Upload the Dataset

```
from google.colab import files
uploaded = files.upload()
```

Choose Files PRSA_data…4.12.31.csv

- **PRSA_data_2010.1.1-2014.12.31.csv**(text/csv) - 2010494 bytes, last modified: 5/11/2025 - 100% done

```
Saving PRSA_data_2010.1.1-2014.12.31.csv to PRSA_data_2010.1.1-2014.12.31.csv
```

Load the Dataset

```
import pandas as pd

# Replace the file name if different
df = pd.read_csv('/content/PRSA_data_2010.1.1-2014.12.31.csv')
df.head()
```

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	cbwd	Iws	Is	Ir
0	1	2010	1	1	0	NaN	-21	-11.0	1021.0	NW	1.79	0	0
1	2	2010	1	1	1	NaN	-21	-12.0	1020.0	NW	4.92	0	0
2	3	2010	1	1	2	NaN	-21	-11.0	1019.0	NW	6.71	0	0
3	4	2010	1	1	3	NaN	-21	-14.0	1019.0	NW	9.84	0	0
4	5	2010	1	1	4	NaN	-20	-12.0	1018.0	NW	12.97	0	0

Next steps: Generate code with df | View recommended plots | New interactive sheet

Data Exploration

```
# Shape and info
print("Shape:", df.shape)
print("\nInfo:\n")
df.info()

# Describe
df.describe()
```

```
Shape: (43824, 13)

Info:

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 43824 entries, 0 to 43823
Data columns (total 13 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   No      43824 non-null  int64
 1   year    43824 non-null  int64
 2   month   43824 non-null  int64
 3   day     43824 non-null  int64
 4   hour    43824 non-null  int64
 5   pm2.5   41757 non-null  float64
 6   DEWP    43824 non-null  int64
 7   TEMP    43824 non-null  float64
 8   PRES    43824 non-null  float64
 9   cbwd    43824 non-null  object
 10  Iws     43824 non-null  float64
 11  Is      43824 non-null  int64
 12  Ir      43824 non-null  int64
dtypes: float64(4), int64(8), object(1)
memory usage: 4.3+ MB
```

	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	
count	43824.000000	43824.000000	43824.000000	43824.000000	43824.000000	41757.000000	43824.000000	43824.000000	43824.000000	43
mean	21912.500000	2012.000000	6.523549	15.727820	11.500000	98.613215	1.817246	12.448521	1016.447654	
std	12651.043435	1.413842	3.448572	8.799425	6.922266	92.050387	14.433440	12.198613	10.268698	
min	1.000000	2010.000000	1.000000	1.000000	0.000000	0.000000	-40.000000	-19.000000	991.000000	
25%	10956.750000	2011.000000	4.000000	8.000000	5.750000	29.000000	-10.000000	2.000000	1008.000000	
50%	21912.500000	2012.000000	7.000000	16.000000	11.500000	72.000000	2.000000	14.000000	1016.000000	
75%	32868.250000	2013.000000	10.000000	23.000000	17.250000	137.000000	15.000000	23.000000	1025.000000	
max	43824.000000	2014.000000	12.000000	31.000000	23.000000	994.000000	28.000000	42.000000	1046.000000	5

