**ML PROJECT REPORT**

**Name:** Anushaa, Joan, Harshita

**Class:** 4BScDS

**Reg\_no.:** 2341490, 2341441, 2341439

**A Case Study on the PRSA Dataset**

**Introduction:**

* Air pollution, particularly PM2.5 (fine particulate matter), poses a severe environmental and health risk in urban areas. Beijing, one of the most polluted cities in the world, has experienced rising PM2.5 levels, leading to respiratory diseases and reduced air quality. Accurately predicting PM2.5 concentrations can help in air quality monitoring, early warning systems, and policy-making to mitigate pollution effects.
* This study aims to develop a machine learning model that predicts PM2.5 air pollution levels in Beijing using meteorological and temporal data. By leveraging factors such as temperature, humidity, wind speed, and time-based variables (hour, month, season), we seek to create a reliable forecasting model that can assist in air pollution management.
* To achieve this, we apply data preprocessing and exploratory data analysis (EDA) techniques to clean and transform the dataset before implementing predictive modeling. The key steps include:

1. Handling missing values using imputation techniques.

1. Performing data wrangling and feature engineering to enhance predictive power.
2. Encoding categorical variables and normalizing numerical features.
3. Visualizing air pollution trends and their relationship with meteorological factors.

* By the end of this study, we aim to evaluate the feasibility of using weather and time-based features to predict PM2.5 concentrations and assess the performance of different machine learning models in forecasting air pollution levels.

**Dataset Description:**

1. No: Row number, serving as a unique identifier for each observation.
2. Year: The year when the observation was recorded (2010–2014).
3. Month: The month when the observation was recorded (1–12).
4. Day: The day of the month when the observation was recorded (1–31).
5. Hour: The hour of the day when the observation was recorded (0–23), using a 24-hour clock format.
6. PM2.5: Concentration of PM2.5 particles in micrograms per cubic meter (µg/m³). This represents air quality and includes some missing values.
7. DEWP: Dew point temperature in degrees Celsius (°C). The temperature at which air becomes saturated with moisture.
8. TEMP: Actual air temperature in degrees Celsius (°C) at the time of observation.
9. PRES: Atmospheric pressure in hectopascals (hPa) at the time of observation.
10. CBWD: Combined wind direction, represented as a categorical variable (e.g., NW, NE, SW, SE) indicating the general direction of the wind.
11. IWS: Cumulated wind speed in meters per second (m/s), showing the strength of the wind over a period.
12. IS: Snowfall indicator, where 0 means no snow and positive values represent the amount of snowfall.
13. IR: Rainfall indicator, where 0 means no rain and positive values represent the amount of rainfall.

This dataset captures hourly environmental conditions from 2010 to 2014, providing information on air quality, temperature, pressure, wind, and precipitation.

**Problem statement:**

"Can we accurately predict PM2.5 air pollution levels in Beijing using meteorological and temporal data? This study aims to develop a machine learning model that leverages weather conditions (temperature, humidity, wind speed, etc.) and time-based factors (hour, month, season) to forecast PM2.5 concentrations, helping in air quality monitoring and early warning systems."

**Need for addressing the problem statement:**

* Air pollution, particularly PM2.5 (fine particulate matter), has become a critical environmental and public health issue, especially in densely populated urban areas like Beijing. Long-term exposure to high PM2.5 levels has been linked to respiratory diseases, cardiovascular conditions, and reduced life expectancy.
* Given the significant health and environmental implications, developing an accurate predictive model for PM2.5 concentrations is crucial for several reasons:

1. Public Health Protection

* High PM2.5 levels contribute to asthma, lung infections, and other respiratory illnesses.
* Early forecasting of pollution spikes can enable timely health advisories and reduce exposure.
* Air Quality Monitoring & Early Warning Systems
* Real-time air pollution prediction helps authorities issue alerts and recommendations.
* A reliable forecast allows citizens to take preventive measures, such as wearing masks or staying indoors.

1. Policy Making & Pollution Control Measures

* Accurate air quality predictions can guide government policies on emissions control.
* Helps in designing traffic restrictions and industrial regulations based on pollution trends.

1. Data-Driven Decision Making

* Meteorological and time-based data can reveal patterns in air pollution, helping researchers and policymakers understand its causes.
* Identifying seasonal and hourly variations in PM2.5 levels can lead to targeted interventions.

1. Sustainable Urban Planning

* Predictive models can assist in designing smarter urban layouts to reduce pollution exposure.
* Helps in optimizing public transportation systems to lower emissions.

By leveraging machine learning and data analytics, this study aims to create a PM2.5 forecasting model that contributes to effective pollution management. A reliable prediction system can aid in reducing health risks, improving air quality management, and fostering sustainable urban development in Beijing and other polluted cities worldwide.

**Objectives:**

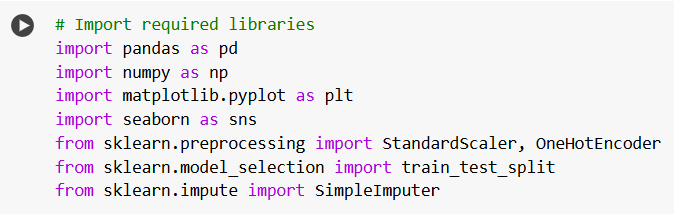
The primary goal of this study is to develop a machine learning model to predict PM2.5 air pollution levels in Beijing using meteorological and temporal data. To achieve this, the following objectives are defined:

1. To collect and preprocess the PRSA dataset by handling missing values, performing feature engineering, and transforming the data for analysis.
2. To explore the relationship between PM2.5 levels and meteorological variables (such as temperature, humidity, wind speed, and pressure) through statistical analysis and data visualization.
3. To analyze the impact of temporal factors (hour, day, month, and season) on PM2.5 concentrations to identify trends and seasonal variations.
4. To apply machine learning techniques for predictive modeling, including data splitting, model training, and hyperparameter tuning.
5. To evaluate and compare the performance of different machine learning models using appropriate metrics such as RMSE (Root Mean Square Error) and R² score.
6. To identify the most significant features influencing PM2.5 levels, aiding in better understanding of air pollution dynamics.
7. To develop a predictive framework that can be used for real-time air quality forecasting and early warning systems.
8. To provide insights and recommendations based on the findings to support policymakers, environmental agencies, and researchers in air quality management.

**Interpretations:**

1. **Preprocessing data**

Data preprocessing and visualisation (cleaning , understanding the data and EDA)- aggregation, wrangling, one hot encoding, scaling, feature engineering, etc, everything relevant to the data is done.

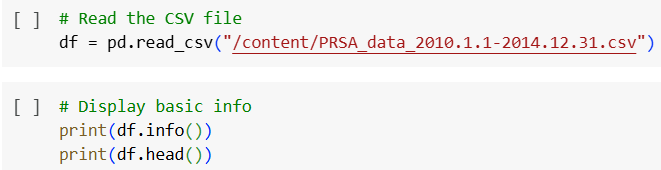


This code imports the necessary libraries required for data preprocessing, visualization, and machine learning tasks.

1. **pandas as pd** – Used for data manipulation and analysis, particularly for working with structured data (DataFrames).
2. **numpy as np** – Provides support for numerical computations, including arrays and mathematical operations.
3. **matplotlib.pyplot as plt** – A visualization library used to create static, animated, and interactive plots.
4. **seaborn as sns** – Built on top of Matplotlib, it simplifies complex statistical visualizations.

**Scikit-learn Imports (For Machine Learning & Preprocessing)**

1. **StandardScaler & OneHotEncoder (from sklearn.preprocessing)**
   * StandardScaler: Normalizes numerical features by scaling them to have **zero mean and unit variance** (useful for ML models).
   * OneHotEncoder: Converts categorical variables into numerical format using **one-hot encoding** (binary representation).
2. **train\_test\_split (from sklearn.model\_selection)**
   * Splits the dataset into **training and testing sets**, ensuring proper model evaluation.
3. **SimpleImputer (from sklearn.impute)**
   * Handles **missing values** by replacing them with a specified strategy (e.g., mean, median, or most frequent value).

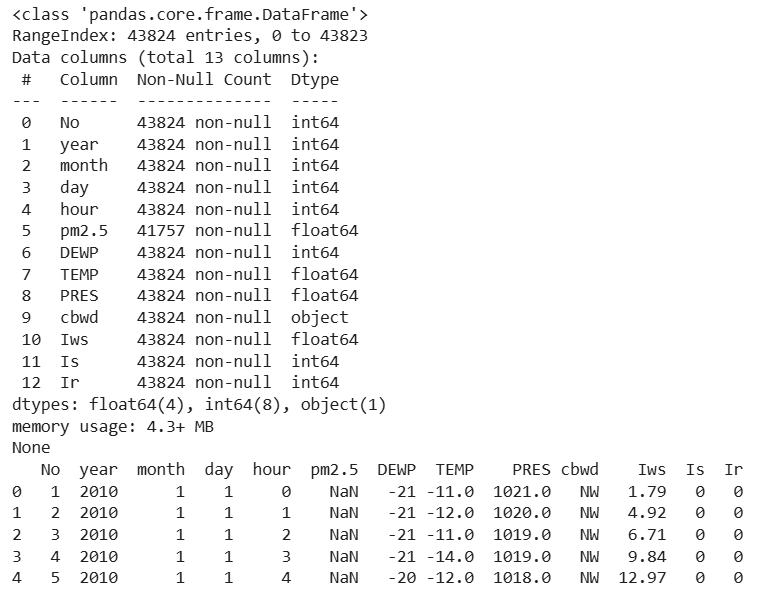


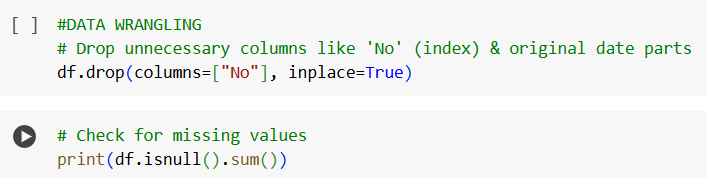
**Reading CSV File**

* Loads the PRSA air pollution dataset (2010–2014) into a Pandas DataFrame using pd.read\_csv().

**Basic Info & Preview**

* df.info(): Displays column names, data types, and missing values.
* df.head(): Shows the first 5 rows to check data structure.



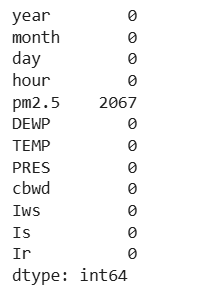


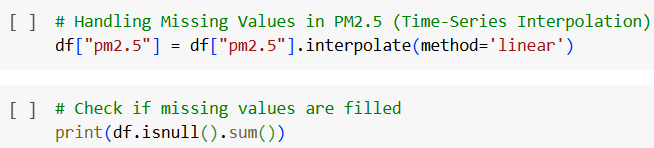
**Dropping Unnecessary Columns**

* df.drop(columns=["No"], inplace=True): Removes the **"No"** column (likely an index or redundant feature) to clean the dataset.

**Checking for Missing Values**

* df.isnull().sum(): Counts missing values in each column to identify data gaps.



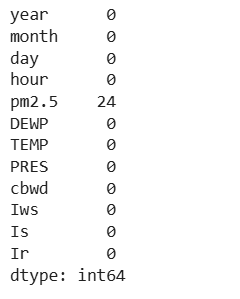


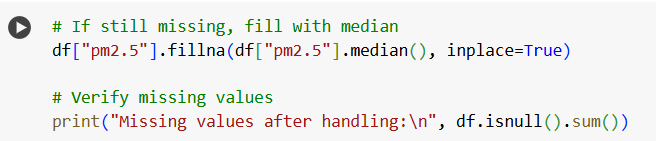
**Handling Missing Values in PM2.5**

* df["pm2.5"] = df["pm2.5"].interpolate(method='linear')
* Uses **linear interpolation** to fill missing values in the **PM2.5** column based on nearby values.

**Checking if Missing Values are Filled**

* df.isnull().sum(): Verifies that missing values have been handled.



