PYTHON WITH DATA SCIENCE

MAJOR PROJECT

EDA (Exploratory Data Analysis)

TEACHNOOK APRIL-MAY

BATCH

—

ABIL BIJU

abilbiju2004@gmail.com

—

22/05/2023

Major Project

Take any Dataset of your choice ,perform EDA(Exploratory Data Analysis) and apply a suitable Classifier, Regressor or Clusterer and calculate the accuracy of the model.

PROGRAM

```
# -*- coding: utf-8 -*-
#HEARTRATE_PREDICTION.ipynb


#Original file is located
at(https://colab.research.google.com/drive/1uK255DwmxbqeOuufmSazmvBgp6ZsLDcO)

#HERE I HAVE USED AHEART RATE CSV
FILE(https://raw.githubusercontent.com/abilbiju/DATASETS/main/heartrate_prediction.csv) FOR PERFORMING THE EDA ANALYSIS AND PREDICTION.

#STEP1 :LOADING THE DATASET

import pandas as pd

# Load the dataset
url =
'https://raw.githubusercontent.com/abilbiju/DATASETS/main/heartrate_prediction.csv'
data = pd.read_csv(url)

# Display the first few rows of the dataset
data.head()

#STEP 2: EDA ANALYSIS

# Import necessary libraries for EDA
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# Check the dimensions of the dataset
print("Dataset Dimensions: ", data.shape)

# Get an overview of the dataset
print("\nDataset Information:")
data.info()

# Check for missing values
```

```
print("\nMissing Values:")
print(data.isnull().sum())

# Summary statistics
print("\nSummary Statistics:")
print(data.describe())

# Correlation matrix
correlation_matrix = data.corr()
plt.figure(figsize=(10, 8))
sns.heatmap(correlation_matrix, annot=True, cmap="YlGnBu")
plt.title("Correlation Matrix")
plt.show()

# Distribution of the target variable
plt.figure(figsize=(8, 6))
sns.histplot(data['target'], kde=True)
plt.title("Distribution of Target Variable")
plt.xlabel("Target Variable")
plt.ylabel("Count")
plt.show()

#Step 3: Preparing the Data for Modeling

from sklearn.model_selection import train_test_split

# Splitting the data into features and target variable
X = data.drop('target', axis=1)
y = data['target']

# Splitting 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)

#Step 4: Model Selection, Training, and Evaluation

from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Create the classifier
classifier = RandomForestClassifier(random_state=42)

# Train the model
classifier.fit(X_train, y_train)
```

```
# Predict on the test set
y_pred = classifier.predict(X_test)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)

#An accuracy score of 1.0 indicates that the model achieved a perfect prediction on the test set.
```

CODE WITH OUTPUT

HERE I HAVE USED AHEART RATE CSV FILE(https://raw.githubusercontent.com/abilbiju/DATASETS/main/heartrate_prediction.csv) FOR PERFORMING THE EDA ANALYSIS AND PREDICTION.

STEP1 :LOADING THE DATASET

```
import pandas as pd

# Load the dataset
url = 'https://raw.githubusercontent.com/abilbiju/DATASETS/main/heartrate_prediction.csv'
data = pd.read_csv(url)

# Display the first few rows of the dataset
data.head()
```

