import pandas as pd

from sklearn.model\_selection import train\_test\_split

from sklearn.preprocessing import LabelEncoder, StandardScaler

from sklearn.linear\_model import LogisticRegression

from sklearn.tree import DecisionTreeClassifier

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import accuracy\_score, roc\_curve, auc

import matplotlib.pyplot as plt

import seaborn as sns

# Load the CSV file into a pandas DataFrame

data = pd.read\_csv("./lungDisease.csv")

# Encode categorical variables

label\_encoder = LabelEncoder()

categorical\_columns = ['GENDER', 'SMOKING', 'YELLOW\_FINGERS', 'ANXIETY', 'PEER\_PRESSURE',

'CHRONIC\_DISEASE', 'FATIGUE ', 'ALLERGY ', 'WHEEZING',

'ALCOHOL CONSUMING', 'COUGHING', 'SHORTNESS OF BREATH',

'SWALLOWING DIFFICULTY', 'CHEST PAIN']

for col in categorical\_columns:

data[col] = label\_encoder.fit\_transform(data[col])

# Encode target variable (LUNG\_CANCER) to binary values

data['LUNG\_CANCER'] = label\_encoder.fit\_transform(data['LUNG\_CANCER'])

# Split the data into features (X) and target variable (y)

X = data.drop('LUNG\_CANCER', axis=1)

y = data['LUNG\_CANCER']

# 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)

# Standardize the features

scaler = StandardScaler()

X\_train = scaler.fit\_transform(X\_train)

X\_test = scaler.transform(X\_test)

# Train a Logistic Regression classifier

logistic\_regression = LogisticRegression()

logistic\_regression.fit(X\_train, y\_train)

# Make predictions on the test set for Logistic Regression

y\_pred\_logistic\_regression = logistic\_regression.predict(X\_test)

# Calculate accuracy for Logistic Regression

accuracy\_logistic\_regression = accuracy\_score(y\_test, y\_pred\_logistic\_regression)

print("Logistic Regression Accuracy:", accuracy\_logistic\_regression)

# Calculate ROC curve and AUC score for Logistic Regression

fpr\_logistic\_regression, tpr\_logistic\_regression, \_ = roc\_curve(y\_test, y\_pred\_logistic\_regression)

roc\_auc\_logistic\_regression = auc(fpr\_logistic\_regression, tpr\_logistic\_regression)

# Train a Decision Tree classifier

decision\_tree = DecisionTreeClassifier()

decision\_tree.fit(X\_train, y\_train)

# Make predictions on the test set for Decision Tree

y\_pred\_decision\_tree = decision\_tree.predict(X\_test)

# Calculate accuracy for Decision Tree

accuracy\_decision\_tree = accuracy\_score(y\_test, y\_pred\_decision\_tree)

print("Decision Tree Accuracy:", accuracy\_decision\_tree)

# Calculate ROC curve and AUC score for Decision Tree

fpr\_decision\_tree, tpr\_decision\_tree, \_ = roc\_curve(y\_test, y\_pred\_decision\_tree)

roc\_auc\_decision\_tree = auc(fpr\_decision\_tree, tpr\_decision\_tree)

# Train a Random Forest classifier

random\_forest = RandomForestClassifier()

random\_forest.fit(X\_train, y\_train)

# Make predictions on the test set for Random Forest

y\_pred\_random\_forest = random\_forest.predict(X\_test)

# Calculate accuracy for Random Forest

accuracy\_random\_forest = accuracy\_score(y\_test, y\_pred\_random\_forest)

print("Random Forest Accuracy:", accuracy\_random\_forest)

# Calculate ROC curve and AUC score for Random Forest

fpr\_random\_forest, tpr\_random\_forest, \_ = roc\_curve(y\_test, y\_pred\_random\_forest)

roc\_auc\_random\_forest = auc(fpr\_random\_forest, tpr\_random\_forest)

# Plot ROC curves

plt.figure(figsize=(8, 6))

plt.plot(fpr\_logistic\_regression, tpr\_logistic\_regression, color='blue', lw=2, label=f'Logistic Regression (AUC = {roc\_auc\_logistic\_regression:.2f})')

plt.plot(fpr\_decision\_tree, tpr\_decision\_tree, color='darkorange', lw=2, label=f'Decision Tree (AUC = {roc\_auc\_decision\_tree:.2f})')

plt.plot(fpr\_random\_forest, tpr\_random\_forest, color='green', lw=2, label=f'Random Forest (AUC = {roc\_auc\_random\_forest:.2f})')

plt.plot([0, 1], [0, 1], color='navy', lw=2, linestyle='--')

plt.xlim([0.0, 1.0])

plt.ylim([0.0, 1.05])

plt.xlabel('False Positive Rate')

plt.ylabel('True Positive Rate')

plt.title('Receiver Operating Characteristic (ROC) Curve')

plt.legend(loc="lower right")

plt.show()

##### Feature Correlation

#Plot a relationship graph between feature

palette = ["#1d7874","#679289","#f4c095","#ee2e31","#ffb563","#918450","#f85e00","#a41623","#9a031e","#d6d6d6","#ffee32","#ffd100","#333533","#202020"]

plt.subplots(figsize =(12, 9))

p=sns.heatmap(data.corr(), cmap = palette, square=True, cbar\_kws=dict(shrink =.99),

annot=True, vmin=-1, vmax=1, linewidths=0.1,linecolor='white',annot\_kws=dict(fontsize =6))

p.axes.set\_title("Pearson Correlation Of Features\n", fontsize=25)

plt.xticks(rotation=90)

plt.show()