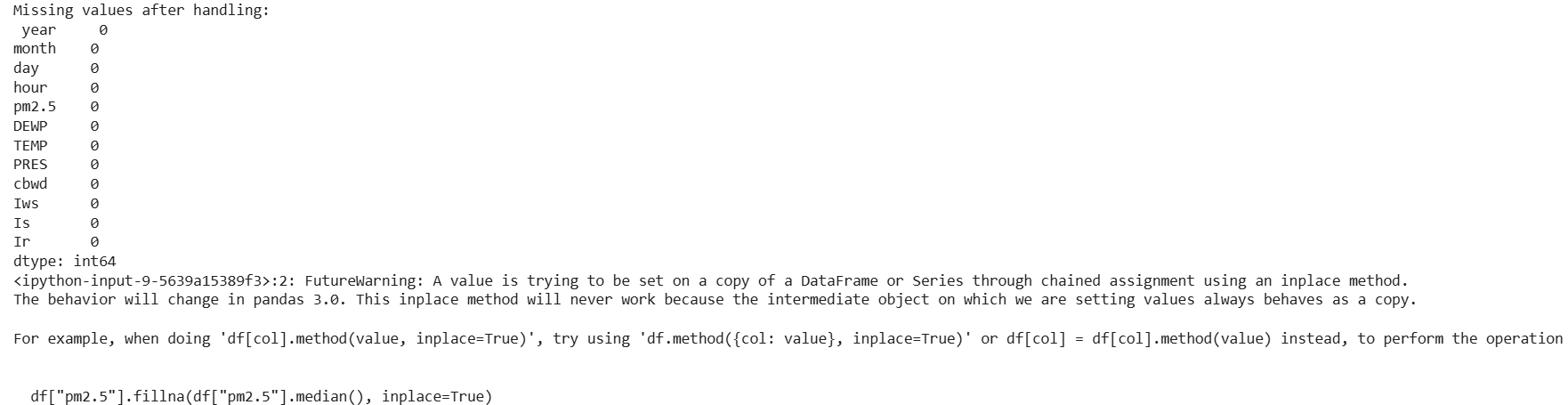


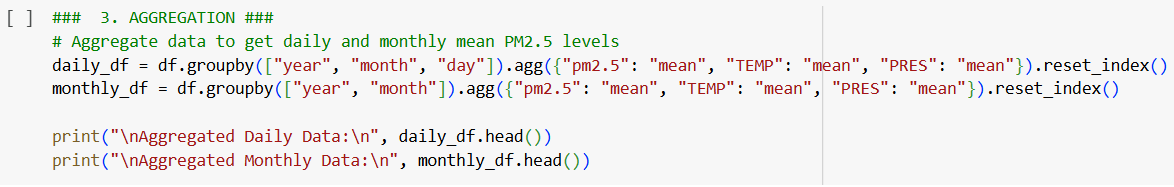
**Filling Remaining Missing Values**

* df["pm2.5"].fillna(df["pm2.5"].median(), inplace=True)
* If any missing values remain after interpolation, they are filled with the **median** of the PM2.5 column.

**Verifying Missing Values**

* Prints the count of missing values after handling to ensure completeness.





**Daily Aggregation**

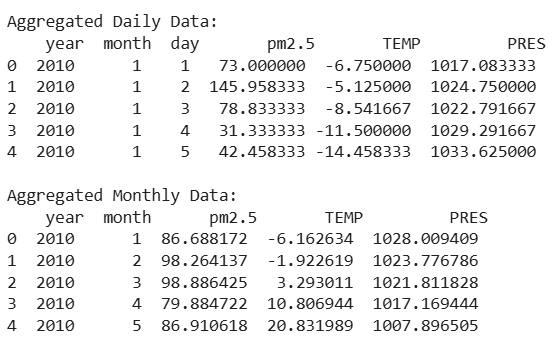
* Groups data by **year, month, and day**.
* Computes the **mean** for PM2.5, temperature (TEMP), and pressure (PRES).

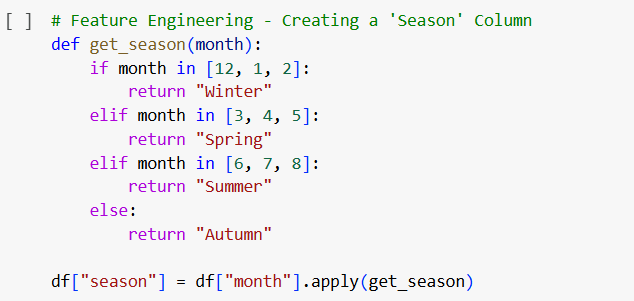
**Monthly Aggregation**

* Groups data by **year and month**.
* Computes the **mean** for the same parameters.

**Output Display**

* Prints the first few rows of aggregated daily and monthly data.





**Function Definition**

* get\_season(month): Maps each month to a season.

**Season Classification**

**Winter**: December, January, February

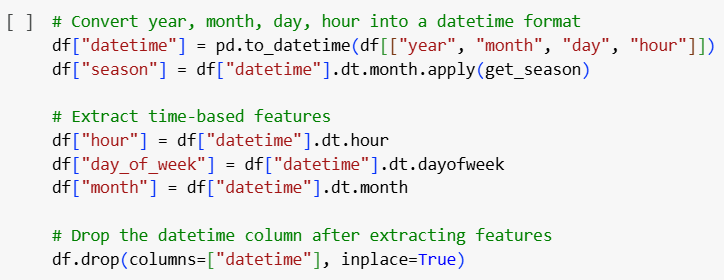
**Spring**: March, April, May

**Summer**: June, July, August

**Autumn**: September, October, November

**Application**

* Uses .apply(get\_season) to create a new "season" column based on the "month" column.



**Datetime Conversion**

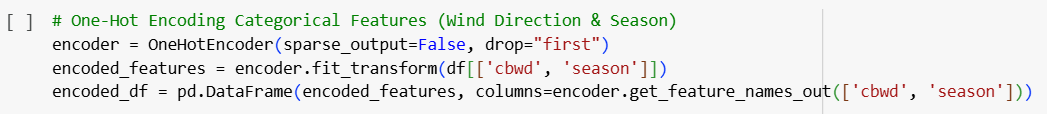
* Merges "year", "month", "day", and "hour" into a "datetime" column.
* Extracts "season" using the get\_season() function.

**Feature Extraction**

* "hour": Extracted from "datetime".
* "day\_of\_week": Extracted to indicate weekdays (0 = Monday, 6 = Sunday).
* "month": Retained from "datetime".

**Cleanup**

* Drops "datetime" after extracting relevant features.

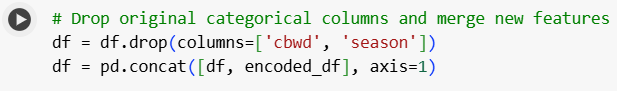


**One-Hot Encoding**

* Encodes categorical variables ('cbwd' for wind direction and 'season').
* Uses OneHotEncoder with drop="first", meaning one category from each is dropped to avoid multicollinearity.
* sparse\_output=False ensures a dense NumPy array output.

**Transformation**

* fit\_transform() applies one-hot encoding to the selected columns.
* Converts the output into a DataFrame with meaningful column names.

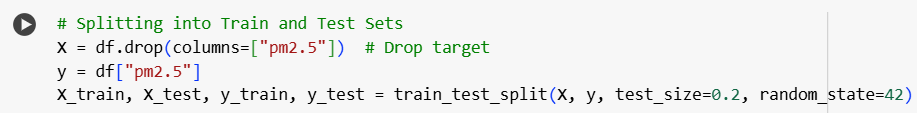


**Dropping Categorical Columns**

* Removes original categorical columns 'cbwd' and 'season' after encoding.

**Concatenating Encoded Features**

* Joins the encoded DataFrame (encoded\_df) with the main DataFrame (df) along columns (axis=1).

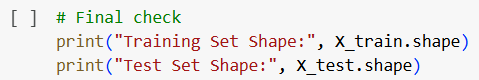


**Feature-Target Split**

* X = df.drop(columns=["pm2.5"]): Removes the target variable ("pm2.5") from the dataset, keeping only features.
* y = df["pm2.5"]: Extracts the target variable ("pm2.5") for prediction.

**Train-Test Split**

* train\_test\_split(X, y, test\_size=0.2, random\_state=42): Splits data into training (80%) and testing (20%) sets.
* random\_state=42: Ensures reproducibility of results.



**Purpose**

* This code checks the shape of the training and test sets after the train\_test\_split() function.
* It prints the number of rows and columns in both X\_train and X\_test.

**Expected Output**

* Training set should contain **80%** of the data.
* Test set should contain **20%** of the data.

**Why It's Important**

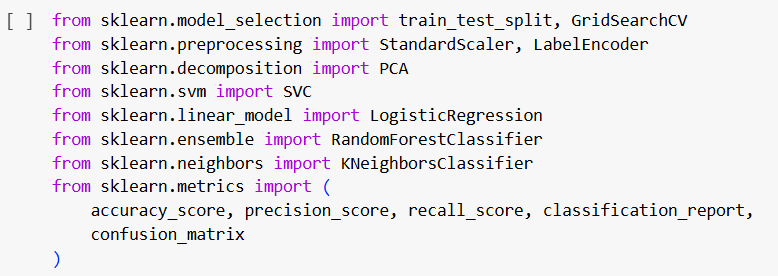
* Ensures the split was performed correctly.
* Helps verify the dataset is correctly partitioned before model training.



1. **Visualizing the data and performing PCA, KNN etc.**

To build classification models (SVM, Logistic Regression, Random Forest, KNN) to predict PM2.5 levels, evaluated them using accuracy, precision, recall, and applied PCA to reduce dimensions.

After retraining models on PCA-transformed data, analyzed the impact on performance, tuned KNN for the best k-value, and visualized results with confusion matrices, classification reports, and ROC curves. Debugging ensured correct feature alignment between models and test datasets.

****

This code imports essential **machine learning** libraries from sklearn for data preprocessing, model selection, classification, and evaluation.

**Model Selection & Hyperparameter Tuning**

train\_test\_split: Splits data into training and test sets.

GridSearchCV: Performs hyperparameter tuning using cross-validation.

**Data Preprocessing**

StandardScaler: Standardizes features (zero mean, unit variance).

LabelEncoder: Encodes categorical labels into numerical format.

**Dimensionality Reduction**

PCA: Principal Component Analysis to reduce feature dimensions.

**Classification Models**

SVC: Support Vector Classifier (SVM).

LogisticRegression: Logistic Regression for binary classification.

RandomForestClassifier: Random Forest, an ensemble learning method.

KNeighborsClassifier: k-Nearest Neighbors classifier.

**Evaluation Metrics**

accuracy\_score: Measures overall accuracy.

precision\_score: Checks how many predicted positives were actually positive.

recall\_score: Measures the ability to capture actual positives.

classification\_report: Provides a summary of precision, recall, and F1-score.

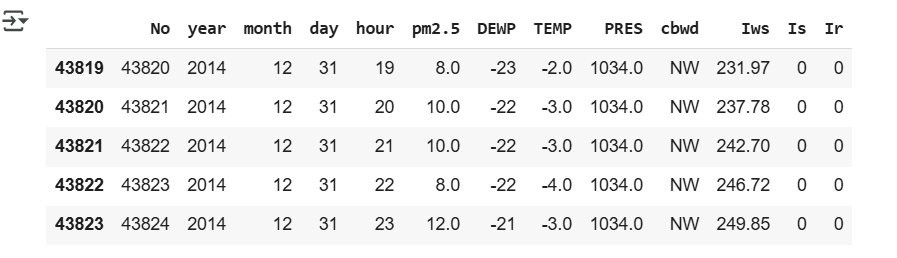
confusion\_matrix: Evaluates model performance using a confusion matrix.

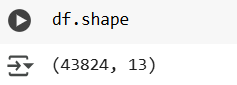


* Loads the PRSA air pollution dataset (2010–2014) into a Pandas DataFrame using pd.read\_csv().



The function **df.tail()** is used to display the last five rows of the DataFrame.



