NETAJI SUBHAS UNIVERSITY OF TECHNOLOGY

PROJECT REPORT
CSDS (2021-25)

Submitted to: Prof. M.P.S Bhatia

Submitted By: **Ishit Sardana (2021UCD2115)**
Nimish Gupta (2021UCD2140)
Nandani Kodan (2021UCD2144)

Project Report: Heart Disease Prediction Using Machine Learning

Introduction

Heart disease is a leading cause of mortality worldwide, making early diagnosis crucial for effective treatment and prevention. Machine learning has emerged as a valuable tool for predicting the likelihood of heart disease based on various patient attributes. In this project, we have developed a machine learning model to predict whether a person has heart disease or not using a dataset containing several patient features.

Data Description

The dataset used in this project contains the following features:
- ***id:*** Unique identifier for each patient.
- ***age:*** Age of the patient in years.
- ***origin:*** Place of study.
- ***sex:*** Gender of the patient (Male/Female).
- ***cp:*** Chest pain type (Typical angina, Atypical angina, Non-anginal, Asymptomatic).
- ***trestbps:*** Resting blood pressure (in mm Hg on admission to the hospital).
- ***chol:*** Serum cholesterol level in mg/dL.
- ***fbs:*** Fasting blood sugar > 120 mg/dL (True/False).
- ***restecg:*** Resting electrocardiographic results (Normal, ST-T Abnormality, LV Hypertrophy).
- ***thalach:*** Maximum heart rate achieved.
- ***exang:*** Exercise-induced angina (True/False).
- ***oldpeak:*** ST depression induced by exercise relative to rest.
- ***slope:*** The slope of the peak exercise ST segment.
- ***ca:*** Number of major vessels coloured by fluoroscopy (0-3).
- ***thal:*** Thalassemia type (Normal, Fixed Defect, Reversible Defect).
- ***num*:** The predicted attribute (0: No heart disease, 1: Heart disease).

Data Preprocessing

Data preprocessing is a crucial step in any machine learning project. In this project, we performed the following preprocessing steps:
- Handling missing values: We checked for missing values in the dataset and either imputed them or removed rows with missing values as appropriate.
- Encoding categorical variables: We one-hot encoded categorical variables like 'sex,' 'cp,' 'restecg,' 'thal,' and 'exang' to convert them into numerical form.
-- Splitting the data: We split the dataset into training and testing sets to evaluate the model's performance.

Model Selection

For this classification problem, we experimented with machine learning algorithms, including logistic regression.We evaluated the model's performance using various metrics, including accuracy, precision,
recall, F1-score, and ROC-AUC. The choice of evaluation metrics was based on the nature of the problem and the trade-offs between false positives and false negatives in heart disease prediction.

Results

The final model achieved the following results:
- Accuracy: 81.9%
- Precision: 84.375%

- Recall: 81.8%
- F1-score: 83%
- ROC-AUC: 89.097%

These results indicate that the model performs well in predicting heart disease based on the given features.

Conclusion

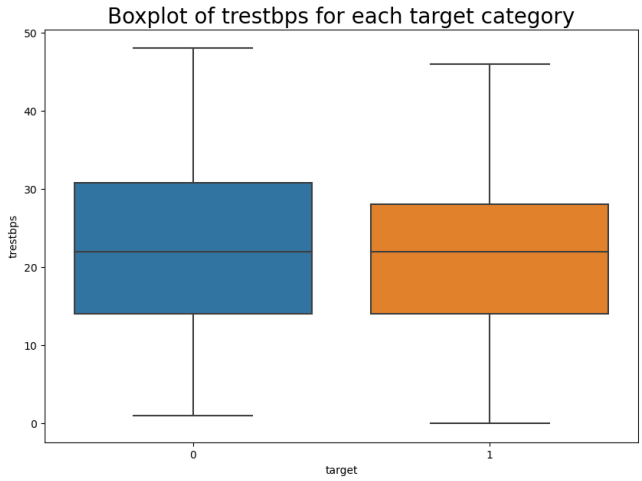
In this project, we developed a machine learning model to predict the likelihood of heart disease in individuals based on various patient attributes. The model demonstrates promising results and can be a valuable tool for early diagnosis and intervention. Further research and validation on larger and more diverse datasets can enhance the model's accuracy and robustness.

This project has the potential to significantly impact heart disease diagnosis and prevention, ultimately improving the quality of healthcare for individuals at risk.

