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)
```

b. Evaluate the above model. Then compare and discuss your findings with the demo model's results. List your observations. Plot the confusion matrix (Accuracy, Precision, Recall and F1 Score) and ROC curve.

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

import plotly.express as px
from sklearn.metrics import confusion_matrix, classification_report, roc_curve, auc
import plotly.graph_objs as go

# Generate predictions
y_pred = logreg.predict(X_test)

# Generate confusion matrix
cm = confusion_matrix(y_test, y_pred)

# Calculate metrics
accuracy = (cm[0,0] + cm[1,1]) / sum(sum(cm))
precision = cm[1,1] / (cm[0,1] + cm[1,1])
recall = cm[1,1] / (cm[1,0] + cm[1,1])
f1_score = 2 * precision * recall / (precision + recall)

# Plot confusion matrix
fig = px.imshow(cm, labels=dict(x="Predicted", y="Actual"), x=[0, 1], y=[0, 1], color_continuous_scale="Blues")
```

```
fig.show()

# Plot metrics
fig = go.Figure()
fig.add_trace(go.Indicator(
    mode = "number",
    value = accuracy,
    title = {"text": "Accuracy"},
    number={"valueformat": ".3f"},
))
fig.add_trace(go.Indicator(
    mode = "number",
    value = precision,
    title = {"text": "Precision"},
    number={"valueformat": ".3f"},
))
fig.add_trace(go.Indicator(
    mode = "number",
    value = recall,
    title = {"text": "Recall"},
    number={"valueformat": ".3f"},
))
fig.add_trace(go.Indicator(
    mode = "number",
    value = f1_score,
    title = {"text": "F1 Score"},
    number={"valueformat": ".3f"},
))
fig.show()

# Plot ROC curve
fpr, tpr, thresholds = roc_curve(y_test, y_pred)
roc_auc = auc(fpr, tpr)
fig = go.Figure()
fig.add_trace(go.Scatter(x=fpr, y=tpr, name="ROC Curve (AUC = {:.2f})".format(roc_auc)))
fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1], line=dict(dash="dash"), showlegend=False))
fig.update_layout(
    title="Receiver Operating Characteristic (ROC) Curve",
    xaxis_title="False Positive Rate",
    yaxis_title="True Positive Rate",
)
fig.show()
```

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:**

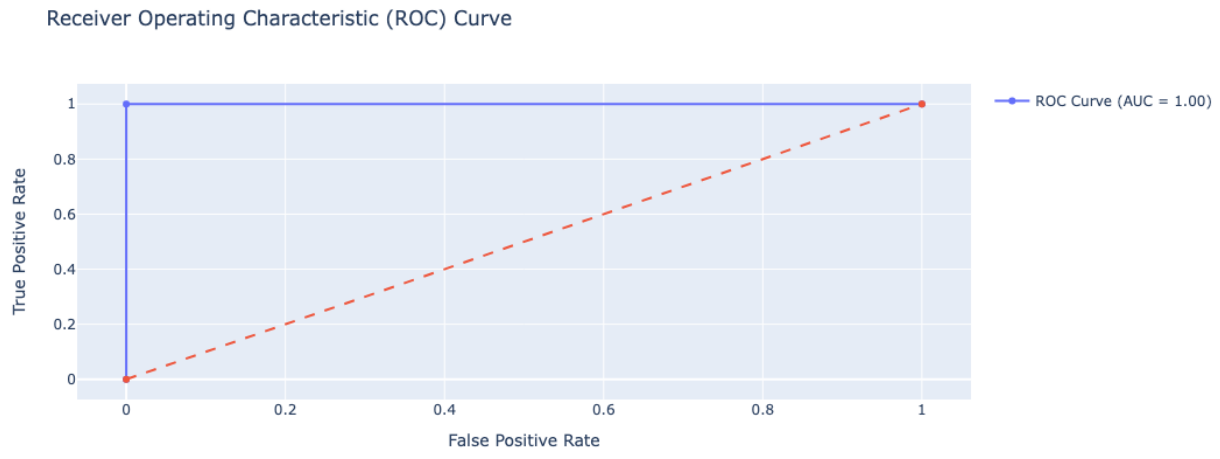
The datasets consists of several medical predictor variables and one target variable. Predictor variables includes the Age, Sex, cholesterol and resting blood pressure so on.