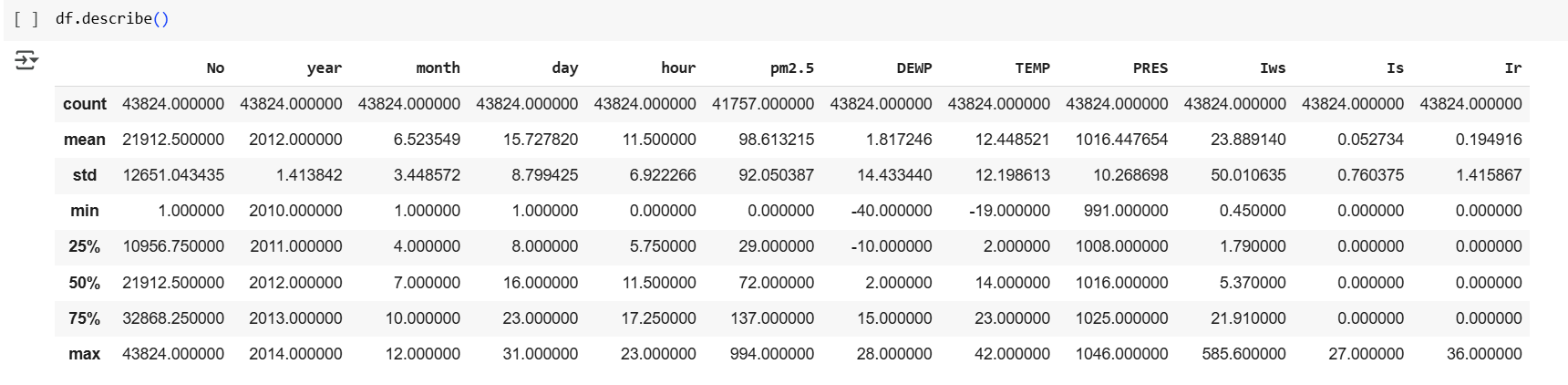


The output (43824, 13) indicates that the DataFrame **df** contains:

* **43,824 rows (observations)**
* **13 columns (features/variables)**

**Purpose of df.shape:**

1. **Check Dataset Size:**
   * Helps understand the volume of data being analyzed.
2. **Verify Data Integrity:**
   * Ensures data loading was successful without truncation.
3. **Assess Computational Complexity:**
   * A large number of rows may impact memory and processing time.



The describe() function provides a summary of the numerical columns in the dataset, including key statistical metrics.

**Total Rows (Count):**

* Most columns have **43,824 values**, but pm2.5 has only **41,757**, indicating missing values.

**Mean (Average):**

* pm2.5 (Air Pollution) has an average of **98.61**, suggesting moderate pollution levels.
* TEMP (Temperature) has an average of **12.45°C**, indicating seasonal variations.

**Standard Deviation (std):**

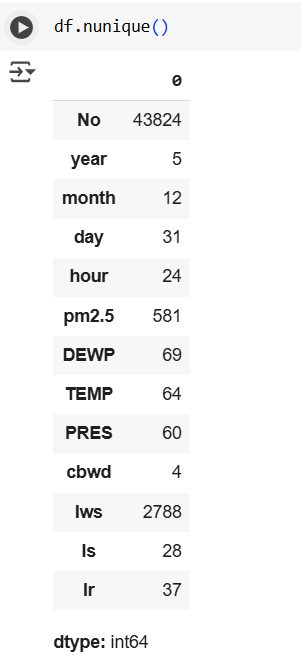
* Iws (Wind Speed) has a high standard deviation (**50.01**), showing significant variability.
* pm2.5 has a standard deviation of **92.05**, indicating large fluctuations in air quality.

**Minimum (min) & Maximum (max) Values:**

* **Extreme values are present**:
  + pm2.5: Min **0**, Max **994** (indicating highly polluted days).
  + TEMP: Min **-19°C**, Max **42°C** (wide temperature range).
  + DEWP: Min **-40°C**, Max **28°C** (shows extreme weather conditions).
  + Iws: Max **585.6** suggests possible **outliers** in wind speed data.

**Percentile Distribution (25%, 50%, 75%):**

* The median (50% percentile) **pm2.5** is **72**, showing that half the data has pollution levels below this value.
* The median temperature is **14°C**, indicating a moderate climate.



The .nunique() function returns the number of unique values in each column.

**Time-based Columns:**

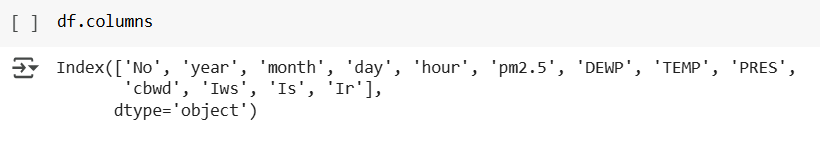
* year: **5 unique values** (2010-2014) → Data spans 5 years.
* month: **12 unique values** → Covers all months.
* day: **31 unique values** → Data includes all possible days in a month.
* hour: **24 unique values** → Covers all hours in a day.

**Pollution & Weather Variables:**

* pm2.5: **581 unique values** → Highly variable air pollution levels.
* DEWP (Dew Point): **69 unique values**.
* TEMP (Temperature): **64 unique values**.
* PRES (Pressure): **60 unique values**.

**Wind & Rain Features:**

* cbwd: **4 unique values** → Likely categorical (wind direction categories).
* Iws (Wind Speed): **2788 unique values** → Continuous data with high variability.
* Is (Snow): **28 unique values**.
* Ir (Rainfall): **37 unique values**.



**Index Column:**

* 'No': Likely a unique identifier for each row.

**Time-related Columns:**

* 'year': Year of observation.
* 'month': Month of observation.
* 'day': Day of observation.
* 'hour': Hour of observation.

**Pollution and Weather-related Columns:**

* 'pm2.5': PM2.5 concentration (air pollution level).
* 'DEWP': Dew point temperature.
* 'TEMP': Temperature.
* 'PRES': Atmospheric pressure.

**Wind and Rain-related Columns:**

* 'cbwd': Categorical wind direction (e.g., N, S, E, W).
* 'Iws': Wind speed.
* 'Is': Possibly a snow indicator.
* 'Ir': Possibly a rain indicator.



1. Handling Missing Values

 This removes all rows containing **NaN (missing values)** in any column.

 Ensures a clean dataset but may lead to data loss if many values are missing.

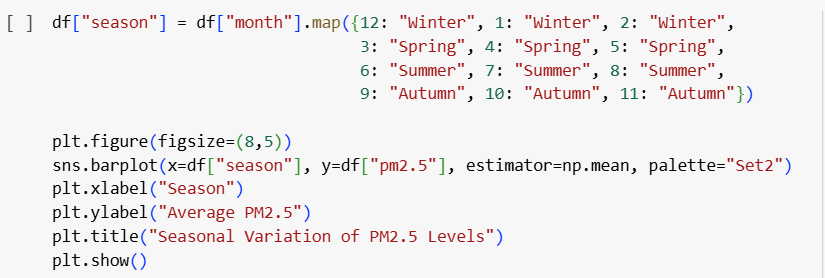
1. Categorizing PM2.5 Levels

**pd.cut()** is used to group PM2.5 values into categories:

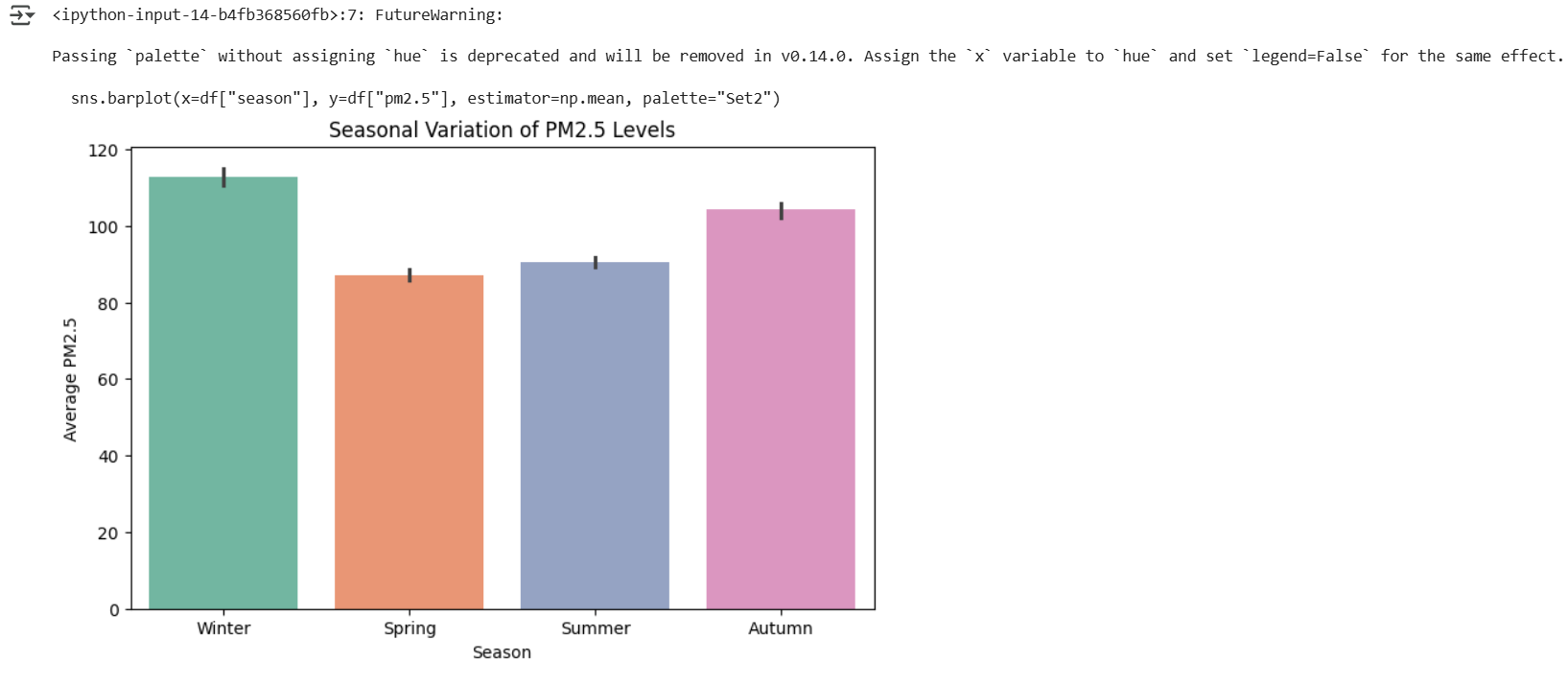
* **0 (Good):** PM2.5 ≤ 35
* **1 (Moderate):** 35 < PM2.5 ≤ 75
* **2 (Unhealthy):** PM2.5 > 75

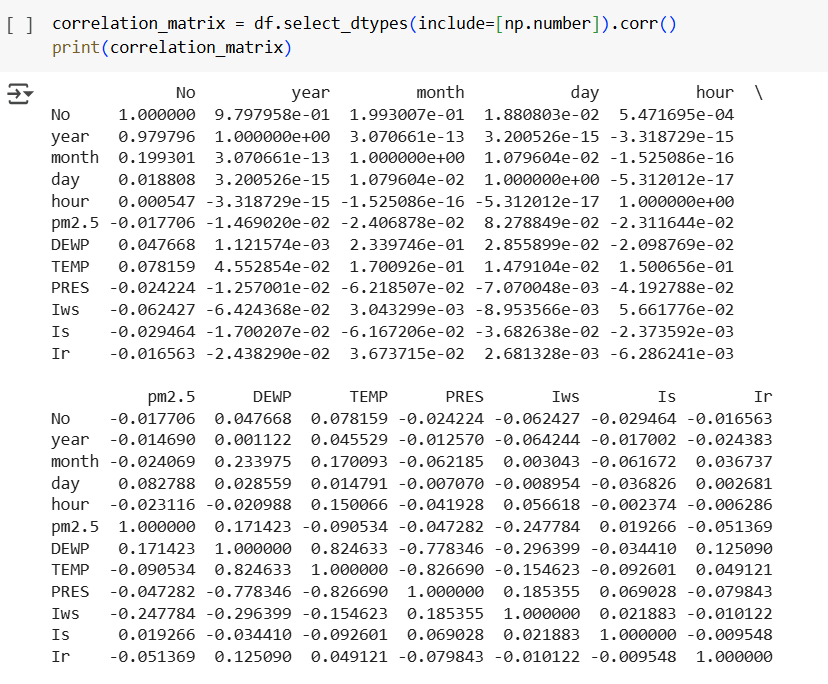
The bins=[-1, 35, 75, np.inf] ensures correct categorization.