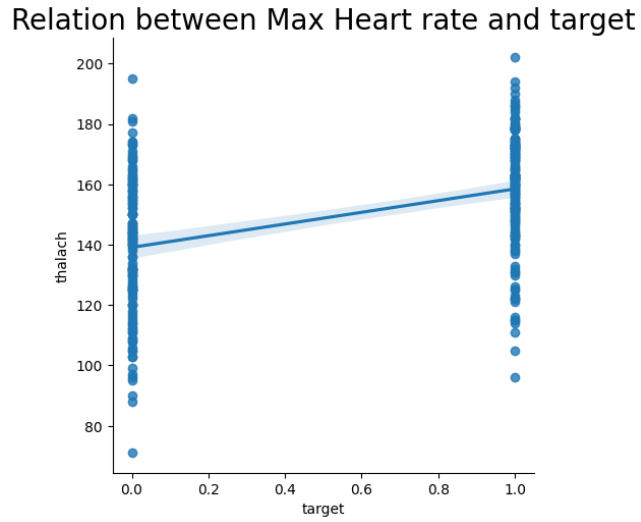
CODE :

```
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.metrics import precision_score, recall_score, f1_score , roc_auc_score,
roc_curve, auc
import matplotlib.pyplot as plt
import seaborn as sns

data = pd.read_csv(‘heart_disease_data.csv')
data.shape
data.info()
data.describe()
plt.rcParams['figure.figsize'] = (15, 15)
sns.pairplot(data)
sns.heatmap(data)
plt.rcParams['figure.figsize'] = (10, 7)
sns.displot(data['age'])
plt.title('Distribution of Age’)
print(data[‘target'].value_counts())
sns.countplot(x = data['target'] , palette = 'pastel')
plt.show()
from sklearn.preprocessing import LabelEncoder

# Assuming 'categorical_column' is the column you want to convert
label_encoder = LabelEncoder()
data['trestbps'] = label_encoder.fit_transform(data[‘trestbps'])


sns.boxplot(x='target', y='trestbps', data=data)
```

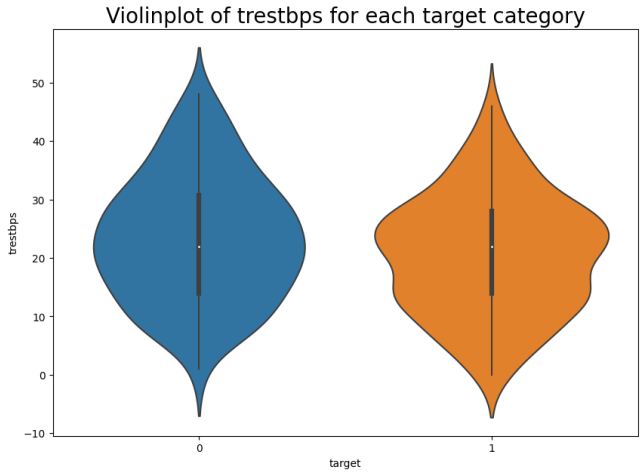
```
plt.title('Boxplot of trestbps for each target category', fontsize=20)
plt.show()
```



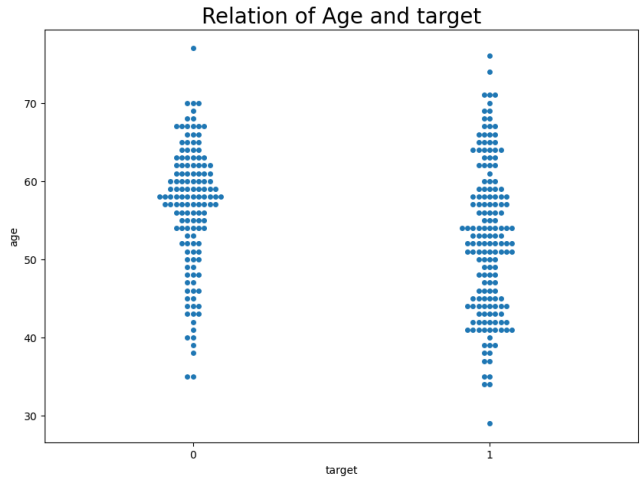

```
data['target'] = data['target'].cat.as_ordered()
plt.rcParams['figure.figsize'] = (10, 7)
sns.lmplot(x='target', y='thalach', data=data)
plt.title('Relation between Max Heart rate and target', fontsize = 20)
```



