import pandas as pd

from sklearn.metrics

import accuracy\_score,confusion\_matrix,classification\_report

from sklearn.ensemble

import RandomForestClassifier, RandomForestRegressor

from sklearn.model\_selection

import train\_test\_split

import warnings

warnings.filterwarnings('ignore')

import matplotlib.pyplot as plt

import seaborn as sns

from sklearn.metrics

import mean\_absolute\_error, mean\_squared\_error, r2\_score, mean\_absolute\_percentage\_error

# Load a sample dataset (you can replace this with your own data)

data = pd.read\_csv('data1.csv')

data

from sklearn.preprocessing import LabelEncoder

# Instantiate LabelEncoder

le = LabelEncoder()

# Apply LabelEncoder to the 'crop' column

data['crop'] = le.fit\_transform(data['crop'])

data

#Defining X and Y

#Classification

X = data.drop(['pump'], axis=1)

y = data['pump']

#Regression

X1 = data.drop(['water\_liters'], axis=1)

y1 = data['water\_liters']

#Regression

X2 = data.drop(['time'], axis=1)

y2 = data['time']

#Regression

X3 = data.drop(['days'], axis=1)

y3 = data['days']

# Split the data into training and testing sets Regression

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X1, y1, test\_size=0.2, random\_state=42)

# Build the ensemble model #regression

reg\_model1 = RandomForestRegressor(n\_estimators=200,max\_depth=None,min\_samples\_split=5,min\_samples\_leaf=5,

min\_weight\_fraction\_leaf=0.01,random\_state=0)

# Fit the models

reg\_model1.fit(X\_train1, y\_train1)

# Make predictions on the test data Regression

reg\_predictions1 = reg\_model1.predict(X\_test1)

reg\_predictions1

# Calculate metrics

mae = mean\_absolute\_error(reg\_predictions1,y\_test1)

mse = mean\_squared\_error(reg\_predictions1,y\_test1)

rmse = mean\_squared\_error(reg\_predictions1,y\_test1)

r2 = r2\_score(reg\_predictions1,y\_test1)

mape = mean\_absolute\_percentage\_error(reg\_predictions1,y\_test1)

print(f"MAE: {mae}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

print(f"R-squared: {r2}")

print(f"MAPE: {mape}")

# Visualize actual vs. predicted values

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

plt.scatter(reg\_predictions1,y\_test1, color='blue', alpha=0.6)

plt.plot([min(y\_test1), max(y\_test1)], [min(y\_test1), max(y\_test1)], linestyle='--', color='red', linewidth=2)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title(f'Actual vs. Predicted Values (RMSE: {rmse:.2f})')

plt.show()

# Split the data into training and testing sets Regression

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X2, y2, test\_size=0.2, random\_state=42)

# Build the ensemble model #regression time

reg\_model2 = RandomForestRegressor(n\_estimators=100,max\_depth=25,min\_samples\_split=19,min\_samples\_leaf=11,

min\_weight\_fraction\_leaf=0.001,random\_state=24)

# Fit the models

reg\_model2.fit(X\_train2, y\_train2)

# Make predictions on the test data Regression

reg\_predictions2 = reg\_model2.predict(X\_test2)

reg\_predictions2

#Classification

X = data.drop(['pump'], axis=1)

y = data['pump']

#Regression

X1 = data.drop(['water\_liters'], axis=1)

y1 = data['water\_liters']

#Regression

X2 = data.drop(['time'], axis=1)

y2 = data['time']

#Regression

X3 = data.drop(['days'], axis=1)

y3 = data['days']

# Split the data into training and testing sets Regression

X\_train1, X\_test1, y\_train1, y\_test1 = train\_test\_split(X1, y1, test\_size=0.2, random\_state=42)

# Build the ensemble model #regression

reg\_model1 = RandomForestRegressor(n\_estimators=200,max\_depth=None,min\_samples\_split=5,min\_samples\_leaf=5,

min\_weight\_fraction\_leaf=0.01,random\_state=0)

# Fit the models

reg\_model1.fit(X\_train1, y\_train1)

# Make predictions on the test data Regression

reg\_predictions1 = reg\_model1.predict(X\_test1)

reg\_predictions1

# Calculate metrics

mae = mean\_absolute\_error(reg\_predictions1,y\_test1)

# Calculate metrics

mae = mean\_absolute\_error(reg\_predictions2,y\_test2)

mse = mean\_squared\_error(reg\_predictions2,y\_test2)

rmse = mean\_squared\_error(reg\_predictions2,y\_test2)

r2 = r2\_score(reg\_predictions2,y\_test2)

mape = mean\_absolute\_percentage\_error(reg\_predictions2,y\_test2)

print(f"MAE: {mae}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

print(f"R-squared: {r2}")

print(f"MAPE: {mape}")

# Visualize actual vs. predicted values

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

plt.scatter(reg\_predictions1,y\_test1, color='blue', alpha=0.6)

plt.plot([min(y\_test1), max(y\_test1)], [min(y\_test1), max(y\_test1)], linestyle='--', color='red', linewidth=2)

plt.xlabel('Actual Values')

plt.ylabel('Predicted Values')

plt.title(f'Actual vs. Predicted Values (RMSE: {rmse:.2f})')

plt.show()