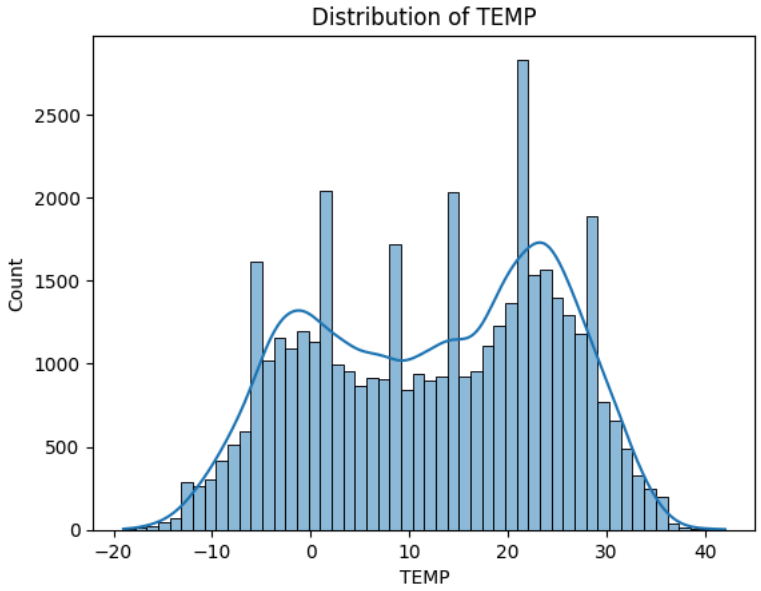
Check for Missing Values and Duplicates

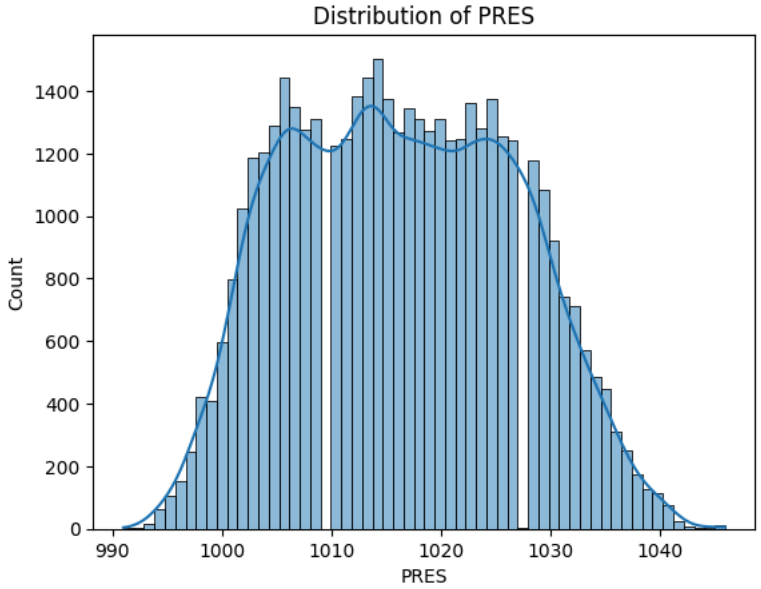
```
# Missing values
print("Missing values:\n", df.isnull().sum())

# Duplicates
print("\nDuplicate rows:", df.duplicated().sum())

# Drop duplicates if necessary
df = df.drop_duplicates()
```

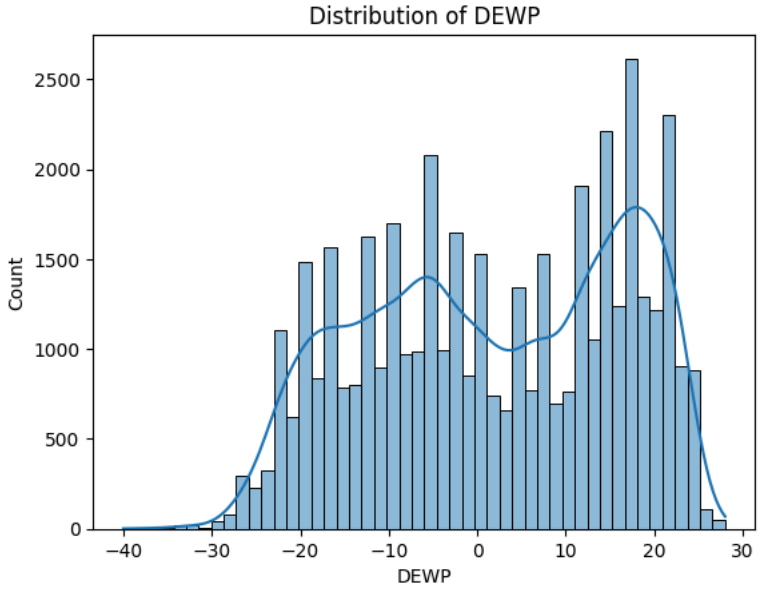
```
Missing values:
 No          0
year         0
month        0
day          0
hour         0
pm2.5     2067
DEWP         0
TEMP         0
PRES         0
cbwd         0
Iws          0
Is           0
Ir           0
dtype: int64

Duplicate rows: 0
```