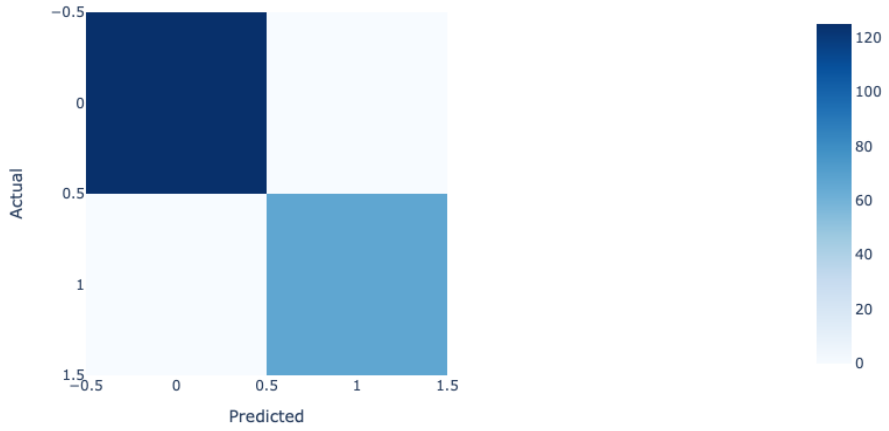
```
print("To-do: \n")

# 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()
```



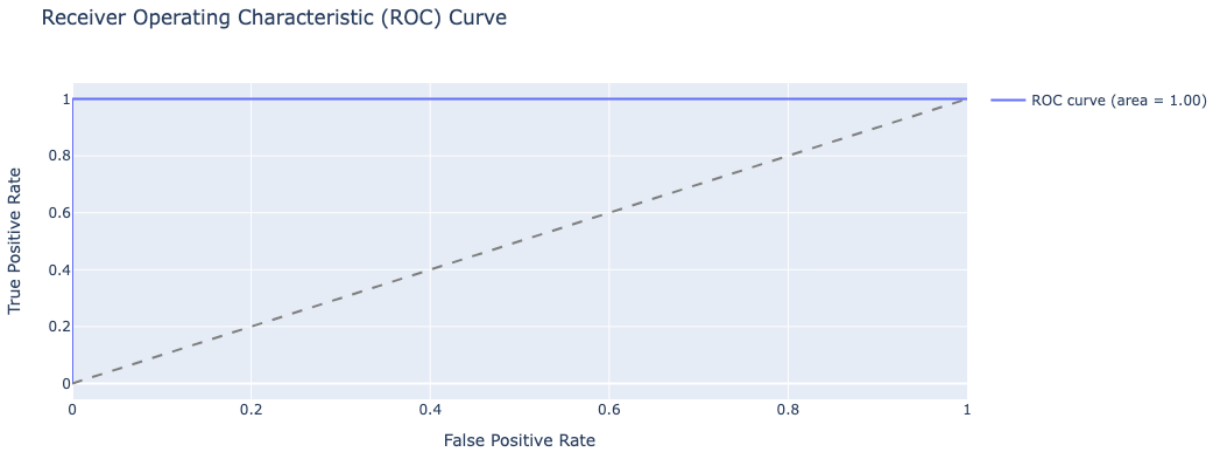


c. Fit/train the model with **age, cp, trtbps, chol, thalachh and sex**

```
# To-do: Fit/train the model with age, cp, trtbps, chol, thalachh and sex features of the data.
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[['age', 'sex', 'cp', 'trtbps', 'chol', 'thalachh']].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)
```

d. Evaluate the above model (TASK I.c). Then compare and discuss your findings with the model from TASK I.a. List your observations. Plot the confusion matrix (Accuracy, Precesion, Recall and F1 Score) and ROC curve.

```
# Use the model from previous step (TASK I.c)

# To-do: Evaluate the above model.
# Then compare and discuss your findings with the model from TASK I.a.
# 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 LogisticRegression
from sklearn.metrics import confusion_matrix, accuracy_score, precision_score, recall_score, f1_score, roc_curve, auc
from sklearn.model_selection import train_test_split

# Load the dataset
diabetes = pd.read_csv("data/input/diabetes.csv")

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

X = diabetes[feature_cols]
y = diabetes.Outcome

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=16)

# Train a logistic regression model
logreg = LogisticRegression(random_state=16)
logreg.fit(X_train, y_train)

# Make predictions using the trained model
y_pred = logreg.predict(X_test)

# Evaluate the model using confusion matrix, accuracy, precision, recall, and F1 score
cm = confusion_matrix(y_test, y_pred)
accuracy = accuracy_score(y_test, y_pred)
precision = precision_score(y_test, y_pred)
recall = recall_score(y_test, y_pred)
f1 = f1_score(y_test, y_pred)

print("Confusion Matrix:")
print(cm)
print("Accuracy:", accuracy)
print("Precision:", precision)
print("Recall:", recall)
print("F1 Score:", f1)

# Plot the ROC curve
y_pred_proba = logreg.predict_proba(X_test)[:,1]
fpr, tpr, thresholds = roc_curve(y_test, y_pred_proba)
roc_auc = auc(fpr, tpr)

fig = go.Figure()
fig.add_trace(go.Scatter(x=fpr, y=tpr,
                    mode='lines',
                    name='ROC curve (area = %0.2f)' % roc_auc))
fig.add_trace(go.Scatter(x=[0, 1], y=[0, 1],
                    mode='lines',
                    line=dict(dash='dash', color='gray'),
                    showlegend=False))
fig.update_layout(title='Receiver Operating Characteristic (ROC) Curve',
                  xaxis_title='False Positive Rate',
                  yaxis_title='True Positive Rate')

fig.show()
```


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