# Split the data into training and testing sets Regression

X\_train2, X\_test2, y\_train2, y\_test2 = train\_test\_split(X2, y2, test\_size=0.2, random\_state=42)

# Build the ensemble model #regression time

reg\_model2 = RandomForestRegressor(n\_estimators=100,max\_depth=25,min\_samples\_split=19,min\_samples\_leaf=11,

min\_weight\_fraction\_leaf=0.001,random\_state=24)

# Fit the models

reg\_model2.fit(X\_train2, y\_train2)

# Make predictions on the test data Regression

reg\_predictions2 = reg\_model2.predict(X\_test2)

reg\_predictions2

#Classification

X = data.drop(['pump'], axis=1)

y = data['pump']

# Split the data into training and testing sets Regression

X\_train3, X\_test3, y\_train3, y\_test3 = train\_test\_split(X3, y3, test\_size=0.2, random\_state=42)

# Build the ensemble model #regression days

reg\_model3 = RandomForestRegressor(n\_estimators=200,max\_depth=None,min\_samples\_split=5

,min\_samples\_leaf=5,min\_weight\_fraction\_leaf=0.01,random\_state=0)

# Fit the models

reg\_model3.fit(X\_train3, y\_train3)

# Make predictions on the test data Regression

reg\_predictions3 = reg\_model2.predict(X\_test3)

reg\_predictions3

# Calculate metrics

mae = mean\_absolute\_error(reg\_predictions3,y\_test3)

mse = mean\_squared\_error(reg\_predictions3,y\_test3)

rmse = mean\_squared\_error(reg\_predictions3,y\_test3)

r2 = r2\_score(reg\_predictions3,y\_test3)

mape = mean\_absolute\_percentage\_error(reg\_predictions3,y\_test3)

print(f"MAE: {mae}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

print(f"R-squared: {r2}")

print(f"MAPE: {mape}")

#Classification

# Split the data into training and testing sets CLASSIFICATION

clf\_X\_train, clf\_X\_test, clf\_y\_train, clf\_y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Build the ensemble model #classification

clf\_model = RandomForestClassifier(random\_state=42)

# Fit the models

clf\_model.fit(clf\_X\_train, clf\_y\_train)

# Make predictions on the test data Classification

clf\_predictions = clf\_model.predict(clf\_X\_test)

clf\_predictions

Accuracy=accuracy\_score(clf\_y\_test,clf\_predictions)\*100

print("Accuracy=",Accuracy)

report=classification\_report(clf\_predictions,clf\_y\_test)

print(report)

cm=confusion\_matrix(clf\_predictions,clf\_y\_test)

plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel('Predicted')

plt.ylabel('Actual')

Accuracy=accuracy\_score(clf\_y\_test,clf\_predictions)\*100

print("Accuracy=",Accuracy)

report=classification\_report(clf\_predictions,clf\_y\_test)

print(report)

plt.title('Confusion Matrix Heat Map')

plt.show()

# Use classification output to determine regression output

combined\_output = []

for clf\_pred, reg\_pred, reg\_pred1, reg\_pred2 in zip(clf\_predictions, reg\_predictions1, reg\_predictions2, reg\_predictions3):

if clf\_pred == 1:

combined\_output.append((reg\_pred, reg\_pred1,reg\_pred2))

print('\n')

print("\*"\*100)

print("Turn on the pump")

print("Number of Liters to be Watered:", reg\_pred)

print("time :", reg\_pred1)

print("Turn on the pump after:", reg\_pred2)

print("\*"\*100)

print('\n')

# Calculate metrics

mae = mean\_absolute\_error(reg\_predictions3,y\_test3)

mse = mean\_squared\_error(reg\_predictions3,y\_test3)

rmse = mean\_squared\_error(reg\_predictions3,y\_test3)

r2 = r2\_score(reg\_predictions3,y\_test3)

mape = mean\_absolute\_percentage\_error(reg\_predictions3,y\_test3)

print(f"MAE: {mae}")

print(f"MSE: {mse}")

print(f"RMSE: {rmse}")

print(f"R-squared: {r2}")

print(f"MAPE: {mape}")

#Classification

# Split the data into training and testing sets CLASSIFICATION

clf\_X\_train, clf\_X\_test, clf\_y\_train, clf\_y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

# Build the ensemble model #classification

clf\_model = RandomForestClassifier(random\_state=42)

# Fit the models

clf\_model.fit(clf\_X\_train, clf\_y\_train)

# Make predictions on the test data Classification

clf\_predictions = clf\_model.predict(clf\_X\_test)

clf\_predictions

Accuracy=accuracy\_score(clf\_y\_test,clf\_predictions)\*100

print("Accuracy=",Accuracy)

report=classification\_report(clf\_predictions,clf\_y\_test)

print(report)

cm=confusion\_matrix(clf\_predictions,clf\_y\_test)

plt.figure(figsize=(6, 5))

sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False)

plt.xlabel('Predicted')

plt.ylabel('Actual')

Accuracy=accuracy\_score(clf\_y\_test,clf\_predictions)\*100

print("Accuracy=",Accuracy)

report=classification\_report(clf\_predictions,clf\_y\_test)

print(report)

else:

print('\n')

print("Keep the pump off")

print('\n')