# Google Colab Python Code

# Upload the dataset  
from google.colab import files  
uploaded = files.upload()  
  
# Load the dataset  
import pandas as pd  
df = pd.read\_csv('/content/PRSA\_data\_2010.1.1-2014.12.31.csv')  
df.head()  
  
# Data Exploration  
print("Shape:", df.shape)  
df.info()  
df.describe()  
  
# Check for missing values and duplicates  
print("Missing values:\n", df.isnull().sum())  
print("\nDuplicate rows:", df.duplicated().sum())  
df = df.drop\_duplicates()  
  
# Visualize features  
import matplotlib.pyplot as plt  
import seaborn as sns  
features = ['TEMP', 'PRES', 'DEWP', 'pm2.5']  
for feature in features:  
 sns.histplot(df[feature].dropna(), kde=True)  
 plt.title(f'Distribution of {feature}')  
 plt.show()  
  
# Identify target and features  
target = 'pm2.5'  
features = df.drop(columns=[target]).columns.tolist()  
print("Features:", features)  
  
# Convert categorical column to numeric  
if 'cbwd' in df.columns:  
 df['cbwd'] = df['cbwd'].astype('category').cat.codes  
  
# One-Hot Encoding  
df = pd.get\_dummies(df, columns=df.select\_dtypes(include='object').columns)  
  
# Feature Scaling  
from sklearn.preprocessing import StandardScaler  
scaler = StandardScaler()  
scaled\_features = scaler.fit\_transform(df.drop(columns=[target]))  
X = pd.DataFrame(scaled\_features, columns=df.drop(columns=[target]).columns)  
y = df[target]  
  
# Train-Test Split  
from sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
  
# Drop rows where target is NaN and reprocess  
df = df.dropna(subset=['pm2.5'])  
X = df.drop(columns=['pm2.5'])  
y = df['pm2.5']  
X['cbwd'] = X['cbwd'].astype('category').cat.codes  
X = pd.get\_dummies(X, drop\_first=True)  
X\_scaled = scaler.fit\_transform(X)  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X\_scaled, y, test\_size=0.2, random\_state=42)  
  
# Model Building  
from sklearn.ensemble import RandomForestRegressor  
model = RandomForestRegressor()  
model.fit(X\_train, y\_train)  
  
# Evaluation  
from sklearn.metrics import mean\_squared\_error, r2\_score  
y\_pred = model.predict(X\_test)  
print("MSE:", mean\_squared\_error(y\_test, y\_pred))  
print("R^2 Score:", r2\_score(y\_test, y\_pred))  
  
# Make Predictions from New Input  
import numpy as np  
sample = X\_test[0].reshape(1, -1) if isinstance(X\_test, np.ndarray) else X\_test.iloc[0].values.reshape(1, -1)  
prediction = model.predict(sample)  
print("Predicted PM2.5:", prediction[0])  
  
# Simulate user input  
user\_input = {'TEMP': 5, 'PRES': 1020, 'DEWP': -3, 'cbwd': 'NW', 'Iws': 10, 'Is': 0, 'Ir': 0}  
user\_df = pd.DataFrame([user\_input])  
if 'cbwd' in user\_df.columns:  
 user\_df['cbwd'] = user\_df['cbwd'].astype('category').cat.codes  
print("Scaler was fitted on:", scaler.feature\_names\_in\_)  
print("User input columns: ", user\_df.columns.tolist())  
  
# Final prediction on small dataset  
data = {  
 'TEMP': [22, 25, 20, 18, 23, 19],  
 'PRES': [1015, 1020, 1012, 1018, 1016, 1022],  
 'DEWP': [10, 12, 8, 7, 9, 11],  
 'cbwd': ['NW', 'NE', 'SE', 'SW', 'NW', 'SE'],  
 'Iws': [15, 18, 13, 12, 14, 16],  
 'Is': [5, 3, 4, 6, 5, 4],  
 'Ir': [0, 1, 0, 0, 1, 0],  
 'PM2.5': [25, 30, 20, 15, 28, 22]  
}  
df = pd.DataFrame(data)  
df['cbwd'] = df['cbwd'].astype('category').cat.codes  
X = df.drop('PM2.5', axis=1)  
y = df['PM2.5']  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)  
scaler = StandardScaler()  
X\_train\_scaled = scaler.fit\_transform(X\_train)  
X\_test\_scaled = scaler.transform(X\_test)  
model = RandomForestRegressor(random\_state=42)  
model.fit(X\_train\_scaled, y\_train)  
  
user\_input = {'TEMP': 5, 'PRES': 1020, 'DEWP': -3, 'cbwd': 'NW', 'Iws': 10, 'Is': 0, 'Ir': 0}  
user\_df = pd.DataFrame([user\_input])  
user\_df['cbwd'] = user\_df['cbwd'].astype('category').cat.codes  
user\_df\_scaled = scaler.transform(user\_df)  
final\_prediction = model.predict(user\_df\_scaled)  
print("Final Predicted PM2.5:", final\_prediction[0])  
  
# Gradio App Setup  
!pip install gradio  
import gradio as gr