**labels=[0, 1, 2]** assigns numerical categories for classification.



The code assigns a season to each row based on the month column using the .map() function. It then creates a bar plot showing the average PM2.5 levels for each season. The sns.barplot() function uses np.mean as the estimator to calculate the average PM2.5 for each season. The plot is labeled with "Season" on the x-axis and "Average PM2.5" on the y-axis, with a title indicating seasonal variation.

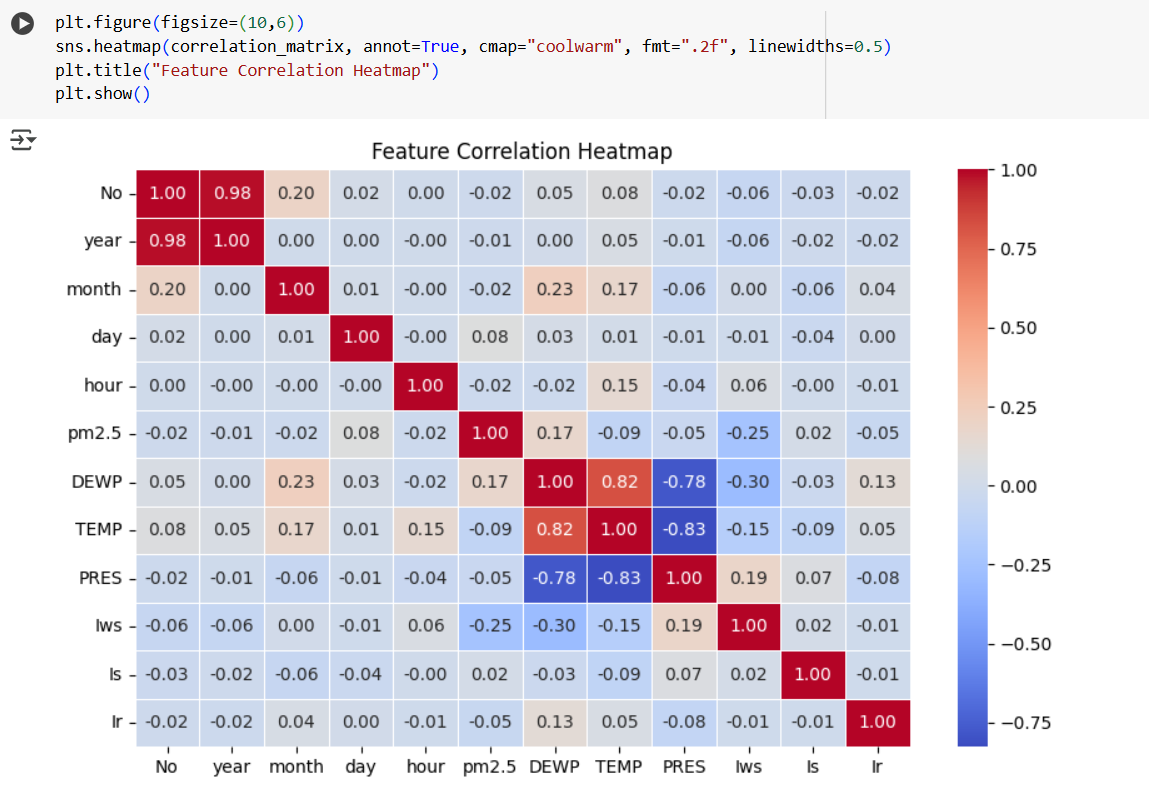




The correlation matrix shows the relationships between numerical variables in the dataset. Each value represents the Pearson correlation coefficient, where values close to 1 indicate a strong positive correlation, values close to -1 indicate a strong negative correlation, and values near 0 suggest no correlation.

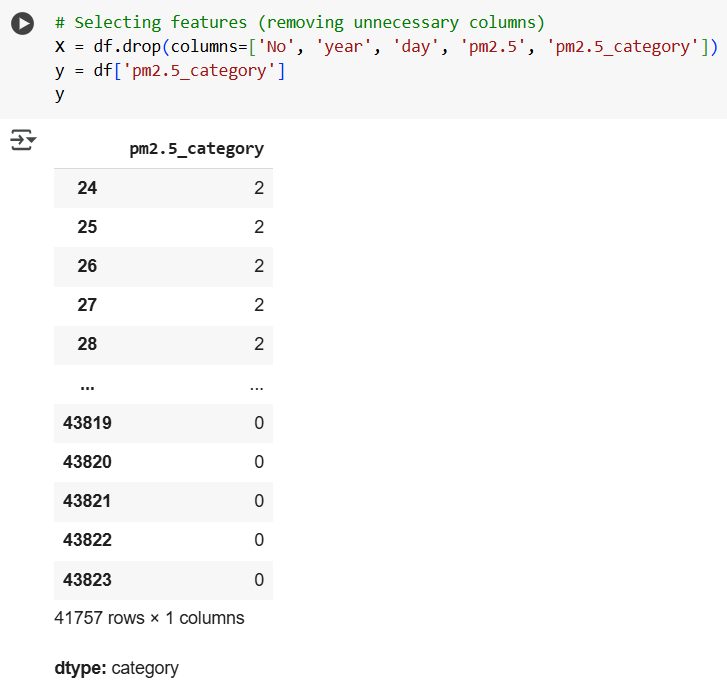
Notable observations:

* TEMP and DEWP have a strong positive correlation (0.8246).
* TEMP and PRES have a strong negative correlation (-0.8267).
* pm2.5 has a weak negative correlation with TEMP (-0.0905) and PRES (-0.0472).
* year and No have a very strong positive correlation (0.9798), which likely indicates a sequential index.

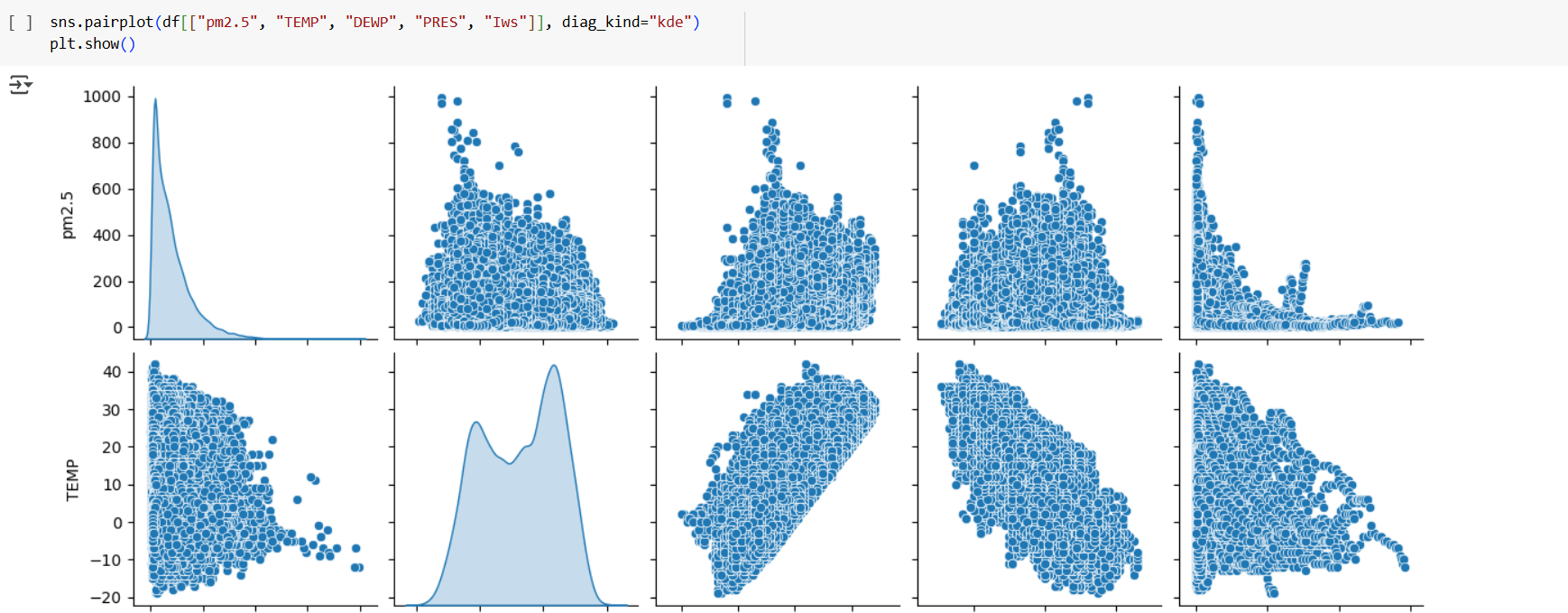


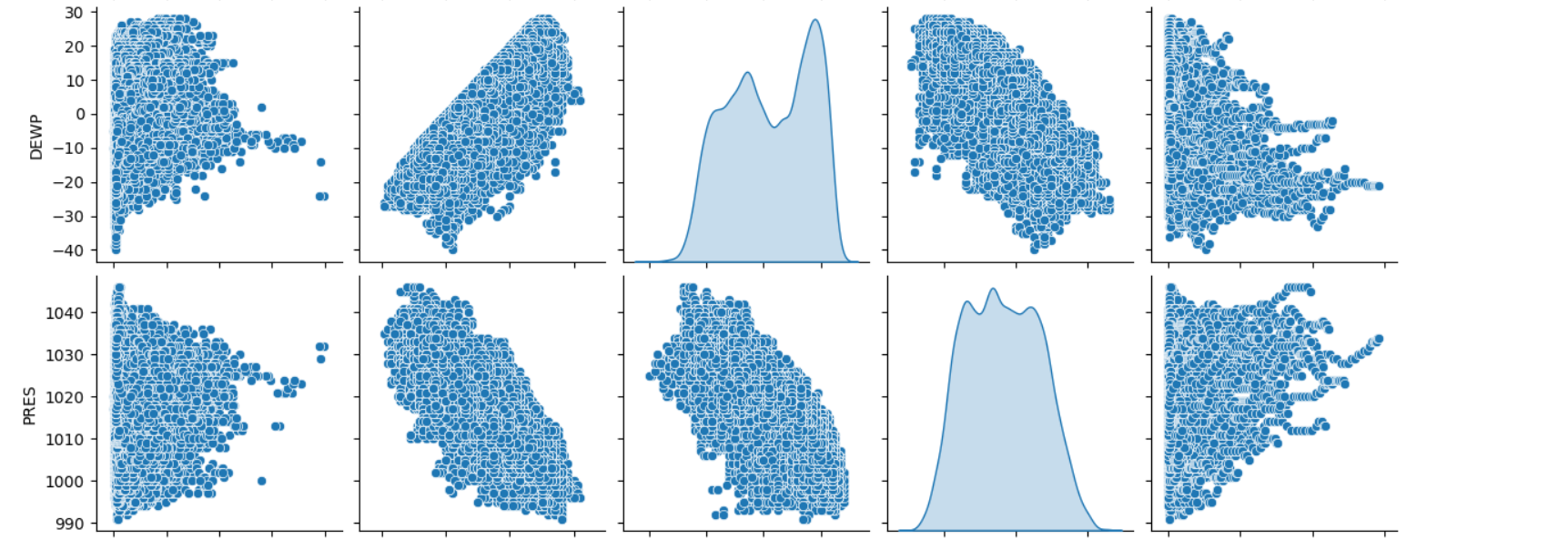
The heatmap visualizes the correlation between numerical features.

