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.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:3

```
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:4

```
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:5

```
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:6

```
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:7

```
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()
```

code:8

```
# 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

```
from sklearn.metrics import
confusion_matrix,
ConfusionMatrixDisplay
import matplotlib.pyplot as
plt
# Replace with your
predictions and true labels
y_true = [...]  # actual labels
y_pred = [...]  # predicted
labels
cm
=confusion_matrix(y_true,
y_pred)
disp =
ConfusionMatrixDisplay(co
nfusion_matrix=cm)
disp.plot(cmap='Blues')
plt.title("Confusion Matrix")
```

```
plt.show()
```

Code:10

```
from sklearn.metrics import
roc_curve, auc
import matplotlib.pyplot as
plt
# If binary classification
fpr, tpr, thresholds =
roc_curve(y_true,
model.predict_proba(X_test
)[:, 1])
roc_auc = auc(fpr, tpr)
plt.figure()
plt.plot(fpr, tpr, label=f'ROC
curve (area =
{roc_auc:.2f})')
plt.plot([0, 1], [0, 1], 'k--')  #
random line
plt.xlabel('False Positive
```

```
Rate')
plt.ylabel('True Positive
Rate')
plt.title('Receiver Operating
Characteristic (ROC)')
plt.legend(loc="lower
right")
plt.show()
```