	Unnamed: 0	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

STEP 2: EDA ANALYSIS

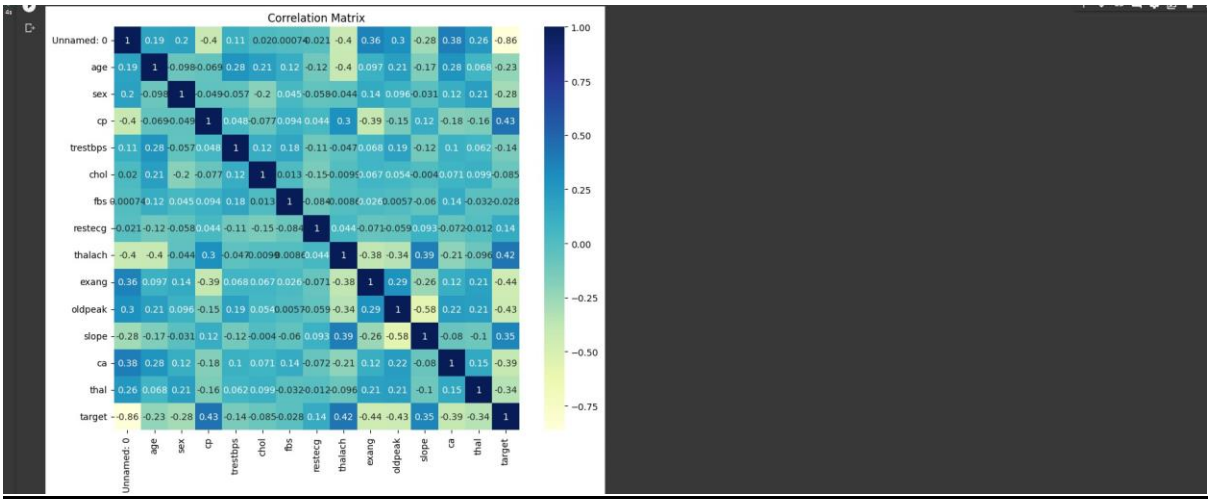
```
[7] # Import necessary libraries for EDA
    import numpy as np
    import seaborn as sns
    import matplotlib.pyplot as plt

    # Check the dimensions of the dataset
    print("Dataset Dimensions: ", data.shape)

    # Get an overview of the dataset
    print("\nDataset Information:")
    data.info()

    # Check for missing values
    print("\nMissing Values:")
    print(data.isnull().sum())

    # Summary statistics
    print("\nSummary Statistics:")
    print(data.describe())
```

