HEART DISEASE PREDICTION

Computer Science and Engineering Department

Thapar Institute of Engineering and Technology

(Deemed to be University), Patiala – 147004

Machine Learning Project

PROJECT GITHUB LINK-
https://github.com/tarunbhatti7/Heart_Disease_Prediction_ML_Project.git

Submitted By:

Tarun Bhatti

102216105

Preetinder Singh Kundi

102216125

Submitted To:

Ms. Kudratdeep Aulakh

Index

1. Introduction

1.1 Name of the dataset : Heart Failure Prediction Dataset

Dataset Link:
https://www.kaggle.com/datasets/fedesoriano/heart-failure-prediction

About Dataset:

The dataset used, heart.csv, is sourced from publicly available health data and includes clinical and demographic features known to be associated with heart disease. Each record in the dataset represents an individual's health metrics along with a target variable indicating the presence (1) or absence (0) of heart disease.

1.2 Description

- **Objective:** Predict heart disease likelihood using machine learning.
- **Interface:** A user-friendly **Stream-lit app** for data input and prediction.
- **Models Used:** Machine learning algorithms such as Logistic Regression, KNN Classifier , Decision Tree Classifier , XG Boost (tree ensembles),Random Forest ,and (SVM).
- **Features:** Inputs include age, sex, chest pain, blood pressure, cholesterol, heart rate, and other key health metrics.
- **Data Processing:** Inputs are standardized using Standard Scaler for consistency with training data.
- **Output:** A binary result indicating if heart disease is likely.
- **Deployment:** Accessible via a web-based platform for quick and easy predictions.

2. Libraries Used

1. **NumPy:** A powerful library for numerical computations and handling multi-dimensional arrays and matrices.
2. **Matplotlib.pyplot:** A plotting library used for creating visualizations such as charts and graphs to analyze data.
3. **Pandas:** A data manipulation and analysis library providing data structures like DataFrames for handling structured data.
4. **Scikit-learn (Sklearn):** A comprehensive library for machine learning and data preprocessing.
    1. **preprocessing (LabelEncoder, StandardScaler):** Tools for encoding categorical data and standardizing numerical features.
    2. **model_selection (train_test_split):** Splits datasets into training and testing subsets.
    3. **linear_model (LogisticRegression):** Implements logistic regression for classification tasks.
    4. **metrics (accuracy_score):** Provides methods to evaluate model performance.
    5. **svm (SVC):** Supports vector machines for classification tasks.
    6. **neighbors (KNeighborsClassifier):** Implements K-Nearest Neighbors for classification.
    7. **tree (DecisionTreeClassifier):** Builds decision trees for classification tasks.
    8. **ensemble (RandomForestClassifier):** Combines multiple decision trees for better predictions.
    9. **xgboost (XGBClassifier):** An efficient, scalable implementation of gradient boosting for classification.
5. **Pickle:** A Python module to serialize and save machine learning models or objects for reuse.

Algorithm(s) Used

- **Logistic Regression:** A statistical method for binary classificationthat models the probability of class membership using a sigmoid function

- **Support Vector Machine (SVM):** A supervised learning algorithm that finds the optimal hyperplane to separate classes by maximizing the margin between data points of different categories.

- **K-Nearest Neighbors (KNN):** A simple, non-parametric algorithm that classifies data points based on the majority class of their nearest neighbors.

- **Random Forest:** An ensemble learning method that constructs multiple decision trees and combines their outputs for more accurate and robust predictions.

- **Decision Tree Classifier:** A tree-based model that splits data into subsets based on feature values, making sequential decisions to classify data points.

- **XGBoost Classifier:** An efficient gradient boosting algorithm that optimizes decision trees iteratively to improve accuracy and handle large datasets effectively.