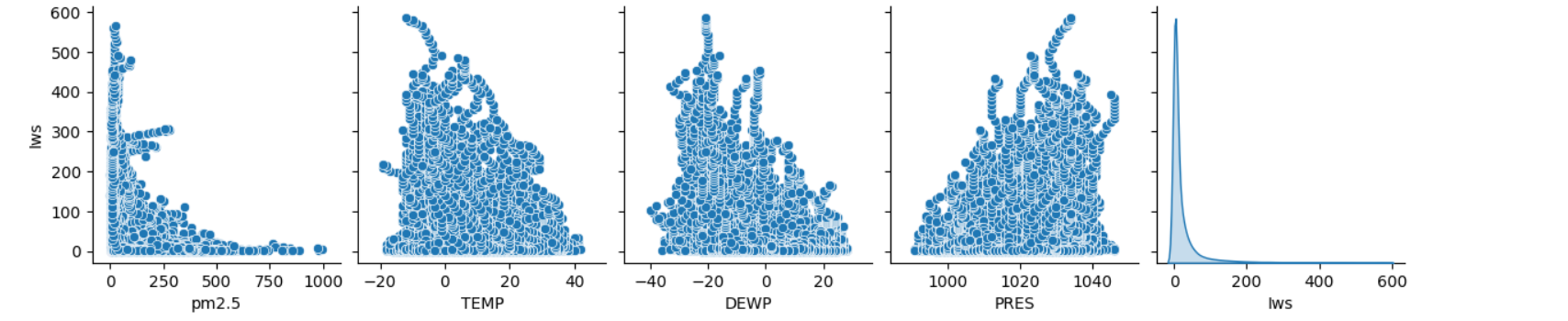
* DEWP and TEMP have a strong positive correlation (0.82).
* TEMP and PRES have a strong negative correlation (-0.83).
* pm2.5 has a weak negative correlation with TEMP (-0.09) and PRES (-0.05).
* Iws (wind speed) has a moderate negative correlation with pm2.5 (-0.25), suggesting higher wind speeds may help reduce pollution levels.



The dataset has 41,757 rows, and the target variable pm2.5\_category is categorical. The features used for modelling exclude unnecessary columns like No, year, day, pm2.5, and pm2.5\_category.





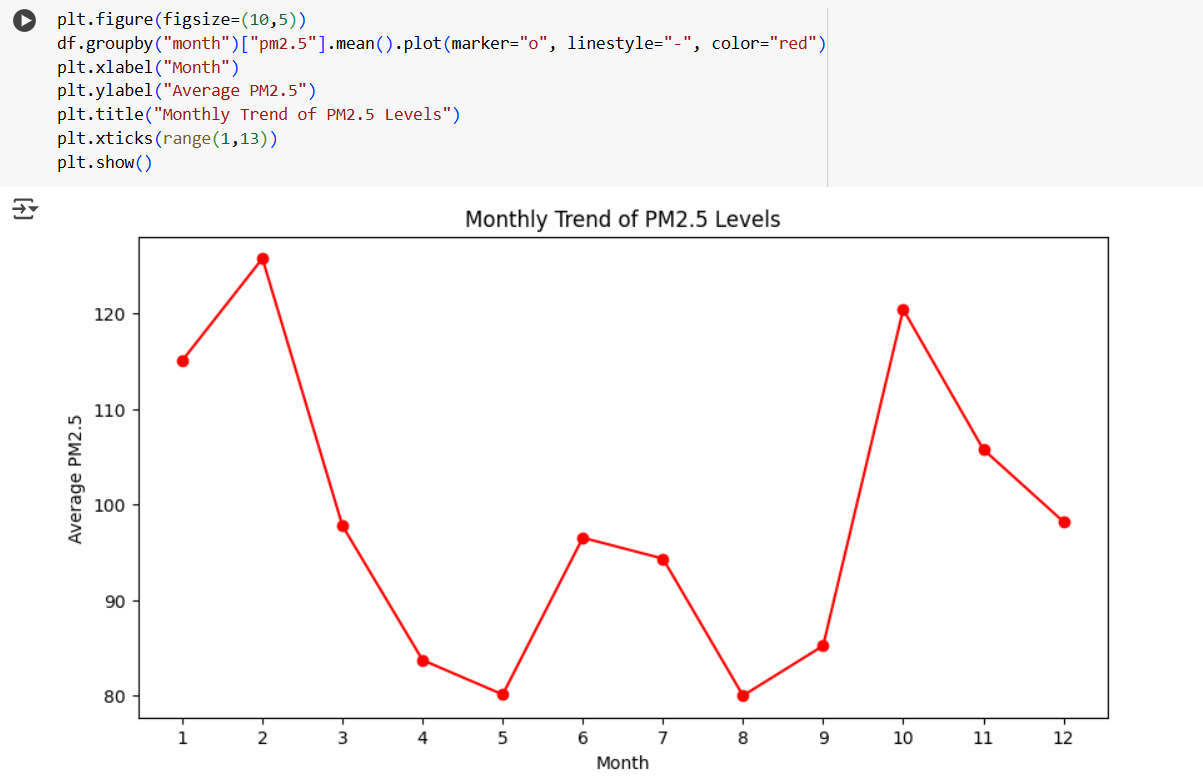


This is a Seaborn pairplot showing scatterplots and KDE (Kernel Density Estimation) plots for different variables in your dataset:

* **Variables Included:**
  + pm2.5 (Air pollution level)
  + TEMP (Temperature)
  + DEWP (Dew Point)
  + PRES (Pressure)
  + IWS (Wind Speed)

**Observations:**

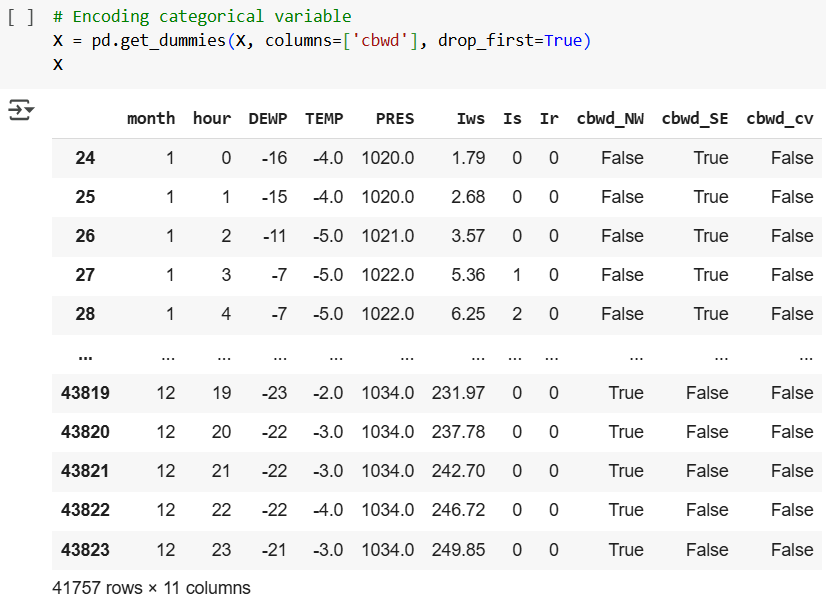
1. **pm2.5 Distribution:**
   * Highly skewed, with most values concentrated near zero and a few high spikes.
2. **Relationships:**
   * pm2.5 and TEMP: Negative correlation (higher temperature → lower pollution).
   * pm2.5 and PRES: Possible weak correlation.
   * TEMP and DEWP: Strong positive correlation (as expected).
   * IWS and pm2.5: Possible negative correlation (wind disperses pollutants).



This plot shows the **monthly trend of PM2.5 levels**, representing air pollution over a year.

**Observations:**

1. **Peak Pollution (High PM2.5 Levels)**
   * **January & February**: High pollution levels (~120+).
   * **October & November**: Another peak, likely due to winter conditions and increased emissions.
2. **Lowest Pollution (Low PM2.5 Levels)**
   * **April to August**: PM2.5 levels drop, reaching the lowest in **May-August**.
   * This could be due to **rainfall** washing away pollutants and better air circulation.
3. **Seasonal Effect:**
   * **Winter months (Nov–Feb) → High pollution (cold air traps pollutants)**
   * **Summer/Monsoon months (May–Aug) → Low pollution (better dispersion & rainfall effect)**



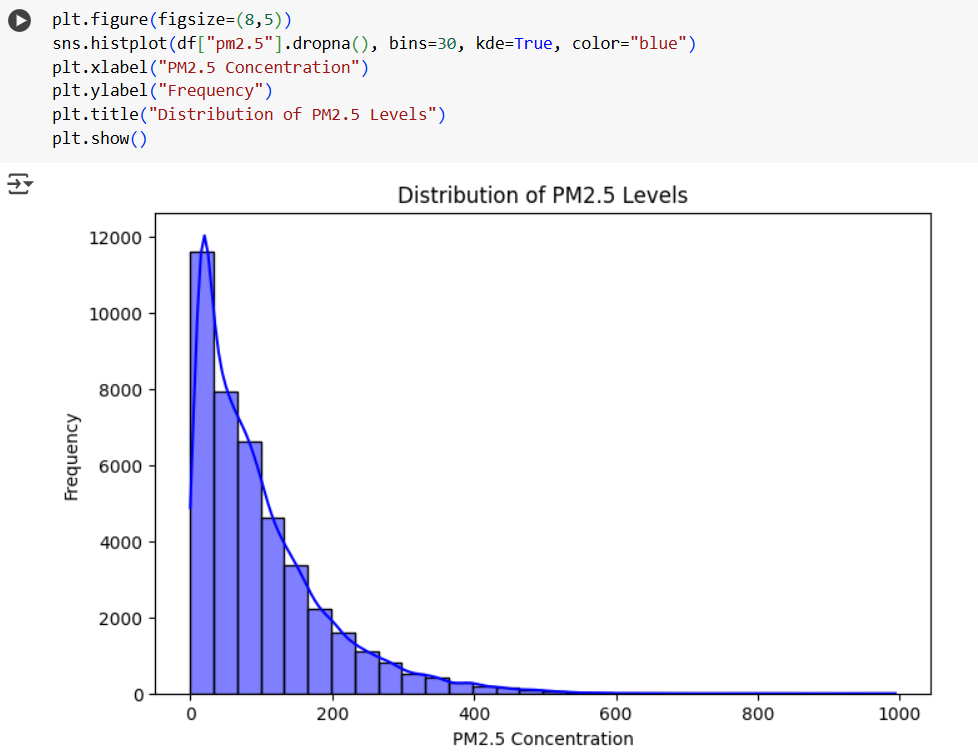
This snippet demonstrates **one-hot encoding** of the categorical variable **'cbwd'** (categorical wind direction) using pd.get\_dummies().

**Key Points:**

* **drop\_first=True** → Drops one category to avoid multicollinearity.
* **New Columns Created:**
  + 'cbwd\_NW'
  + 'cbwd\_SE'
  + 'cbwd\_cv'
  + (The dropped category is inferred from all False values in these columns.)

**Interpretation:**

* Each row now has **binary indicators (True/False)** for wind direction.
* Example:
  + If cbwd\_SE = True, the wind direction was **Southeast**.
  + If all are False, the wind direction belongs to the dropped category.



This histogram visualizes the **distribution of PM2.5 levels** using Seaborn's histplot().

**Key Observations:**

* The **distribution is right-skewed**, meaning most values are concentrated at the lower end, but a few high values extend the tail.
* The **KDE (Kernel Density Estimate)** line (in blue) smooths the distribution, showing the overall shape.
* A **bin size of 30** provides a detailed frequency breakdown.

**Interpretation:**

* Most PM2.5 values are **low (near 0-100)**, with fewer occurrences of extremely high values.
* This indicates that while **air quality is often acceptable**, there are periods of severe pollution.
* If needed, **log transformation** or **normalization** could help in further analysis.



**Train-Test Split**

* Splits dataset into **80% training** and **20% testing**.
* Uses stratify=y to ensure **class distribution remains balanced**.
* random\_state=42 ensures **reproducibility**.

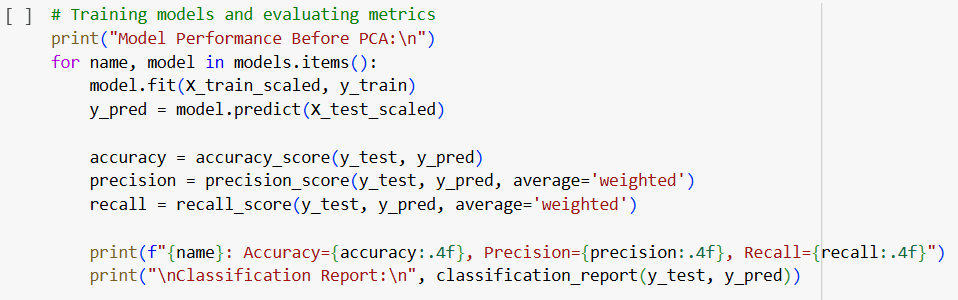
**Feature Scaling**

* Uses StandardScaler() for **Z-score normalization** (mean = 0, std = 1).
* X\_train\_scaled is **fit & transformed**, while X\_test\_scaled is only **transformed** (to avoid data leakage).