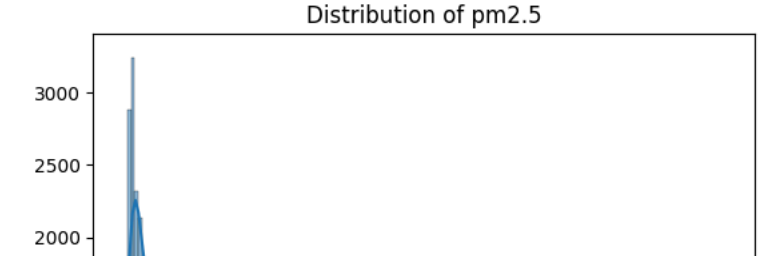
Visualize a Few Features

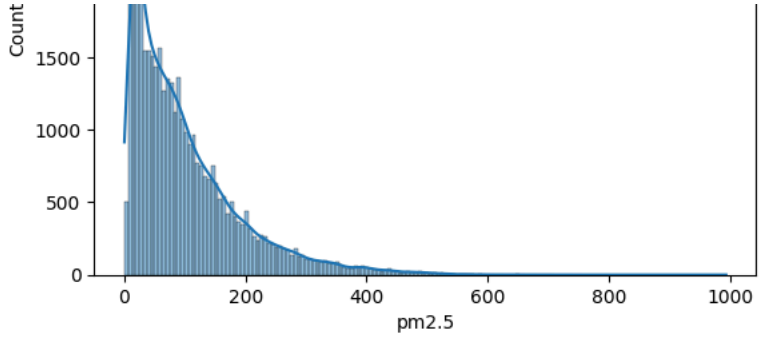
```
import matplotlib.pyplot as plt
import seaborn as sns

# Distribution of a few numeric features
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()
```


Distribution of TEMP
Count
TEMP
2500
2000
1500
1000
500
0
−20
−10
0
10
20
30
40


Distribution of PRES
Count
PRES
1400
1200
1000
800
600
400
200
0
990
1000
1010
1020
1030
1040


Distribution of DEWP
Count
DEWP
2500
2000
1500
1000
500
0
−40
−30
−20
−10
0
10
20
30


Distribution of pm2.5
3000
2500
2000



Identify Target and Features

```
# Assuming 'pm2.5' is the target variable
target = 'pm2.5'
features = df.drop(columns=[target]).columns.tolist()
print("Features:", features)
```

```
Features: ['No', 'year', 'month', 'day', 'hour', 'DEWP', 'TEMP', 'PRES', 'cbwd', 'Iws', 'Is', 'Ir']
```

Convert Categorical Columns to Numerical

```
# Example: converting 'cbwd' (wind direction) to numerical codes
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)
```

Model Building

```
# Drop rows where target (pm2.5) is NaN
df = df.dropna(subset=['pm2.5'])

# Now reassign X and y
X = df.drop(columns=['pm2.5'])
y = df['pm2.5']

# Also, re-encode and scale as before
X['cbwd'] = X['cbwd'].astype('category').cat.codes
X = pd.get_dummies(X, drop_first=True)

from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y, test_size=0.2, random_state=42)



from sklearn.ensemble import RandomForestRegressor

model = RandomForestRegressor()
model.fit(X_train, y_train)
```

```
▾ RandomForestRegressor
RandomForestRegressor()
```

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))
```

```
MSE: 1207.8875614703065
R^2 Score: 0.8629194230462242
```

Make Predictions from New Input

```
import numpy as np

# If X_test is a DataFrame
if hasattr(X_test, 'iloc'):
    sample = X_test.iloc[0].values.reshape(1, -1)
else:  # If it's a NumPy array
    sample = X_test[0].reshape(1, -1)

prediction = model.predict(sample)
print("Predicted PM2.5:", prediction[0])
```

```
Predicted PM2.5: 146.54
```

Convert to DataFrame and Encode

```
# 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])

# Convert categorical columns
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())
```

```
Scaler was fitted on: ['No' 'year' 'month' 'day' 'hour' 'DEWP' 'TEMP' 'PRES' 'cbwd' 'Iws' 'Is'
 'Ir']
User input columns:  ['TEMP', 'PRES', 'DEWP', 'cbwd', 'Iws', 'Is', 'Ir']
```

Predict the Final Grade (PM2.5)

```
import pandas as pd
import numpy as np
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.model_selection import train_test_split

# Simulated training data (replace with your actual data)
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]  # Target variable (Final Grade - PM2.5)
}

df = pd.DataFrame(data)

# Preprocessing
# Handle categorical variable 'cbwd' (convert to numerical codes)
df['cbwd'] = df['cbwd'].astype('category').cat.codes

# Feature columns and target variable
X = df.drop('PM2.5', axis=1)
y = df['PM2.5']

# Split the data into train and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Feature scaling
scaler = StandardScaler()
X_train_scaled = scaler.fit_transform(X_train)
X_test_scaled = scaler.transform(X_test)

# Model training (Random Forest Regressor)
model = RandomForestRegressor(random_state=42)
model.fit(X_train_scaled, y_train)

