import numpy as np

import pandas as pd

from sklearn.impute import SimpleImputer

from sklearn.preprocessing import StandardScaler

from sklearn.preprocessing import LabelEncoder

from sklearn.metrics import accuracy\_score

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

from sklearn.ensemble import RandomForestClassifier

# Load the dataset

data = pd.read\_csv('weatherAUS.csv')

# Select the features and target variable

features = data.iloc[:, [1, 2, 3, 4, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21]].values

target = data.iloc[:, -1].values

# Impute missing values with most frequent value

imputer = SimpleImputer(missing\_values=np.nan, strategy='most\_frequent')

features = imputer.fit\_transform(features)

# Encode categorical features

encoder1 = LabelEncoder()

features[:, 0] = encoder1.fit\_transform(features[:, 0])

encoder2 = LabelEncoder()

features[:, 4] = encoder2.fit\_transform(features[:, 4])

encoder3 = LabelEncoder()

features[:, 6] = encoder3.fit\_transform(features[:, 6])

encoder4 = LabelEncoder()

features[:, 7] = encoder4.fit\_transform(features[:, 7])

encoder5 = LabelEncoder()

features[:, -1] = encoder5.fit\_transform(features[:, -1])

# Encode target variable

encoder6 = LabelEncoder()

target = encoder6.fit\_transform(target)

# Scale the features

scaler = StandardScaler()

features = scaler.fit\_transform(features)

# Split the data into training and testing sets

train\_features, test\_features, train\_target, test\_target = train\_test\_split(features, target, test\_size=0.2, random\_state=0)

# Train and evaluate the Decision Tree Classifier

from sklearn.tree import DecisionTreeClassifier

dtc = DecisionTreeClassifier(random\_state=0)

dtc.fit(train\_features, train\_target)

dtc\_y\_pred = dtc.predict(test\_features)

dtc\_accuracy = accuracy\_score(test\_target, dtc\_y\_pred)

# Train and evaluate the Logistic Regression Classifier

from sklearn.linear\_model import LogisticRegression

lr = LogisticRegression(random\_state=0)

lr.fit(train\_features, train\_target)

lr\_y\_pred = lr.predict(test\_features)

lr\_accuracy = accuracy\_score(test\_target, lr\_y\_pred)

# Train and evaluate the KNN Classifier

from sklearn.neighbors import KNeighborsClassifier

knn = KNeighborsClassifier()

knn.fit(train\_features, train\_target)

knn\_y\_pred = knn.predict(test\_features)

knn\_accuracy = accuracy\_score(test\_target, knn\_y\_pred)

# Train and evaluate the Random Forest Classifier

from sklearn.ensemble import RandomForestClassifier

rfc = RandomForestClassifier(n\_estimators=100,random\_state=0)

rfc.fit(train\_features, train\_target)

rfc\_y\_pred = rfc.predict(test\_features)

rfc\_accuracy = accuracy\_score(test\_target, rfc\_y\_pred)

# Print the accuracies

print("Decision Tree Accuracy:", dtc\_accuracy)

print("Logistic Regression Accuracy:", lr\_accuracy)

print("KNN Accuracy:", knn\_accuracy)

print("Random Forest Accuracy:", rfc\_accuracy)

# Ask the user for input

print('Please provide the following information:')

location = input('Location: ')

min\_temp = float(input('Minimum temperature (in degrees Celsius): '))

max\_temp = float(input('Maximum temperature (in degrees Celsius): '))

rainfall = float(input('Rainfall (in mm): '))

wind\_gust\_dir = input('Wind gust direction: ')

wind\_gust\_speed = float(input('Wind gust speed (in km/h): '))

wind\_dir\_9am = input('Wind direction at 9am: ')

wind\_speed\_9am = float(input('Wind speed at 9am (in km/h): '))

humidity\_9am = float(input('Humidity at 9am: '))

pressure\_9am = float(input('Pressure at 9am (in hPa): '))

cloud\_9am = float(input('Cloud at 9am: '))

temp\_9am = float(input('Temperature at 9am (in degrees Celsius): '))

humidity\_3pm = float(input('Humidity at 3pm: '))

pressure\_3pm = float(input('Pressure at 3pm (in hPa): '))

cloud\_3pm = float(input('Cloud at 3pm: '))

temp\_3pm = float(input('Temperature at 3pm (in degrees Celsius): '))

rain\_today = input('Did it rain today? (yes or no): ')

# Encode the user input

location\_encoded = encoder1.transform([location])[0]

wind\_gust\_dir\_encoded = encoder2.transform([wind\_gust\_dir])[0]

wind\_dir\_9am\_encoded = encoder3.transform([wind\_dir\_9am])[0]

rain\_today\_encoded = encoder5.transform([rain\_today])[0] # Fix incorrect encoder index

user\_input = np.array([[location\_encoded, min\_temp, max\_temp, rainfall, wind\_gust\_dir\_encoded, wind\_gust\_speed, wind\_dir\_9am\_encoded, wind\_speed\_9am, humidity\_9am, pressure\_9am, cloud\_9am, temp\_9am, humidity\_3pm, pressure\_3pm, cloud\_3pm, temp\_3pm, rain\_today\_encoded, 0, 0]]) # Add missing columns and set dummy values

user\_input = scaler.transform(user\_input)

user\_input = user\_input.reshape(1, -1) # Convert to 2D array

prediction = rfc.predict(user\_input)

decoded\_prediction = encoder6.inverse\_transform(prediction)

print('The predicted rainfall for tomorrow is:', decoded\_prediction)