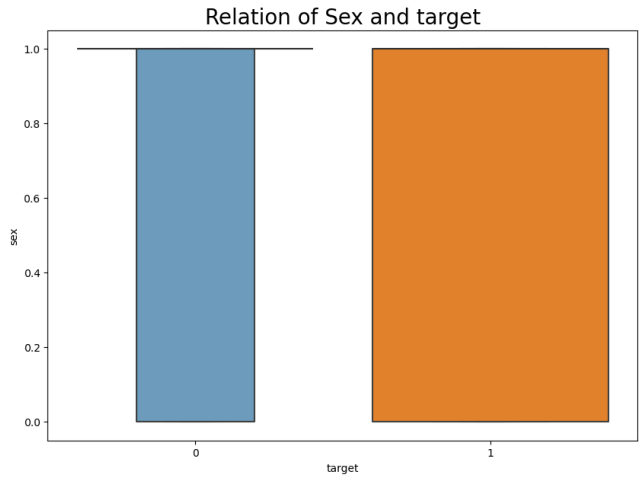

```
plt.rcParams['figure.figsize'] = (10, 7)

# Assuming 'target' is a categorical variable and 'trestbps' is a numerical variable
sns.violinplot(x='target', y='trestbps', data=data)
plt.title('Violinplot of trestbps for each target category', fontsize=20)
plt.show()
```



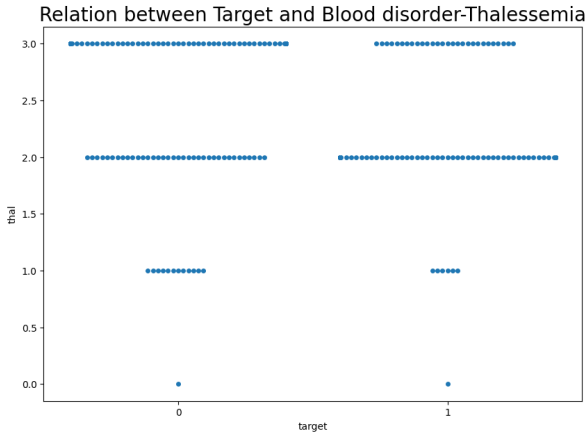
sns.swarmplot(x = 'target',y ='age', data = data)
plt.title('Relation of Age and target', fontsize = 20)



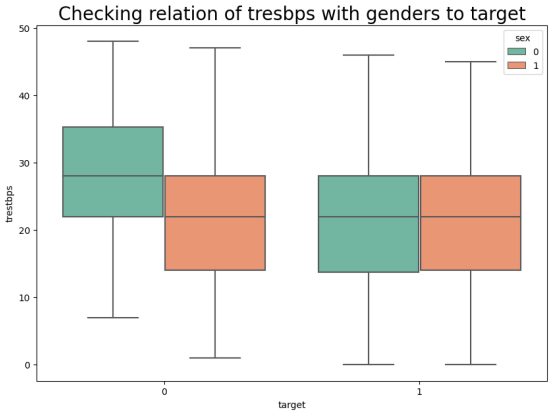
sns.boxenplot(x = 'target', y='sex', data=data)
plt.title('Relation of Sex and target', fontsize = 20)



sns.swarmplot(x ='target', y='thal', data=data)
plt.title('Relation between Target and Blood disorder-Thalessemia', fontsize = 20)




```
sns.boxplot(x = data['target'], y = data['trestbps'], hue = data['sex'], palette = 'Set2')
plt.title('Checking relation of tresbps with genders to target', fontsize = 20)
```




```
X = data.drop(columns = 'target' , axis = 1)
Y = data[‘target']
X_train , X_test , Y_train , Y_test = train_test_split(X,Y, test_size = 0.2 , stratify=Y ,
random_state =2)
print("Shape of x_train :", X_train.shape)
print("Shape of x_test :", X_test.shape)
print("Shape of y_train :", Y_train.shape)
print("Shape of y_test :", Y_test.shape)
model = LogisticRegression()
model.fit(X_train , Y_train)
```

```
X_train_pred = model.predict(X_train)
```

```
training_data_acc = accuracy_score(X_train_pred , Y_train)
X_test_pred = model.predict(X_test)
test_data_acc = accuracy_score(X_test_pred , Y_test)

input_data = (57 , 0 , 0 ,120 , 354 , 0 , 1 , 163 , 1 , 0.6 , 2 , 0 ,2 )

in_data_nparr = np.asarray(input_data)
in_data_reshape = in_data_nparr.reshape(1 , -1)

prediction = model.predict(in_data_reshape)
print(prediction)

if (prediction[0]==0):
  print("Dont Have HEART disease :)")

else :
  print("You have a HEART CONDITION :( “)

f1 = f1_score(Y_test , X_test_pred)
print(f1)
recall = recall_score(Y_test, X_test_pred)
precision = precision_score(Y_test, X_test_pred)
fpr, tpr, _ = roc_curve(Y_test, model.decision_function(X_test))
roc_auc = auc(fpr, tpr)

from sklearn.metrics import confusion_matrix
cm = confusion_matrix(Y_test, X_test_pred)
plt.rcParams['figure.figsize'] = (5, 5)
sns.heatmap(cm, annot = True, annot_kws = {‘size':15})
```