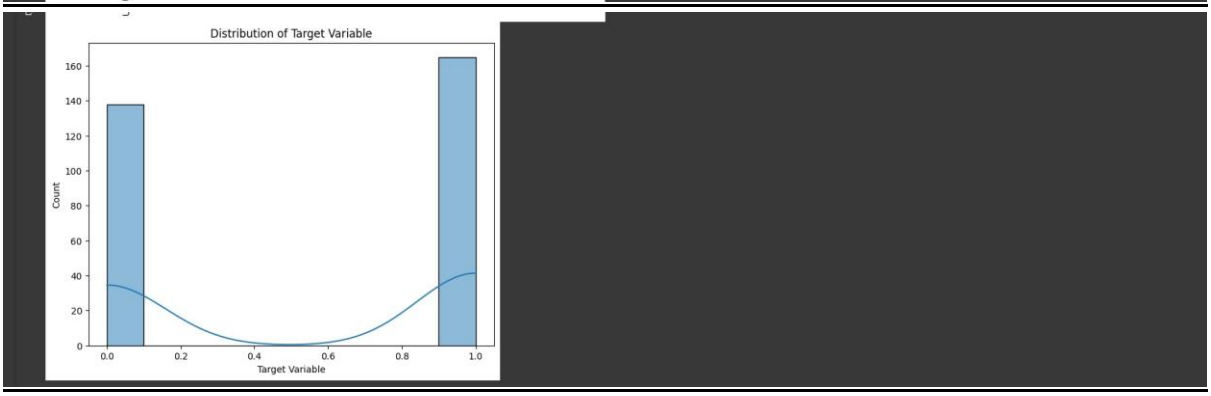
```
[7] # Correlation matrix
    correlation_matrix = data.corr()
    plt.figure(figsize=(10, 8))
    sns.heatmap(correlation_matrix, annot=True, cmap="YlGnBu")
    plt.title("Correlation Matrix")
    plt.show()

    # Distribution of the target variable
    plt.figure(figsize=(8, 6))
    sns.histplot(data['target'], kde=True)
    plt.title("Distribution of Target Variable")
    plt.xlabel("Target Variable")
    plt.ylabel("Count")
    plt.show()
```

```
[7] Dataset Dimensions:  (303, 15)

    Dataset Information:
    <class 'pandas.core.frame.DataFrame'>
    RangeIndex: 303 entries, 0 to 302
    Data columns (total 15 columns):
     #   Column      Non-Null Count  Dtype
    ---  ------      --------------  -----
     0   Unnamed: 0  303 non-null    int64
     1   age         303 non-null    int64
     2   sex         303 non-null    int64
     3   cp          303 non-null    int64
     4   trestbps    303 non-null    int64
     5   chol        303 non-null    int64
     6   fbs         303 non-null    int64
     7   restecg     303 non-null    int64
     8   thalach     303 non-null    int64
     9   exang       303 non-null    int64
     10  oldpeak     303 non-null    float64
     11  slope       303 non-null    int64
     12  ca          303 non-null    int64
     13  thal        303 non-null    int64
     14  target      303 non-null    int64
    dtypes: float64(1), int64(14)
    memory usage: 35.6 KB
```

```
Missing Values:
Unnamed: 0    0
age           0
sex           0
cp            0
trestbps      0
chol          0
fbs           0
restecg       0
thalach       0
exang         0
oldpeak       0
slope         0
ca            0
thal          0
target        0
dtype: int64
```

```
Summary Statistics:
       Unnamed: 0         age         sex          cp    trestbps        chol  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean   151.000000   54.366337    0.683168    0.966997  131.623762  246.264026
std     87.612784    9.082101    0.466011    1.032052   17.538143   51.830751
min      0.000000   29.000000    0.000000    0.000000   94.000000  126.000000
25%     75.500000   47.500000    0.000000    0.000000  120.000000  211.000000
50%    151.000000   55.000000    1.000000    1.000000  130.000000  240.000000
75%    226.500000   61.000000    1.000000    2.000000  140.000000  274.500000
max    302.000000   77.000000    1.000000    3.000000  200.000000  564.000000

              fbs     restecg     thalach       exang     oldpeak       slope  \
count  303.000000  303.000000  303.000000  303.000000  303.000000  303.000000
mean     0.148515    0.528053  149.646865    0.326733    1.039604    1.399340
std      0.356198    0.525860   22.905161    0.469794    1.161075    0.616226
min      0.000000    0.000000   71.000000    0.000000    0.000000    0.000000
25%      0.000000    0.000000  133.500000    0.000000    0.000000    1.000000
50%      0.000000    1.000000  153.000000    0.000000    0.800000    1.000000
75%      0.000000    1.000000  166.000000    1.000000    1.600000    2.000000
max      1.000000    2.000000  202.000000    1.000000    6.200000    2.000000

               ca        thal      target
count  303.000000  303.000000  303.000000
mean     0.729373    2.313531    0.544554
std      1.022606    0.612277    0.498835
min      0.000000    0.000000    0.000000
25%      0.000000    2.000000    0.000000
50%      0.000000    2.000000    1.000000
75%      1.000000    3.000000    1.000000
max      4.000000    3.000000    1.000000
```





Step 3: Preparing the Data for Modeling

```
[9] from sklearn.model_selection import train_test_split

    # Splitting the data into features and target variable
    X = data.drop('target', axis=1)
    y = data['target']

    # Splitting 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)
```

Step 4: Model Selection, Training, and Evaluation

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

# Create the classifier
classifier = RandomForestClassifier(random_state=42)

# Train the model
classifier.fit(X_train, y_train)

# Predict on the test set
y_pred = classifier.predict(X_test)

# Model evaluation
accuracy = accuracy_score(y_test, y_pred)

print("Accuracy:", accuracy)
```

```
Accuracy: 1.0
```

An accuracy score of 1.0 indicates that the model achieved a perfect prediction on the test set.