Heart Disease Prediction

Project for Machine Learning module

Institute Name- Itvedant Education Pvt. Ltd
Name- Sushant Baliram Chavan

Email Address- chavansushant92@gmail.com

Date of Submission- 20 December 2023

Project Description

This project aims to predict the presence or absence of heart disease based on various medical and clinical features. The dataset includes information such as age, sex, chest pain type, resting blood pressure, cholesterol levels, resting electrocardiographic results, maximum heart rate achieved during exercise, presence of exercise-induced angina, oldpeak, ST slope, and the target variable 'HeartDisease.'

Dataset Information

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
df = pd.read_csv(r"C:\Users\chava\Desktop\heart_disease_prediction.csv")
df
```

	Age	Sex	ChestPainType	RestingBP	Cholesterol	RestingECG	MaxHR	ExerciseAngina	Oldpeak	ST_Slope	HeartDisease
0	40	M	ATA	140	289	Normal	172	No	0.0	Up	Absence
1	49	F	NAP	160	180	Normal	156	No	1.0	Flat	Presence
2	37	M	ATA	130	283	ST	98	No	0.0	Up	Absence
3	48	F	ASY	138	214	Normal	108	Yes	1.5	Flat	Presence
4	54	M	NAP	150	195	Normal	122	No	0.0	Up	Absence
...	...	...	...	...	...	...	...	...	...	...	...
913	45	M	TA	110	264	Normal	132	No	1.2	Flat	Presence
914	68	M	ASY	144	193	Normal	141	No	3.4	Flat	Presence
915	57	M	ASY	130	131	Normal	115	Yes	1.2	Flat	Presence
916	57	F	ATA	130	236	LVH	174	No	0.0	Flat	Presence
917	38	M	NAP	138	175	Normal	173	No	0.0	Up	Absence

Features

1. Age
2. Sex
3. Chest Pain Type
4. Resting Blood Pressure
5. Cholesterol
6. Resting Electrocardiographic Results
7. Maximum Heart Rate Achieved (MaxHR)
8. Exercise-Induced Angina
9. Oldpeak

10. ST Slope

Target Variable

- HeartDisease

Data Preprocessing

- Checked for missing values.

i found ? in RestingBP and Cholesterol column, so i replace the ? with NAN because pandas and sklearn only handle NaN values

```
df["RestingBP"].replace("?", np.nan, inplace=True)
df["Cholesterol"].replace("?", np.nan, inplace=True)
```

change the datatype because in the latest steps we need to get mean value of RestingBP and Cholesterol column

```
df["RestingBP"] = df["RestingBP"].astype("float64")
df["Cholesterol"] = df["Cholesterol"].astype("float64")
```

- Replace zero values with mean of each independent column.

from describe i found that RestingBP has min 0 value, but RestingBP must be greater than zero,so i need to 1st replace 0 value with NAN and then again NAN repalce with mean of RestingBP

```
df["RestingBP"].replace(0, np.nan, inplace=True)
```

```
df["RestingBP"].replace(np.nan,df["RestingBP"].mean(),inplace=True)
```

- Doing scaling of numerical column if need.

in oldpeak column positive and negative value between -2.60 to 6.20, so we need to scaling this column

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
df[['Oldpeak']] = scaler.fit_transform(df[['Oldpeak']])
```

- Preprocessing: Convert categorical variables to numerical by Label Encoder.