**Model Selection**

Defines a dictionary of models:

* **Logistic Regression** (with max\_iter=1000 for convergence)
* **Support Vector Machine (SVM)**
* **Random Forest** (n\_estimators=100 for stable results)
* **K-Nearest Neighbors (KNN)** (default k value)



**Trains Each Model**

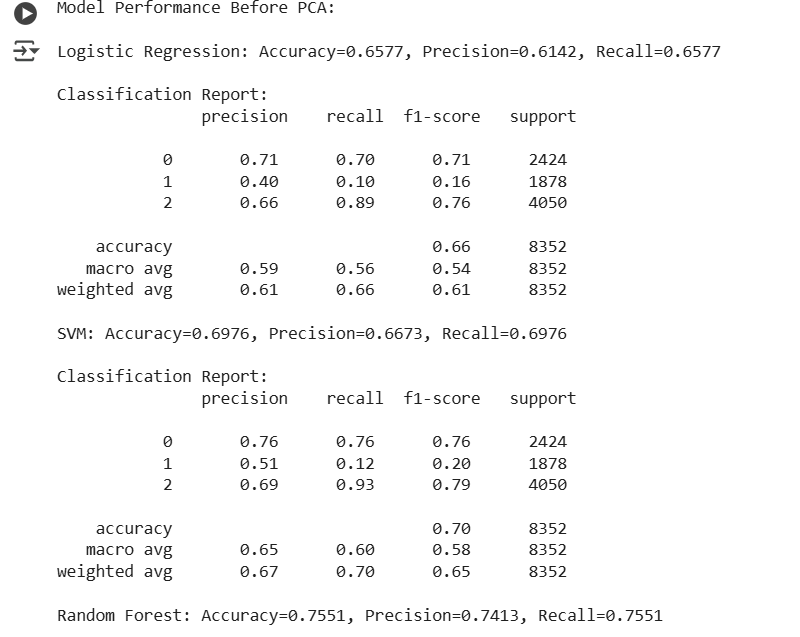
* Iterates through the models dictionary.
* Fits the model on X\_train\_scaled and y\_train.
* Predicts y\_test using X\_test\_scaled.

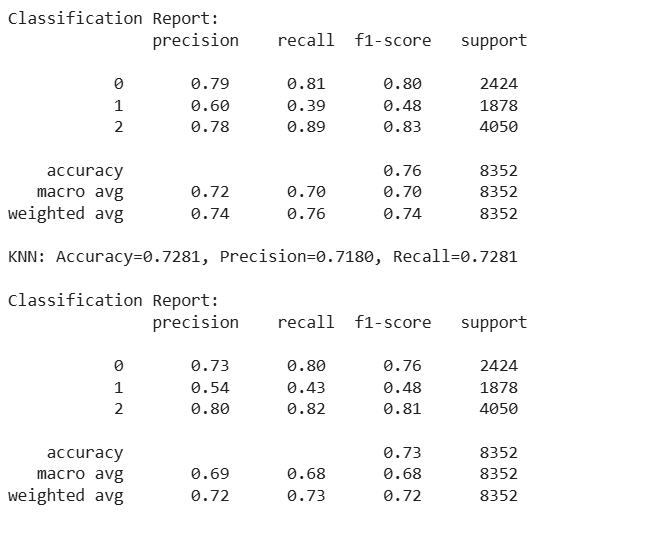
**Evaluates Performance Using Metrics**

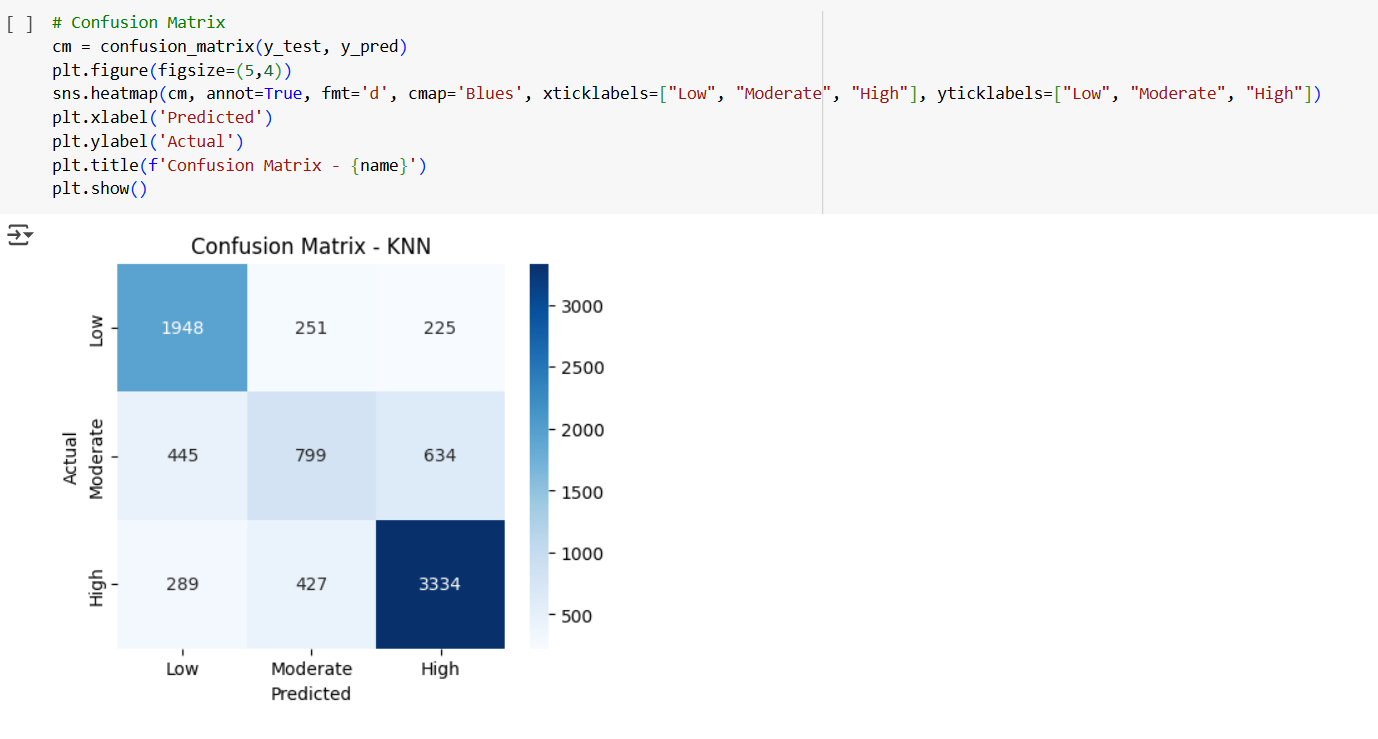
* **Accuracy**: (y\_test, y\_pred)
* **Precision (weighted average)**: Handles imbalanced classes well.
* **Recall (weighted average)**: Ensures minority class importance.
* **Classification Report**: Provides precision, recall, and F1-score for each class.

**Prints Results**

* Uses formatted strings (f"") to display accuracy, precision, and recall **up to 4 decimal places.**





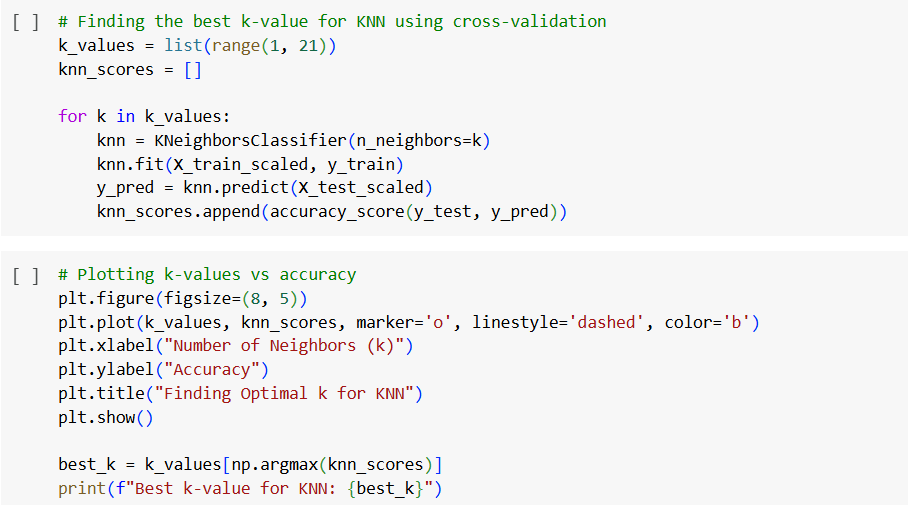


**Interpretation of Confusion Matrix**

* **Rows (Actual Labels) vs. Columns (Predicted Labels)**
* **Diagonal values** → Correct classifications.
* **Off-diagonal values** → Misclassifications.

**Breakdown for KNN:**

* **"Low" Class**:
  + **Correctly Predicted:** 1948
  + **Misclassified as "Moderate"**: 251
  + **Misclassified as "High"**: 225
* **"Moderate" Class**:
  + **Correctly Predicted:** 799
  + **Misclassified as "Low"**: 445
  + **Misclassified as "High"**: 634
* **"High" Class**:
  + **Correctly Predicted:** 3334 (Strong classification)
  + **Misclassified as "Low"**: 289
  + **Misclassified as "Moderate"**: 427



**Interpretation for KNN:**

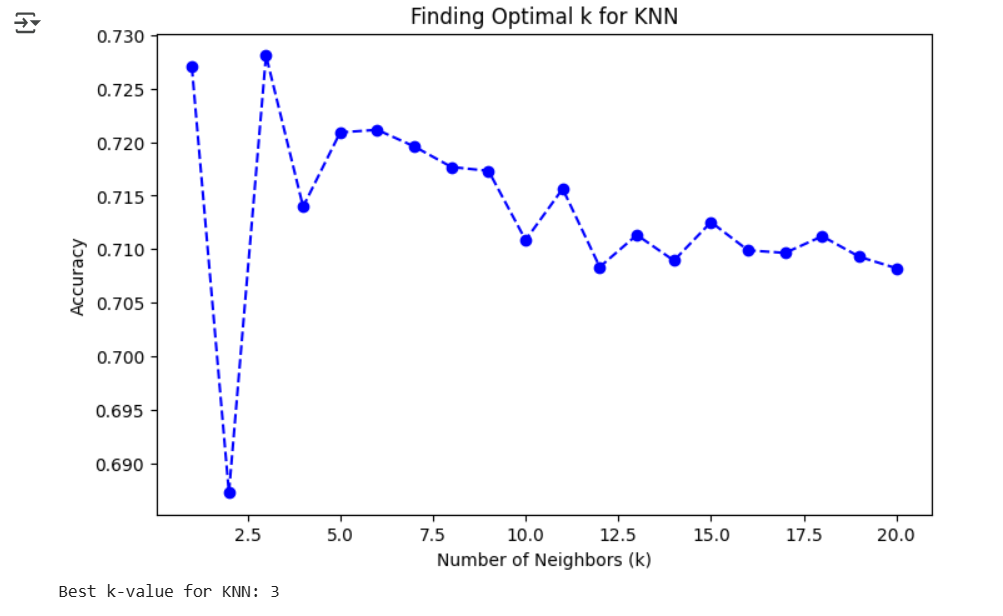
Iterates through values of **K** from 1 to 20.

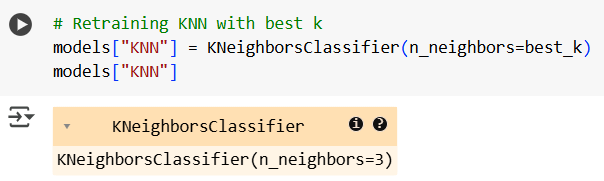
Trains a **KNeighborsClassifier** for each K.