3. Code and Screenshots

1. LOGISTIC REGRESSION (accuracy score)

```
# testing accuracy of logistic regression in %
from sklearn.metrics import accuracy_score
                    # accuracy_score(original value , predicted values )

logistic_acc = accuracy_score(Y_test,y_prediction_logistic)*100
print(logistic_acc)
```

[14]

```
83.69565217391305
```

2. SUPPORT VECTOR MACHINE (SVM) (accuracy score)

```
# testing accuracy of svm model in %
svm_acc= accuracy_score(Y_test,y_pred_svm)*100
print(svm_acc)
```

[17] ✓ 0.0s

```
86.41304347826086
```

3. KNEIGHBORS CLASSIFICATION (accuracy score)

```
# testing accuracy of k neighbors in %
kneighbors_acc= accuracy_score(Y_test,y_pred_kneighbors)*100
print(kneighbors_acc)
```

[20] ✓ 0.0s

```
84.78260869565217
```

4. DECISION TREE (accuracy score)

```
# accuracy of testing data in %
decision_tree_acc = accuracy_score(Y_test,y_pred_decision_tree) *100
print(decision_tree_acc)
```

[23] ✓ 0.0s

```
77.71739130434783
```

5. RANDOM FOREST ACCURACY SCORE

```
# testing accuracy of random forest in %
random_forest_acc = accuracy_score(Y_test,y_pred_randomforest) *100
print(random_forest_acc)
```

[26] ✓ 0.0s

```
85.86956521739131
```

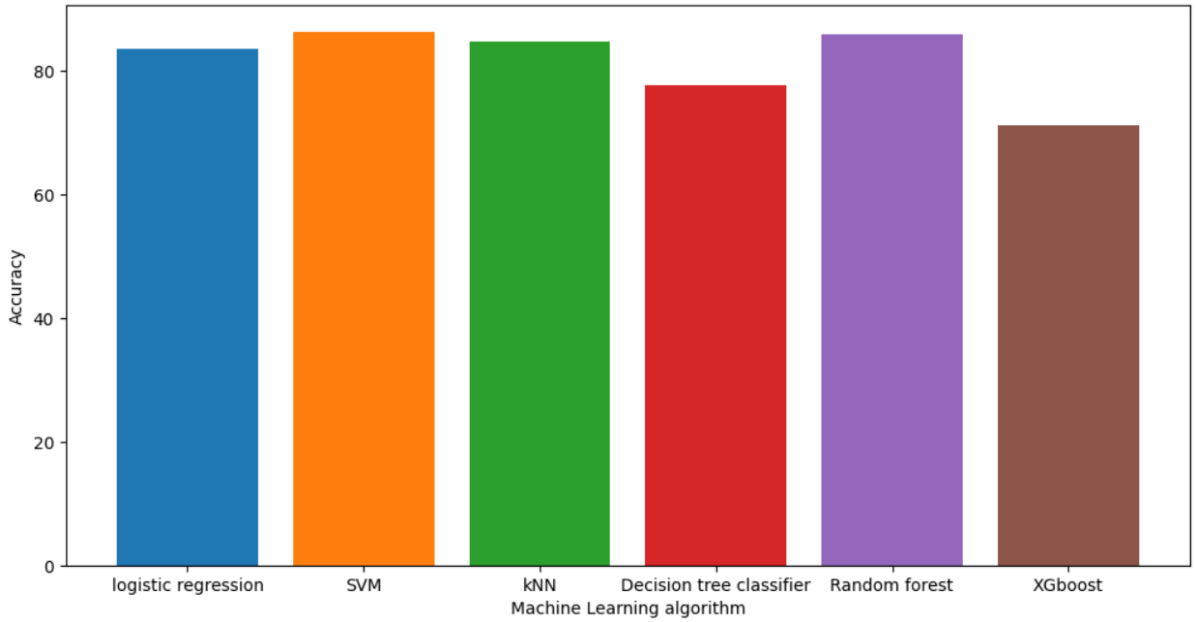
6. XGBOOST ACCURACY SCORE

```
# testing the accuracy in %
xgboost_acc = accuracy_score(Y_test,y_pred_xgboost)*100
print(xgboost_acc)
```