Preprocessing: Convert categorical variables to numerical

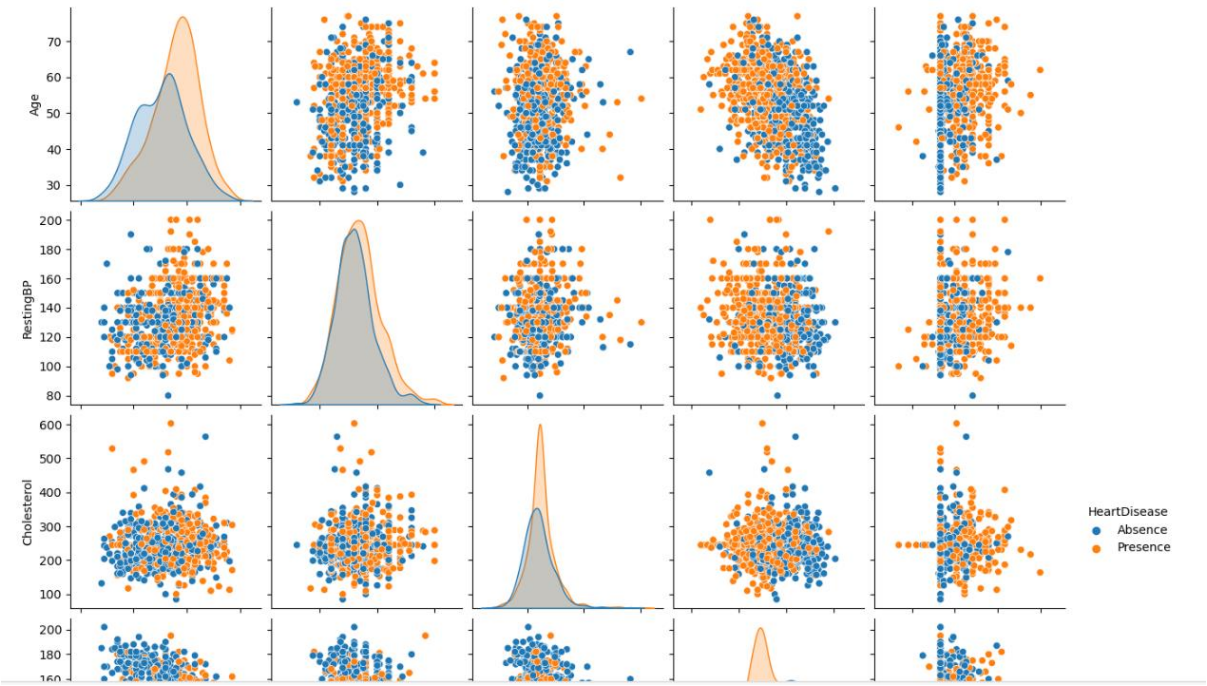
```
le = LabelEncoder()
df['Sex'] = le.fit_transform(df['Sex'])
df['ChestPainType'] = le.fit_transform(df['ChestPainType'])
df['RestingECG'] = le.fit_transform(df['RestingECG'])
df['ExerciseAngina'] = le.fit_transform(df['ExerciseAngina'])
df['ST_Slope'] = le.fit_transform(df['ST_Slope'])
```

- Separate features (X) and target variable (y)

```
X = df.drop('HeartDisease', axis=1)
y = df['HeartDisease']
```

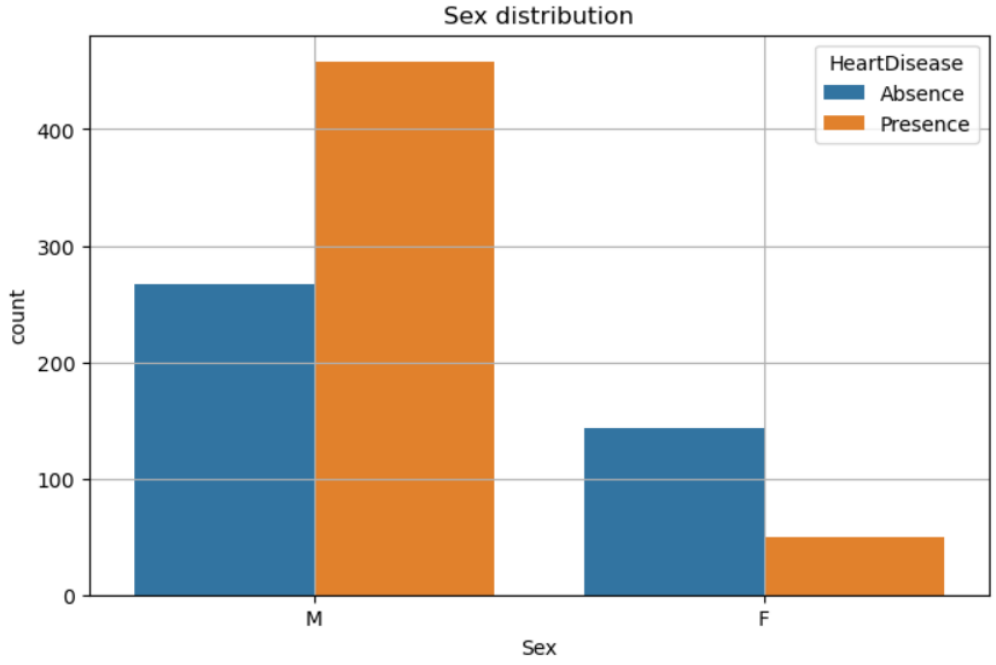
data visualization for numerical dataset

```
sns.pairplot(df[['Age', 'RestingBP', 'Cholesterol', 'MaxHR', 'Oldpeak', 'HeartDisease']], hue='HeartDisease')
plt.show()
```



data visualization for categorical dataset

```
categorical_columns = ['Sex', 'ChestPainType', 'RestingECG', 'ExerciseAngina', 'ST_Slope']
for column in categorical_columns:
    plt.figure(figsize=(8, 5))
    sns.countplot(x=column, hue='HeartDisease', data=df)
    plt.title(f'{column} distribution')
    plt.grid()
    plt.show()
```