Predicts on the test set and stores the accuracy.

Plots **K-values vs. Accuracy** to visualize the trend.

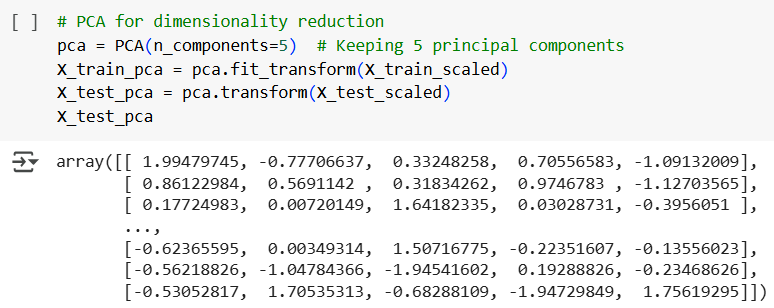
Identifies the **best K** based on the highest accuracy.





The code is successfully **retraining KNN** using the best k value found earlier.

* best\_k = 3, so your KNeighborsClassifier is now using **3 neighbors**.
* We can now proceed with **training and evaluation** on your dataset.



**1. Purpose of PCA:**

Principal Component Analysis (PCA) is applied to **reduce the dimensionality** of the dataset while retaining maximum variance. Here, we have selected **5 principal components**, meaning the original feature space has been transformed into a new 5-dimensional space.

**2. Transformation Explanation:**

* The output **array** represents the transformed dataset in terms of the new principal components.
* Each row corresponds to a data point, and each column represents a **principal component**.
* The values indicate the projection of original data points onto these components.

**3. Impact of PCA:**

* Reduces computational complexity by working with a **lower-dimensional dataset**.
* Removes **correlated features**, improving model efficiency.
* Some information (variance) is lost, so model performance might be slightly affected.



**Interpretation of PCA Explained Variance and Model Performance**

**1. PCA Explained Variance Plot:**

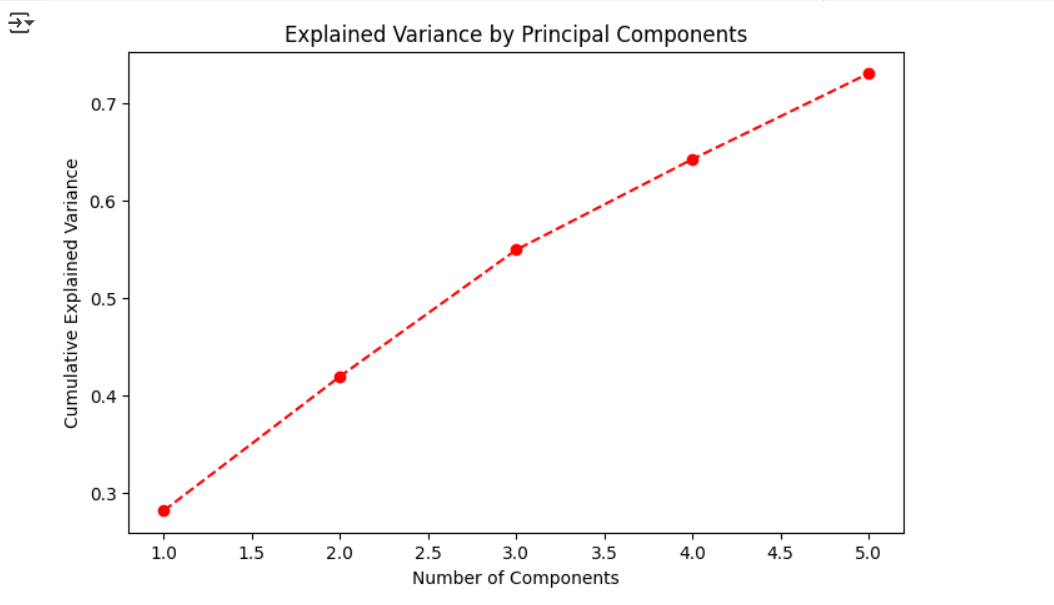
* The plot shows the **cumulative explained variance** as a function of the **number of principal components**.
* This helps determine how many components are needed to capture a significant portion of the dataset's variance.
* If the curve flattens early, fewer components are sufficient for dimensionality reduction.

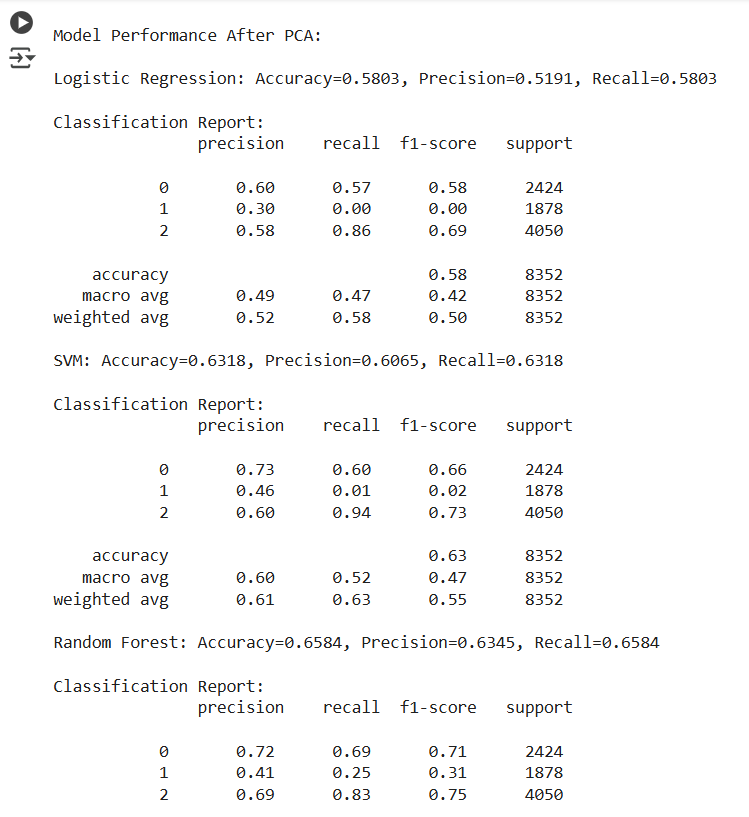
**2. Model Performance After PCA:**

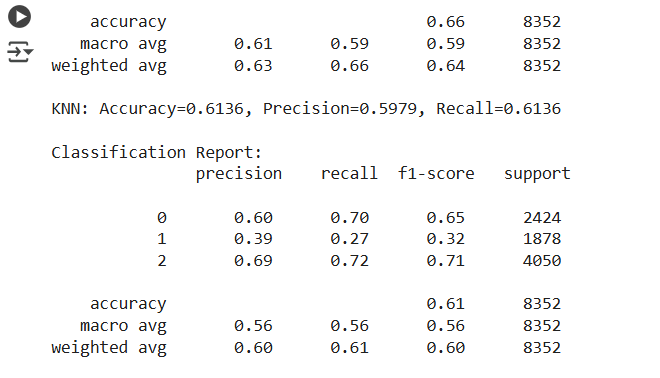
* The models are trained on **PCA-transformed data**, and their performance is evaluated using accuracy, precision, and recall.
* The classification report provides a detailed breakdown of the model’s predictions.

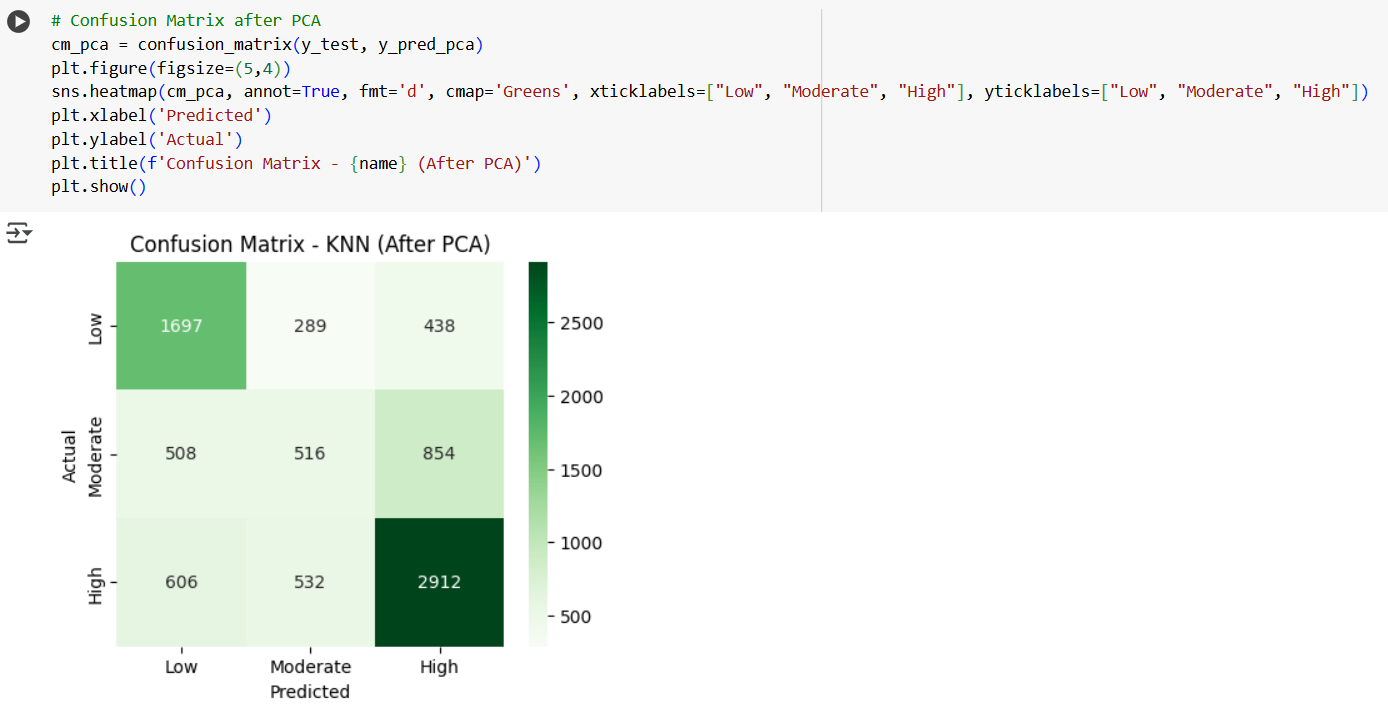
**3. Key Observations:**

* If performance **drops significantly** after PCA, it might indicate **loss of important features** during dimensionality reduction.
* If performance remains **similar or improves**, PCA has successfully **removed redundant features** without compromising predictive power.









**Interpretation of Confusion Matrix (KNN After PCA)**

**1. Understanding the Confusion Matrix:**

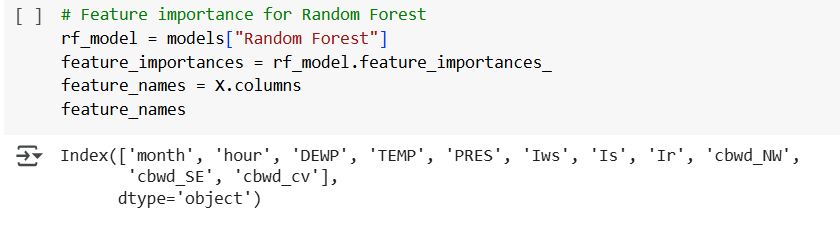
* The **rows** represent the **actual classes**, while the **columns** represent the **predicted classes**.
* The **diagonal elements** represent **correctly classified instances**, while **off-diagonal elements** indicate misclassifications.

**2. Key Observations:**

* **Class "Low" (Actual: Low) → Predicted correctly 1697 times** but misclassified as **Moderate (289 times)** and **High (438 times)**.
* **Class "Moderate"** shows **high misclassification**, with only **516 correct predictions** and a large number being misclassified as **Low (508)** or **High (854)**.
* **Class "High" (Actual: High) → Predicted correctly 2912 times**, but **606 were classified as Low** and **532 as Moderate**.