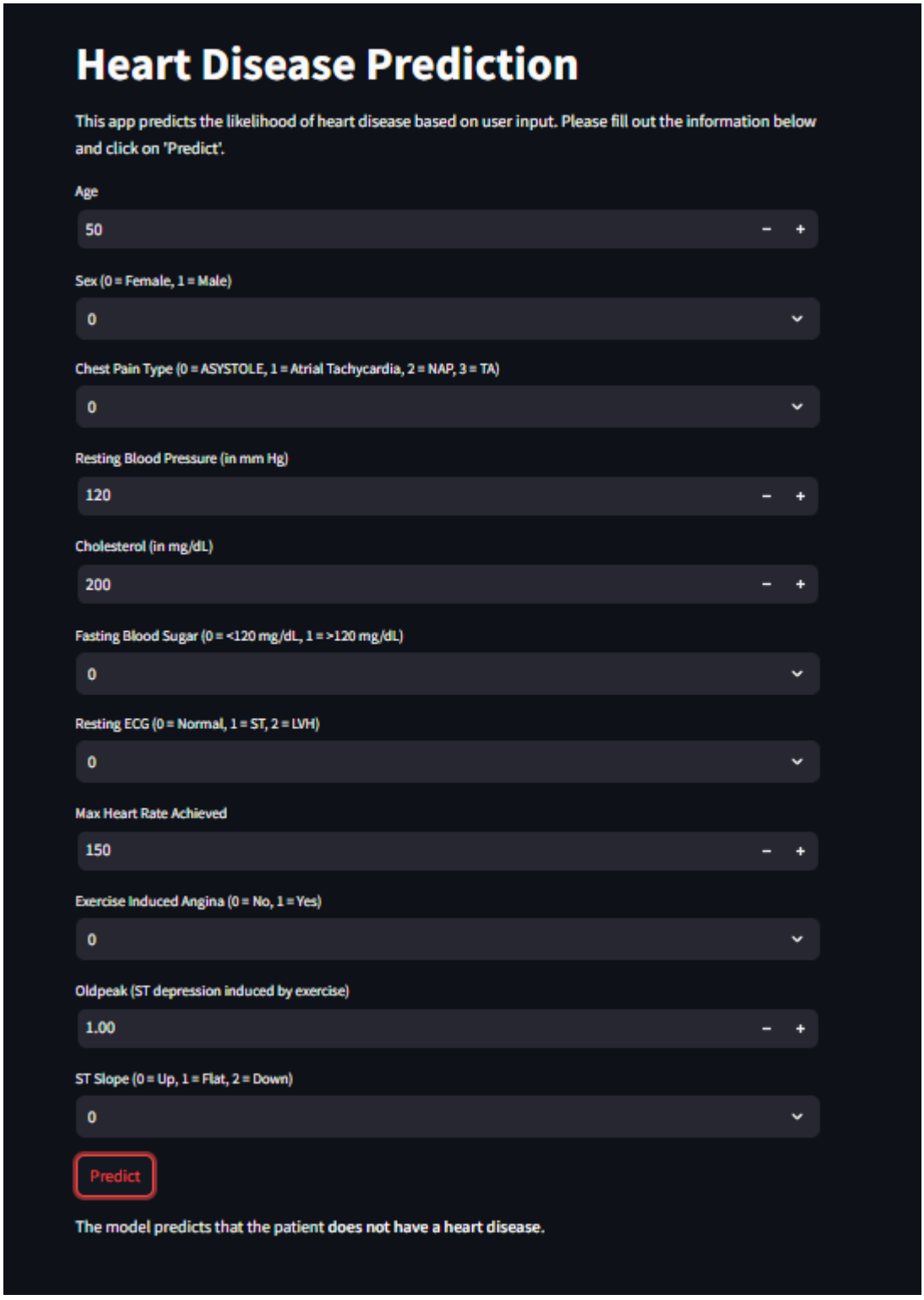
[29] ✓ 0.0s

```
71.19565217391305
```

BAR GRAPH BETWEEN ACCURACY SCORE AND MACHINE LEARNING ALGORITHMS



HEART DISEASE PREDICTION FINAL INTERFACE



OUTPUT- The model predicts that the patient does not have a heart disease .