Predicting Heart Disease in Patients Using Machine Learning

First Large Project
Course: SAT5114
Proff: Dr. Weihua Zhou
Submitted By: Usama Ayub
Group Members: Usama Ayub & Navya Sadineni

<u>Introduction:</u>

The objective of this project is to evaluate different machine learning models to predict the presence of heart disease in patients based on various features. This report outlines the data analysis, preprocessing steps, model selection, hyperparameter tuning, and performance evaluation.

1. Data Exploration and Visualization:

The dataset contains 13 predicting features such as age, gender, cholesterol levels, resting blood pressure, and more. Target variable is Binary series with '0' and '1' entries as negative and positive disease labels. Visualized data distribution and target variable distribution.

```
df = pd.read_csv('heart.csv')
df.head()
```

✓ 0.0s

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	0
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	0
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	0
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	0
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	0

```
df.info()
```

✓ 0.0s

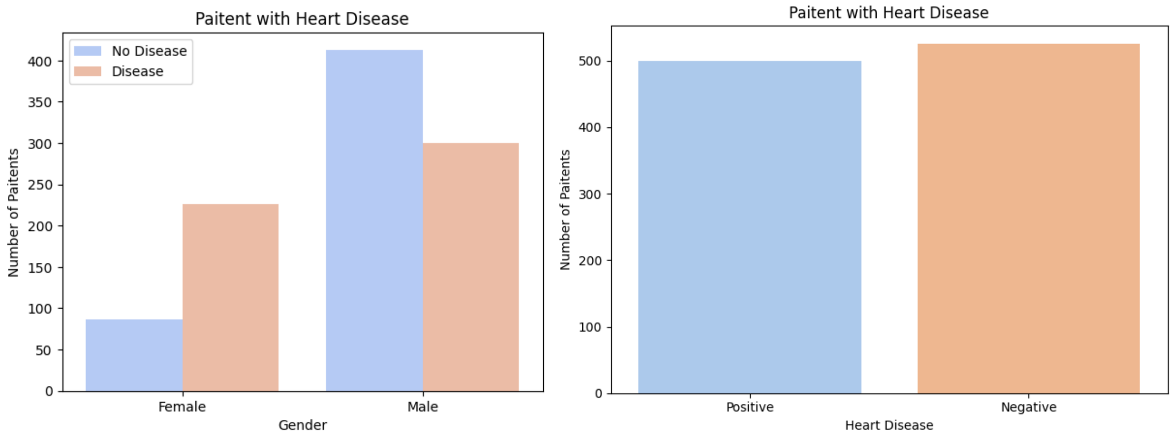
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1025 entries, 0 to 1024
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   age       1025 non-null   int64
 1   sex       1025 non-null   int64
 2   cp        1025 non-null   int64
 3   trestbps  1025 non-null   int64
 4   chol      1025 non-null   int64
 5   fbs       1025 non-null   int64
 6   restecg   1025 non-null   int64
 7   thalach   1025 non-null   int64
 8   exang     1025 non-null   int64
 9   oldpeak   1025 non-null   float64
 10  slope     1025 non-null   int64
 11  ca        1025 non-null   int64
 12  thal      1025 non-null   int64
 13  target    1025 non-null   int64
dtypes: float64(1), int64(13)
memory usage: 112.2 KB
```

