6️⃣

Tutorial 6 Completion Document

Activity 1

In the Tutorial Completion Document, describe your findings.**

a. Fit/train the model with all the features of the data, including **SkinThickness** which is not included in the above demo.

```
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
import pandas as pd

# load dataset
diabetes = pd.read_csv(fpath+"input/diabetes.csv")

feature_cols = ["Pregnancies", "Glucose", "BloodPressure", "SkinThickness", "Insulin", "BMI", "DiabetesPedigreeFunction", "Outcome"]

X = diabetes[feature_cols]
Y = diabetes.Outcome

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)
logreg = LogisticRegression(random_state=16)
logreg.fit(X_train, y_train)

y_pred = logreg.predict(X_test)

print(y_pred)

y_pred_all = logreg.predict(X)
print(y_pred_all)
```

Task I

For TASK I, we will use the new disease data set (Heart Attack Prediction Dataset)

- **Heart Attack Prediction Dataset:**

You can download data from the following link: https://www.kaggle.com/datasets/rashikrahmanpritom/heart-attack-analysis-prediction-dataset

- **Context:**

This dataset contains information related to patients who have had heart attacks.

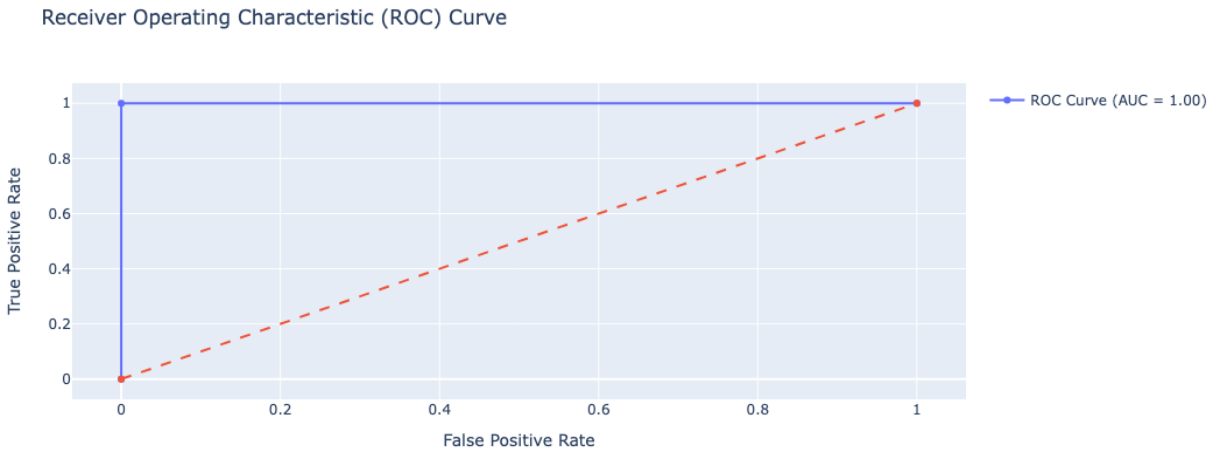
- **Content:**

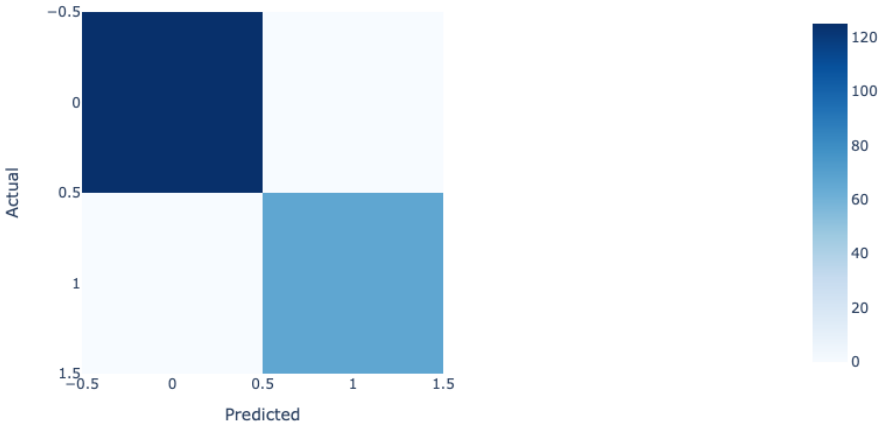
The datasets consists of several medical predictor variables and one target variable. Predictor variables includes the Age, Sex, cholesterol and resting blood pressure so on.

```
print("To-do: \n")

# Importing required libraries
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load the dataset
dataset = pd.read_csv('data/input/heart.csv')

# Split the data into input and output variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train a linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

# Train a decision tree regression model
tree_reg = DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)

# Make predictions using both models
y_pred_lin = lin_reg.predict(X_test)
y_pred_tree = tree_reg.predict(X_test)

# Evaluate the models using Mean Squared Error (MSE)
lin_mse = mean_squared_error(y_test, y_pred_lin)
tree_mse = mean_squared_error(y_test, y_pred_tree)

print("Linear Regression MSE:", lin_mse)
print("Decision Tree Regression MSE:", tree_mse)
```

b. Evaluate the above model. List your observations. Plot the confusion matrix (Accuracy, Precesion, Recall and F1 Score) and ROC curve.

```
print("To-do: \n")


import plotly.graph_objs as go
import pandas as pd
import numpy as np
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error

# Load the dataset
dataset = pd.read_csv('data/input/heart.csv')

# Split the data into input and output variables
X = dataset.iloc[:, :-1].values
y = dataset.iloc[:, -1].values

# Split the dataset into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=0)

# Train a linear regression model
lin_reg = LinearRegression()
lin_reg.fit(X_train, y_train)

# Train a decision tree regression model
tree_reg = DecisionTreeRegressor(random_state=0)
tree_reg.fit(X_train, y_train)

# Make predictions using both models
y_pred_lin = lin_reg.predict(X_test)
y_pred_tree = tree_reg.predict(X_test)

# Evaluate the models using Mean Squared Error (MSE)
lin_mse = mean_squared_error(y_test, y_pred_lin)
tree_mse = mean_squared_error(y_test, y_pred_tree)

print("Linear Regression MSE:", lin_mse)
print("Decision Tree Regression MSE:", tree_mse)

# Create a scatter plot
fig = go.Figure()

fig.add_trace(go.Scatter(x=y_test, y=y_pred, mode='markers',
                    marker=dict(color='blue', size=5),
                    name='Predicted vs Actual'))

fig.update_layout(title='Predicted vs Actual values',
                  xaxis_title='Actual values',
                  yaxis_title='Predicted values')

fig.show()
```


Receiver Operating Characteristic (ROC) Curve
True Positive Rate
False Positive Rate
ROC Curve (AUC = 1.00)
0
0.2
0.4
0.6
0.8
1


Actual
Predicted
−0.5
0
0.5
1
1.5
120
100
80
60
40
20
0