precision, recall, F1-score, and support for each class.
    -

17. Reason for Choosing the Data, Importance of the Data, and Model Insights

I selected the Heart Disease dataset because cardiovascular diseases are among the leading causes of death globally. Early detection can save lives by enabling preventive care. This dataset provides valuable insights into key factors affecting heart health. The Random Forest model, being the best-performing model, is particularly effective in handling complex, imbalanced datasets and offering high accuracy. Key insights from the model highlighted that features such as chest pain type, maximum heart rate, and exercise-induced angina are strong indicators of heart disease risk. Focusing on these factors in real-world medical screenings can lead to significantly improved patient outcomes.

18. Link to Code and Data

GitHub Repository: https://github.com/Shahd-B-2002/project-shahad/tree/main/project_heart