```
df.describe()
```

✓ 0.0s

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak
count	1025.000000	1025.000000	1025.000000	1025.000000	1025.00000	1025.000000	1025.000000	1025.000000	1025.000000	1025.000000
mean	54.434146	0.695610	0.942439	131.611707	246.00000	0.149268	0.529756	149.114146	0.336585	1.071512
std	9.072290	0.460373	1.029641	17.516718	51.59251	0.356527	0.527878	23.005724	0.472772	1.175053
min	29.000000	0.000000	0.000000	94.000000	126.00000	0.000000	0.000000	71.000000	0.000000	0.000000
25%	48.000000	0.000000	0.000000	120.000000	211.00000	0.000000	0.000000	132.000000	0.000000	0.000000
50%	56.000000	1.000000	1.000000	130.000000	240.00000	0.000000	1.000000	152.000000	0.000000	0.800000
75%	61.000000	1.000000	2.000000	140.000000	275.00000	0.000000	1.000000	166.000000	1.000000	1.800000
max	77.000000	1.000000	3.000000	200.000000	564.00000	1.000000	2.000000	202.000000	1.000000	6.200000

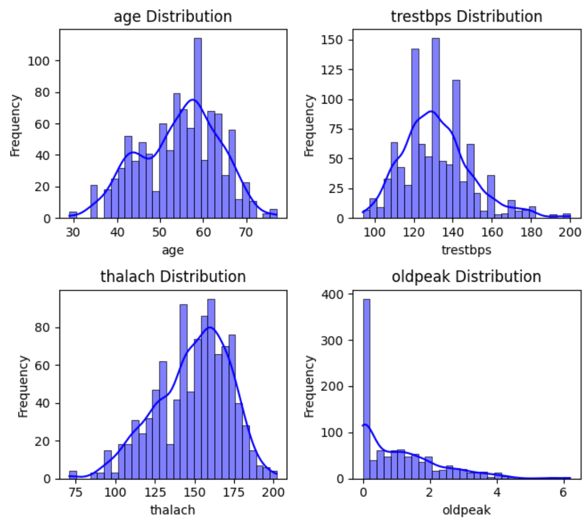
Distribution of Patients with respect to genders and disease



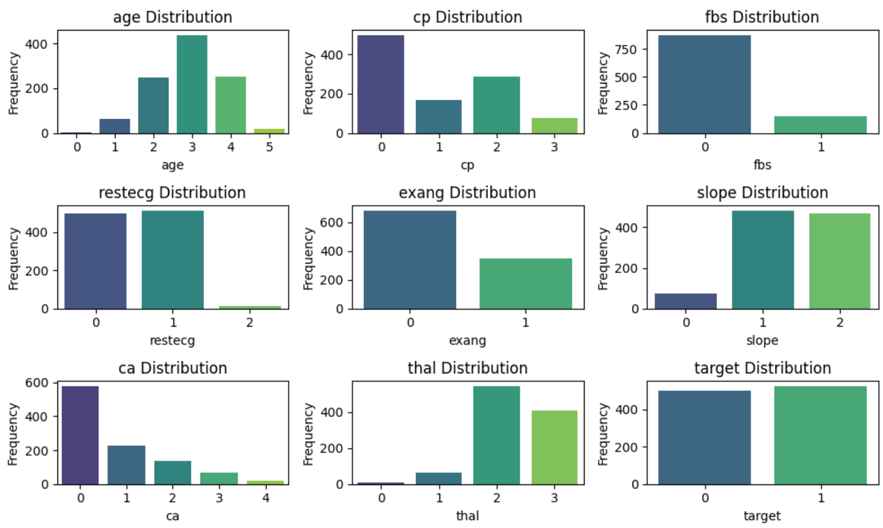