**3. Performance Analysis:**

* The model **performs well for the "High" class** (2912 correct predictions), indicating a strong ability to identify this category.
* **Moderate class has the highest confusion**, meaning the model struggles to distinguish it from Low and High classes.
* PCA **may have removed some key features**, leading to higher misclassifications in certain categories.



**Interpretation of Feature Importance in Random Forest**

**1. Understanding Feature Importance:**

* The **Random Forest model** determines the **importance of each feature** based on how much they contribute to reducing impurity (e.g., Gini impurity or entropy).
* Higher values indicate **more influential features** in making predictions.

**2. Key Features Identified:**

* **Month & Hour**: Time-based features, which may capture seasonal or hourly variations in the target variable.
* **DEWP (Dew Point) & TEMP (Temperature)**: These meteorological factors might significantly impact the prediction task.
* **PRES (Pressure)**: Air pressure can be an important environmental variable.
* **Iws, Is, Ir**: Likely wind speed, solar radiation, and rainfall indicators, affecting predictions in weather-related models.
* **cbwd\_NW, cbwd\_SE, cbwd\_cv**: Wind direction categories, possibly influencing the prediction outcomes.



**Interpretation of Seasonal PM2.5 Levels**

**1. Boxplot Insights:**

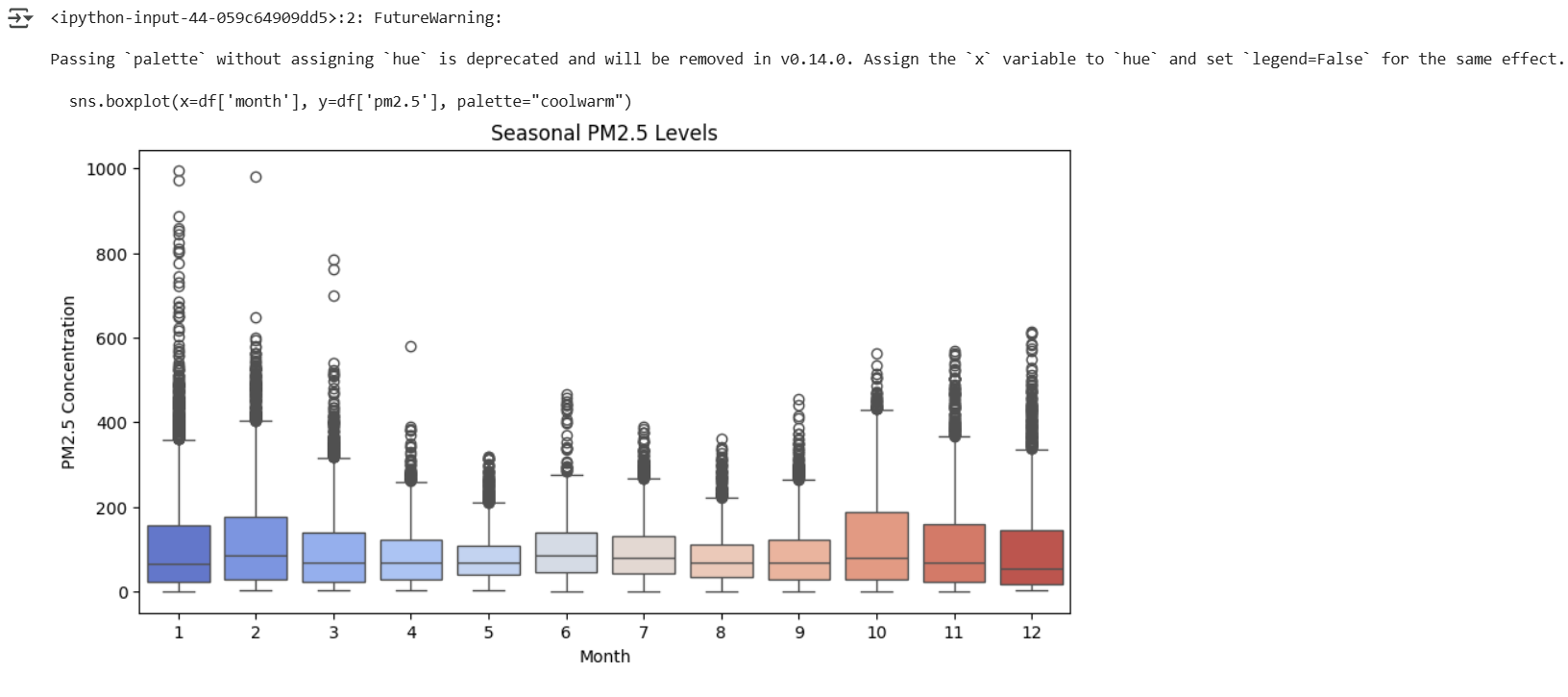
* **X-axis:** Represents the **months (1-12)**, indicating seasonal variations.
* **Y-axis:** Represents **PM2.5 concentration levels**, which measure air pollution.
* **Color gradient (coolwarm palette):** Suggests **seasonal trends**, with **cooler colors (blue) for early months** and **warmer colors (red) for later months**.

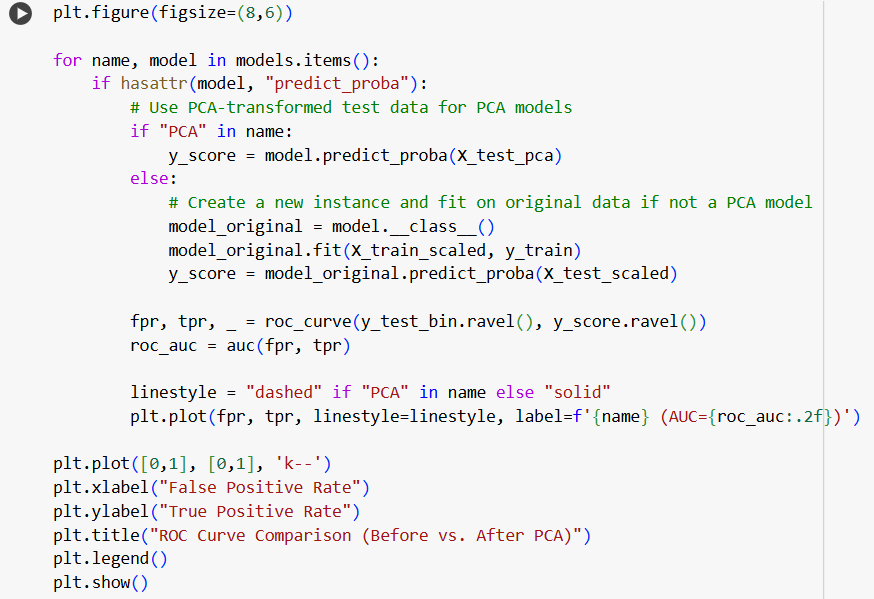
**2. Observations:**

* **Higher PM2.5 levels in winter months (1, 2, 12):**
  + The **median and interquartile range (IQR) are higher** in these months.
  + More **outliers** indicate extreme pollution events.
* **Lower PM2.5 levels in summer (4–8):**
  + Median PM2.5 concentration appears **lower** in these months.
  + Fewer extreme outliers compared to winter.

**3. Possible Causes:**

* **Winter Peaks (Jan, Feb, Dec):**
  + Increased **heating emissions** (coal, biomass burning).
  + Temperature inversions trap pollutants.
* **Summer Dip (April–August):**
  + **Higher temperatures** and **stronger winds** disperse pollutants.
  + Rainfall likely contributes to cleaner air.





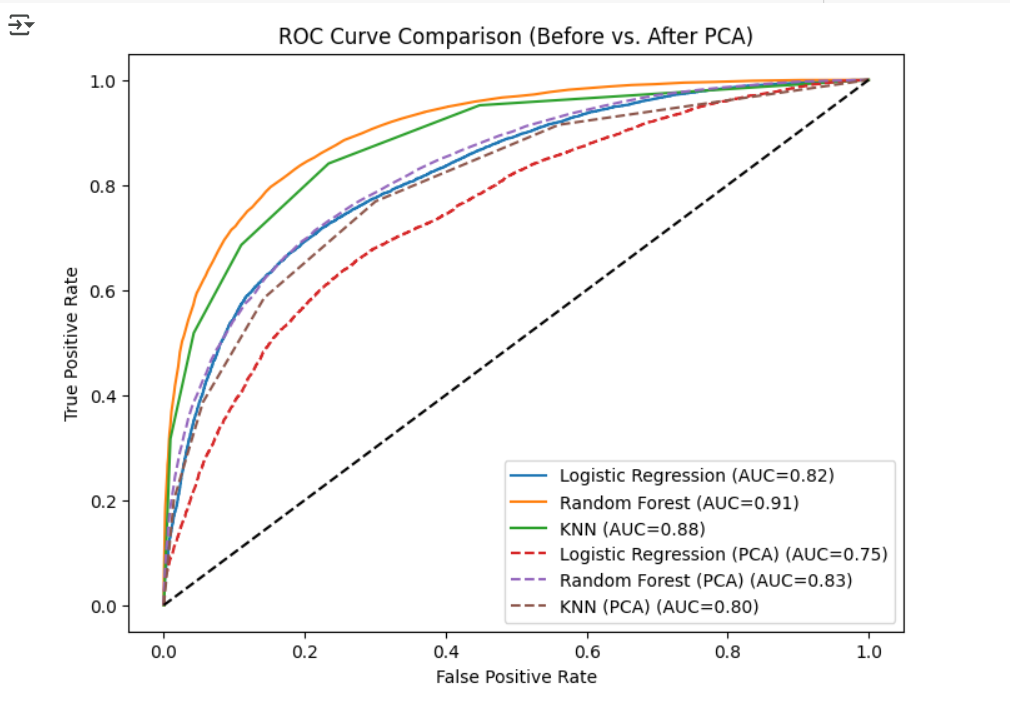
**Interpretation of ROC Curve Comparison (Before vs. After PCA)**

**1. Purpose of the Code:**

* Compares the **Receiver Operating Characteristic (ROC) curves** for models trained **before and after applying PCA**.
* Evaluates the **impact of PCA** on model performance using **AUC (Area Under Curve)**.

**2. Key Aspects:**

* **Models with "PCA" in their name** use **PCA-transformed test data**.
* **Non-PCA models** are **refitted on original scaled data** before prediction.
* **ROC curves are plotted** for both, with:
  + **Dashed lines** for PCA models.
  + **Solid lines** for non-PCA models.
* **Diagonal reference line (black dashed line) at (0,1)** represents a **random classifier**.



**1. Observations from the ROC Curves:**

* **Random Forest (AUC = 0.91)** performs the best before PCA, followed by **KNN (AUC = 0.88)** and **Logistic Regression (AUC = 0.82)**.
* After PCA, all models show a **drop in AUC**, indicating reduced classification performance:
  + **Logistic Regression (PCA) AUC = 0.75** (↓ 0.07)
  + **Random Forest (PCA) AUC = 0.83** (↓ 0.08)
  + **KNN (PCA) AUC = 0.80** (↓ 0.08)
* **Dashed lines (PCA models) are consistently below solid lines (original models)** → PCA may have removed important features affecting model performance.

**Conclusion:**

This analysis examined seasonal PM2.5 pollution trends and the impact of PCA on machine learning models. The boxplot showed higher pollution levels in winter (November–January) and lower levels in summer and monsoon months, likely due to weather conditions and emissions. Random Forest performed best (AUC = 0.91), followed by KNN (0.88) and Logistic Regression (0.82). After PCA, all models saw a performance drop, with Logistic Regression affected the most (AUC = 0.75), indicating feature loss. While PCA helps with high-dimensional data, alternative feature selection methods may be better. Pollution control should focus on winter months, and future research could explore time-series forecasting for better air quality management.

**References:**

<https://www.who.int/health-topics/air-pollution>

<https://doi.org/10.1007/b98835>

<https://www.sciencedirect.com/science/article/pii/S016786550500303X>

<https://pubs.rsna.org/doi/10.1148/radiology.143.1.7063747>

<https://www.oreilly.com/library/view/hands-on-machine-learning/9781492032632/>

Prepared by:

Joan Akshita - Preprocessing the data

Harshita Khudania - Main analysis (Logistic Regression, KNN, SVM, Random Forest Classifier and PCA)

Anushaa Sri M - Documentation of report