Code :1

import pandas as pd

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

from sklearn.ensemble import RandomForestClassifier

from sklearn.metrics import classification\_report, accuracy\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

# Assuming you uploaded the file using the file upload feature:

try:

df = pd.read\_csv('air\_quality\_dataset-2.csv') # Replace with the actual uploaded filename if different

except FileNotFoundError:

print("File not found. Make sure it's in the same directory or uploaded correctly.")

# Handle the case where the file is not found, e.g., exit or raise an exception

# For this example, we'll simply exit

import sys

sys.exit(1)

# Drop rows with missing values

df = df.dropna()

# Drop 'rownames' column if it exists

if 'rownames' in df.columns:

df = df.drop(columns=['rownames'])

# Categorize Ozone values into AQI-like categories

def categorize\_ozone(value):

if value <= 50:

return 'Good

elif value <= 100:

return 'Moderate'

else:

return 'Unhealthy'

df['AQI\_Category'] = df['Ozone'].apply(categorize\_ozone)

# Separate features and target variable

X = df.drop(['Ozone', 'AQI\_Category'], axis=1)

y = df['AQI\_Category']

# Split 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)

# Train the Random Forest model

model = RandomForestClassifier(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Predict on the test set

y\_pred = model.predict(X\_test)

# Evaluate the model

print("Accuracy Score:", accuracy\_score(y\_test, y\_pred))

print("\nClassification Report:\n", classification\_report(y\_test, y\_pred))

# Plot feature importances

feature\_importances = pd.Series(model.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

sns.barplot(x=feature\_importances, y=feature\_importances.index)

plt.xlabel('Feature Importance Score')

plt.ylabel('Features')

plt.title("Important Features for AQI Classification")

plt.tight\_layout()

plt.show()

Code:2

import pandas as pd

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

from sklearn.ensemble import RandomForestRegressor

from sklearn.metrics import mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Load dataset

df = pd.read\_csv('air\_quality\_dataset-2.csv')

# Drop rows with missing values

df = df.dropna()

# Drop 'rownames' if it's just an index

df = df.drop('rownames', axis=1)

# Features and target

X = df.drop('Ozone', axis=1)

y = df['Ozone']

# Split into training and test sets

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

# Train Random Forest Regressor

model = RandomForestRegressor(n\_estimators=100, random\_state=42)

model.fit(X\_train, y\_train)

# Make predictions

y\_pred = model.predict(X\_test)

# Evaluation

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

print("R^2 Score:", r2\_score(y\_test, y\_pred))

# Feature Importance

feature\_imp = pd.Series(model.feature\_importances\_, index=X.columns).sort\_values(ascending=False)

sns.barplot(x=feature\_imp, y=feature\_imp.index)

plt.xlabel('Feature Importance Score')

plt.ylabel('Features')

plt.title("Feature Importance for Predicting Ozone Levels")

plt.show()

Code:3

import pandas as pd

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

from sklearn.linear\_model import LinearRegression

from sklearn.metrics import mean\_absolute\_error, mean\_squared\_error, r2\_score

import matplotlib.pyplot as plt

import seaborn as sns

# Load the dataset

df = pd.read\_csv('air\_quality\_dataset-2.csv')

# Drop rows with missing values

df = df.dropna()

# Drop 'rownames' if it's just an index

df = df.drop('rownames', axis=1)

# Define features and target

X = df.drop('Ozone', axis=1)

y = df['Ozone']

# Split the data

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

# Train Linear Regression model

model = LinearRegression()

model.fit(X\_train, y\_train)

# Predict

y\_pred = model.predict(X\_test)

# Evaluation

print("Mean Absolute Error:", mean\_absolute\_error(y\_test, y\_pred))

print("Mean Squared Error:", mean\_squared\_error(y\_test, y\_pred))

print("R^2 Score:", r2\_score(y\_test, y\_pred))

# Plotting predicted vs actual values

plt.scatter(y\_test, y\_pred, color='blue')

plt.xlabel("Actual Ozone")

plt.ylabel("Predicted Ozone")

plt.title("Actual vs Predicted Ozone Levels")

plt.plot([y.min(), y.max()], [y.min(), y.max()], 'k--', lw=2)

plt.show()

Code:4

import pandas as pd

import matplotlib.pyplot as plt

from sklearn.linear\_model import LinearRegression

# Load dataset

df = pd.read\_csv('air\_quality\_dataset-2.csv')

# Drop rows with missing values

df = df.dropna()

# Drop 'rownames' column if present

df = df.drop('rownames', axis=1)

# Use Temp as the feature (X) and Ozone as target (y)

X = df[['Temp']]

y = df['Ozone']

# Train the model

model = LinearRegression()

model.fit(X, y)

# Predict values for the regression line

y\_pred = model.predict(X)

# Plotting

plt.scatter(X, y, color='blue', label='Actual data')

plt.plot(X, y\_pred, color='red', linewidth=2, label='Regression line')

plt.xlabel('Temperature (Temp)')

plt.ylabel('Ozone')

plt.title('Temperature vs Ozone Levels')

plt.legend()

plt.show()

Code:5

import pandas as pd

import matplotlib.pyplot as plt

# Load dataset

df = pd.read\_csv('air\_quality\_dataset-2.csv')

# Drop rows with missing Ozone values

df = df.dropna(subset=['Ozone'])

# Categorize Ozone levels

def categorize\_ozone(value):

if value <= 50:

return 'Good'

elif value <= 100:

return 'Moderate'

else:

return 'Unhealthy'

df['Ozone\_Category'] = df['Ozone'].apply(categorize\_ozone)

# Count category occurrences

category\_counts = df['Ozone\_Category'].value\_counts()

# Pie chart

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

plt.pie(category\_counts, labels=category\_counts.index, autopct='%1.1f%%', startangle=140, colors=['green', 'gold', 'red'])

plt.title('Ozone Level Categories')

plt.axis('equal') # Equal aspect ratio makes the pie circular

plt.show()

Code:6

import seaborn as sns

import matplotlib.pyplot as plt

# Univariate Analysis - Distribution Plot

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

sns.histplot(df['Ozone'], bins=30, kde=True, color='blue')

plt.title('Distribution of Ozone Levels')

plt.xlabel('Ozone')

plt.ylabel('Frequency')

plt.show()

Code:7

import seaborn as sns

import matplotlib.pyplot as plt

# Bivariate Analysis - Boxplot (Ozone across Months)

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

sns.boxplot(x='Month', y='Ozone', data=df)

plt.title('Ozone Levels Across Different Months')

plt.xlabel('Month')

plt.ylabel('Ozone')

plt.show()

Code:8

import seaborn as sns

import matplotlib.pyplot as plt

# Multivariate Analysis - Correlation Matrix

plt.figure(figsize=(10,8))

corr\_matrix = df.corr()

sns.heatmap(corr\_matrix, annot=True, cmap='coolwarm', linewidths=0.5)

plt.title('Correlation Matrix of Air Quality Features')

plt.show()

# Multivariate Analysis - Pairplot

sns.pairplot(df[['Ozone', 'Solar.R', 'Wind', 'Temp']], diag\_kind='kde')

plt.suptitle('Pairplot of Selected Features', y=1.02)

plt.show()

Code:9

import seaborn as sns

import pandas as pd

import matplotlib.pyplot as plt

# Load a dataset (you can replace this with your own CSV file or dataset)

df = sns.load\_dataset('iris')

# Create a pair plot

sns.pairplot(df)

plt.show()