EDA applied on Dataset which includes identification of null values, outliers, discretization, and the distribution of data.

Data visualization techniques, including histograms, boxplots, and count plots, were used to understand the distribution and relationships between different variables.

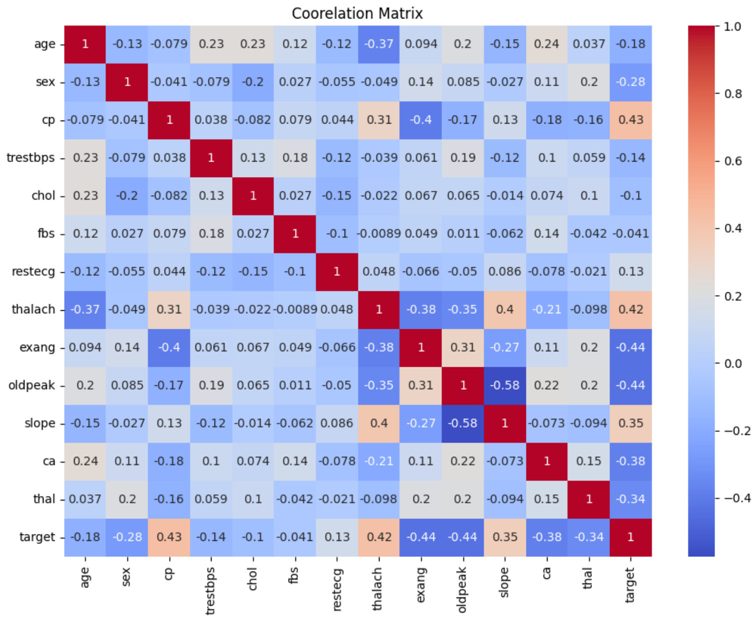
Histograms for Numeric Variables. All features are almost uniformly distributed and there is not much of skewness.



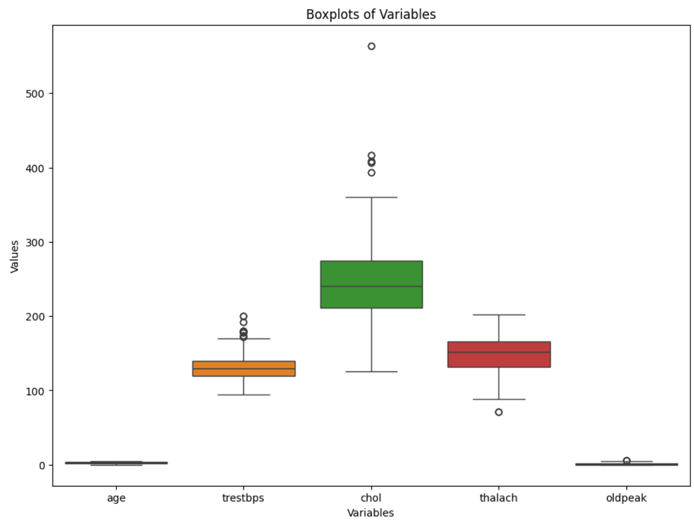
Distribution of Categorical features.



Heatmap plot is used to check the Correlation among features.
Key insights:

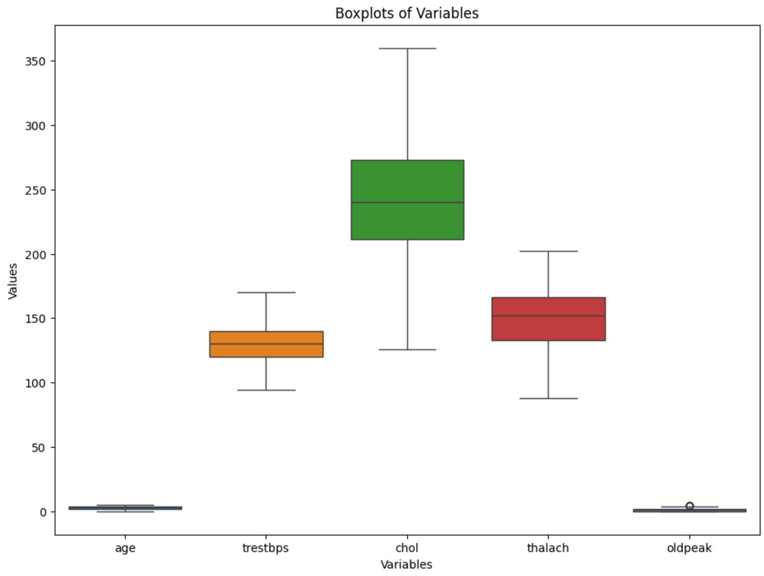


The dataset comprises both numerical and binary features.
Some features exhibit outliers, shown in the plot below.



Outliers are removed using statistical techniques and alimented data points those lie outside the min and maximum values.

After removing outliers.



2. Data Preprocessing:

Standardization and label encoding applied to prepare the data for model training.