Logistic Regression:

- import important libraries for logistic regression

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

- Split the data into training and testing sets, make predictions on the test set

Split the data into training and testing sets

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Initialize and train the logistic regression model

```
model = LogisticRegression()
model.fit(X_train, y_train)
```

```
LogisticRegression()
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

- Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)
```

```
print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.8260869565217391
Confusion Matrix:
[[68  9]
 [23 84]]
Classification Report:
              precision    recall  f1-score   support

      Absence       0.75      0.88      0.81        77
     Presence       0.90      0.79      0.84       107

     accuracy                           0.83       184
    macro avg       0.83      0.83      0.82       184
 weighted avg       0.84      0.83      0.83       184
```

Decision Trees:

- import important libraries for DecisionTreeClassifier

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

- Split the data into training and testing sets, make predictions on the test set.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Initialize and train the decision tree model

```
model = DecisionTreeClassifier(random_state=42)
model.fit(X_train, y_train)
model = DecisionTreeClassifier(max_depth=5, min_samples_leaf=5, random_state=42)
model.fit(X_train, y_train)
```

```
DecisionTreeClassifier(max_depth=5, min_samples_leaf=5, random_state=42)
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

- Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.8478260869565217
Confusion Matrix:
[[69  8]
 [20 87]]
Classification Report:
              precision    recall  f1-score   support

     Absence       0.78      0.90      0.83        77
    Presence       0.92      0.81      0.86       107

    accuracy                           0.85       184
   macro avg       0.85      0.85      0.85       184
weighted avg       0.86      0.85      0.85       184
```

Support Vector Machines (SVM):

- import important libraries for Support Vector Machines.

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix
from sklearn.preprocessing import StandardScaler, LabelEncoder
```

- Split the data into training and testing sets, make predictions on the test set.

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
```

Standardize the features (SVMs are sensitive to feature scaling)

```
scaler = StandardScaler()
X_train = scaler.fit_transform(X_train)
X_test = scaler.transform(X_test)
```

Initialize and train the SVM model

```
model = SVC(C=1.0, gamma='scale', random_state=42)
model.fit(X_train, y_train)
```

```
SVC(random_state=42)
```

Make predictions on the test set

```
y_pred = model.predict(X_test)
```

- Evaluate the model

```
accuracy = accuracy_score(y_test, y_pred)
conf_matrix = confusion_matrix(y_test, y_pred)
classification_rep = classification_report(y_test, y_pred)

print(f'Accuracy: {accuracy}')
print(f'Confusion Matrix:\n{conf_matrix}')
print(f'Classification Report:\n{classification_rep}')
```

```
Accuracy: 0.8369565217391305
Confusion Matrix:
[[64 13]
 [17 90]]
Classification Report:
              precision    recall  f1-score   support

     Absence       0.79      0.83      0.81        77
    Presence       0.87      0.84      0.86       107

    accuracy                           0.84       184
   macro avg       0.83      0.84      0.83       184
weighted avg       0.84      0.84      0.84       184
```

Model Building

Machine Learning Algorithms Used

1. Logistic Regression
    - Accuracy: 82%
2. Decision Tree
    - Accuracy: 84%
3. Support Vector Machine (SVM)
    - Accuracy: 83%

Model Evaluation Metrics

- Confusion matrix
- Precision, Recall, F1-Score
- ROC-AUC (if applicable)

Conclusion

In this project, we successfully applied three different machine learning algorithms—Logistic Regression, Decision Tree, and Support Vector Machine—to predict heart disease. Each model was evaluated based on accuracy and other relevant metrics. The Decision Tree model showed the highest accuracy among the three.

Future Work

- Hyperparameter tuning for each model
- Feature engineering to improve model performance
- Exploration of additional machine learning algorithms
- Deployment of the selected model for real-world predictions

Acknowledgments

The successful completion of this project was made possible through the invaluable guidance and support of our class teacher Mr.Sameer Warsolkar. Their expertise, encouragement, and commitment to fostering a deep understanding of machine learning concepts have been instrumental in our journey.

Additionally, the project owes its efficiency and efficacy to the extensive use of Python's rich ecosystem of libraries and algorithms. i express my gratitude to the developers and contributors of scikit-learn, the library that provided us with seamless access to machine learning algorithms such as Logistic Regression, Decision Tree, and Support Vector Machine. The user-friendly interfaces and comprehensive documentation of these algorithms facilitated their implementation and evaluation in my project.Our class teacher's guidance, coupled with the power and versatility of Python's machine learning libraries, has played a pivotal role in shaping our understanding and enhancing the practical application of machine learning concepts. This project stands as a testament to the collaborative efforts of both human mentorship and technological resources.