# Prediction for new data (simulated user input)
user_input = {'TEMP': 5, 'PRES': 1020, 'DEWP': -3, 'cbwd': 'NW', 'Iws': 10, 'Is': 0, 'Ir': 0}
user_df = pd.DataFrame([user_input])

# Convert categorical feature (same as training time)
user_df['cbwd'] = user_df['cbwd'].astype('category').cat.codes

# Scale the user input
user_df_scaled = scaler.transform(user_df)

# Make prediction
final_prediction = model.predict(user_df_scaled)
print("Final Predicted PM2.5:", final_prediction[0])
```

```
Final Predicted PM2.5: 19.66
```

Deployment - Building an Interactive App

```
!pip install gradio
import gradio as gr
```

```
Requirement already satisfied: certifi in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (2025.4.26)
Requirement already satisfied: httpcore==1.* in /usr/local/lib/python3.11/dist-packages (from httpx>=0.24.1->gradio) (1.0.9)
Requirement already satisfied: h11>=0.16 in /usr/local/lib/python3.11/dist-packages (from httpcore==1.*->httpx>=0.24.1->gradio) (0
Requirement already satisfied: filelock in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (3.18.0)
Requirement already satisfied: requests in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (2.32.3)
Requirement already satisfied: tqdm>=4.42.1 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->gradio) (4.6
Requirement already satisfied: hf-xet<2.0.0,>=1.1.0 in /usr/local/lib/python3.11/dist-packages (from huggingface-hub>=0.28.1->grad
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas<3.0,>=1.0->gradio) (2025.2)
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio
Requirement already satisfied: pydantic-core==2.33.2 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->gradio)
Requirement already satisfied: typing-inspection>=0.4.0 in /usr/local/lib/python3.11/dist-packages (from pydantic<2.12,>=2.0->grad
Requirement already satisfied: click>=8.0.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (8.1.8)
Requirement already satisfied: shellingham>=1.3.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (1.5
Requirement already satisfied: rich>=10.11.0 in /usr/local/lib/python3.11/dist-packages (from typer<1.0,>=0.12->gradio) (13.9.4)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas<3.0,>=1.0-
Requirement already satisfied: markdown-it-py>=2.2.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>=0
Requirement already satisfied: pygments<3.0.0,>=2.13.0 in /usr/local/lib/python3.11/dist-packages (from rich>=10.11.0->typer<1.0,>
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.11/dist-packages (from requests->huggingface-hub>=0.28
Requirement already satisfied: mdurl~=0.1 in /usr/local/lib/python3.11/dist-packages (from markdown-it-py>=2.2.0->rich>=10.11.0->t
Downloading gradio-5.29.1-py3-none-any.whl (54.1 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 54.1/54.1 MB 12.3 MB/s eta 0:00:00
Downloading gradio_client-1.10.1-py3-none-any.whl (323 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 323.1/323.1 kB 19.6 MB/s eta 0:00:00
Downloading aiofiles-24.1.0-py3-none-any.whl (15 kB)
Downloading fastapi-0.115.12-py3-none-any.whl (95 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 95.2/95.2 kB 6.7 MB/s eta 0:00:00
Downloading groovy-0.1.2-py3-none-any.whl (14 kB)
Downloading python_multipart-0.0.20-py3-none-any.whl (24 kB)
Downloading ruff-0.11.9-py3-none-manylinux_2_17_x86_64.manylinux2014_x86_64.whl (11.5 MB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 11.5/11.5 MB 101.8 MB/s eta 0:00:00
Downloading safehttpx-0.1.6-py3-none-any.whl (8.7 kB)
Downloading semantic_version-2.10.0-py2.py3-none-any.whl (15 kB)
Downloading starlette-0.46.2-py3-none-any.whl (72 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 72.0/72.0 kB 4.9 MB/s eta 0:00:00
Downloading tomlkit-0.13.2-py3-none-any.whl (37 kB)
Downloading uvicorn-0.34.2-py3-none-any.whl (62 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 62.5/62.5 kB 4.0 MB/s eta 0:00:00
Downloading ffmpy-0.5.0-py3-none-any.whl (6.0 kB)
Downloading pydub-0.25.1-py2.py3-none-any.whl (32 kB)
Installing collected packages: pydub, uvicorn, tomlkit, semantic-version, ruff, python-multipart, groovy, ffmpy, aiofiles, starlet
Successfully installed aiofiles-24.1.0 fastapi-0.115.12 ffmpy-0.5.0 gradio-5.29.1 gradio-client-1.10.1 groovy-0.1.2 pydub-0.25.1 p
```