```
# Applying the standard scaler to the dataset
from sklearn.preprocessing import StandardScaler
new_df = df.copy()
scaler = StandardScaler()
new_df[nominal_features] = scaler.fit_transform(new_df[nominal_features])
```
✓ 0.2s

```
# Applying the label encoder to the binary features
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
new_df[binom] = new_df[binom].apply(lambda col: label_encoder.fit_transform(col))
```
✓ 0.0s

The dataset splited into training and testing with stratified random sampling and sets to evaluate model performance.

3. Model Selection and Training:

Three machine learning models were chosen: Logistic Regression, Support Vector Machine (SVM), and Random Forest Classifier.

Each model was trained on the training dataset, and its performance was evaluated using accuracy metrics.

Trainning and Testing of Models on Trainning Dataset

```
# creating a function to fit the models for Training Dataset
# Predicting on the training dataset and calculating the accuracy score

def fit_and_score_train(models, X_train, X_test, y_train, y_test):
    np.random.seed(42)
    model_scores = {}
    for name, model in models:
        model.fit(X_train, y_train)
        model_scores[name] = model.score(X_train, y_train)
    return (model_scores)

model_scores = fit_and_score_train(models, X_train, X_test, y_train, y_test)
model_scores
```
21] ✓ 0.5s

```
{'Logistic Regression': 0.8780487804878049,
 'SVM': 0.901219512195122,
 'Random Forest': 1.0}
```

Trainning and Testing of Models and Predicting on Test Dataset

```python
# creating a function to fit the models for Test Dataset

def fit_and_score_test(models, X_train, X_test, y_train, y_test):
    np.random.seed(42)
    model_scores = {}
    for name, model in models:
        model.fit(X_train, y_train)
        model_scores[name] = model.score(X_test, y_test)
    return (model_scores)

model_scores = fit_and_score_test(models, X_train, X_test, y_train, y_test)
model_scores
```

✓ 0.4s

```
{'Logistic Regression': 0.8926829268292683,
 'SVM': 0.8780487804878049,
 'Random Forest': 1.0}
```

K-Fold CV is used to select the best model, which in our case is Random Forest Classifier based on the mean accuracy score. RFC achieved 99.4% accuracy.

Applying K-Fold Cross Validation

```python
# Selecting best model using Cross validation by iterating through the models

for name, model in models:
    cv_scores = cross_val_score(model, X, y, cv=5, scoring='accuracy')
    print(f'{name} has a cross-validated score of {(cv_scores)}')
    print(f'{name} has a cross-validated mean score of {np.mean(cv_scores)}\n')
```

✓ 1.7s

```
Logistic Regression has a cross-validated score of [0.92195122 0.86829268 0.88780488 0.83414634 0.84390244]
Logistic Regression has a cross-validated mean score of 0.8712195121951221

SVM has a cross-validated score of [0.89756098 0.87804878 0.89268293 0.84390244 0.86341463]
SVM has a cross-validated mean score of 0.8751219512195123

Random Forest has a cross-validated score of [1.         1.         0.98536585 1.         0.98536585]
Random Forest has a cross-validated mean score of 0.9941463414634146
```

4. Feature Selection and Hyperparameter Tuning:

Selected 50% features for final model.

Selecting Random Forest as a Best Model

Feature Selection and Testing

```python
from sklearn.feature_selection import SelectFromModel

clf = RandomForestClassifier(random_state=42)

selector = SelectFromModel(estimator = clf)
selector.fit(X_train, y_train)

# Get the selected features
selected_features = X_train.columns[selector.get_support()]

# Transform the training and test sets with selected features
X_train_selected = selector.transform(X_train)
X_test_selected = selector.transform(X_test)
```

Random search technique is used to identify the best hyperparameters for the Random Forest Classifier. The selected hyperparameters are used to train the final model.

```
param_grid = {
    'n_estimators': [50, 100, 150],
    'max_depth': [5, 10, 15, None],
    'min_samples_split': [2, 5, 10],
    'min_samples_leaf': [1, 2, 4],
    'max_features': ['sqrt', 'log2', None]
}
```

Performing Random Search for Hyperparameter Tunning

```
from sklearn.model_selection import RandomizedSearchCV

grid_search = RandomizedSearchCV(estimator = RandomForestClassifier(), param_distributions = param_grid, n_iter=10, cv=3,
                                 random_state=42, scoring='accuracy')
grid_search.fit(X_train_selected, y_train)
best_param = grid_search.best_params_
```

✓ 5.2s

5. Model Evaluation:

The final Random Forest Classifier achieved an accuracy of 100% on train dataset and an accuracy of 100% on the test dataset.
Confusion matrix, classification report, sensitivity, and specificity metrics were used to assess model performance. See figure below.

Model Performance

Accuracy with Best-RFC on Train dataset: 1.0

Accuracy with Best-RFC on Test dataest: 1.0

Confusion Matrix

[[100 0]

[3 105]]

Classification Report

```
              precision    recall  f1-score   support

           0       1.00      1.00      1.00       100
           1       1.00      1.00      1.00       105

    accuracy                           1.00       205
```

macro avg 1.00 1.00 1.00 205 weighted avg 1.00 1.00 1.00 205

Sensitivity : 1.0

Specificity : 1.0

6. Conclusion:

The Random Forest Classifier demonstrated outstanding performance in predicting heart disease in patients.s

7. Future Work:

Investigate additional machine learning algorithms, ensemble methods and deep learning algorithms to further enhance predictive performance.

Explore the use of advanced techniques such as feature engineering and dimensionality reduction using PCA and t-SNE to improve model efficiency.