1) Introduction

Air quality is a growing concern in urban environments worldwide, where pollutants such as PM2.5, nitrogen dioxide, and sulphur dioxide are common byproducts of industrial activity and vehicle emissions. Poor air quality can lead to serious health problems, including respiratory and cardiovascular diseases, which affect millions of people each year.

In this project, we focus on predicting the Air Quality Index (AQI), a widely used measure of pollution levels. The AQI is a continuous numeric value that reflects the severity of air pollution and its potential impact on health. We use machine learning methods to predict AQI based on historical environmental and pollutant data. The application domain of this project is environmental monitoring, specifically predicting air pollution levels in Beijing, China, one of the most polluted cities globally.

This report explores the application of two machine learning methods, Random Forest Regressor and Ridge Regression, to predict AQI values. Section 2 outlines the problem formulation, describing the dataset and the features used for the prediction task. Section 3 discusses the methods, including preprocessing steps, feature selection, and the choice of machine learning models and evaluation metrics. In Section 4, we present the results of both models, compare their performance, and select the final method. Finally, Section 5 summarises the findings, discusses limitations, and proposes potential improvements for future work. Section 6 is bibliography and Section 7 is appendix.

Platforms such as Kaggle have similar projects predicting air quality using methods ranging from time-series analysis to deep learning models such as LSTM. However, in this study, the Random Forest Regressor and Ridge Regression were chosen due to their different aspects, which will be discussed in detail later in this report. Therefore, this project differs from the other studies in the methods and the loss function aspects.

2) Problem Formulation

2.1) Formalising the ML Problem

The goal of this project is to predict the Air Quality Index (AQI) in Beijing using historical environmental and pollutant data. This is a supervised regression problem where the AQI, a continuous variable, is predicted based on multiple environmental features.

- **Input:** A set of environmental features, such as pollutant concentrations and meteorological data.
- **Output:** A continuous target variable, the AQI, representing the air pollution level.

2.2) Dataset Source, Data Points, and Features

The dataset is sourced from the UCI Machine Learning Repository, specifically the Beijing PM2.5 dataset. It has hourly pollutant and meteorological data, which were turned into daily values for this project. This dataset is well-suited for air quality prediction tasks due to its detailed coverage of pollution and environmental conditions over multiple years in a highly polluted city. Specifically, the dataset contains 8,388 daily observations of air quality and meteorological conditions from 2010 to 2014. Each data point is multidimensional, meaning it includes multiple feature variables representing different aspects of the environment and pollution levels. Specifically, the features are:

- **PM2.5 (continuous)**: The concentration of fine particulate matter (μg/m^3), a critical pollutant known for its adverse health effects.
- **Dew Point (continuous)**: The temperature at which air becomes saturated with moisture (°C), affecting atmospheric stability.
- **Temperature (continuous)**: The ambient atmospheric temperature (°C).
- **Humidity (continuous)**: The percentage of moisture present in the air (%), which influences air quality.
- **Wind Speed (continuous)**: The wind speed (m/s), which impacts the dispersion of pollutants.
- **Pressure (continuous)**: The atmospheric pressure (hPa), which affects the movement and concentration of pollutants.

The target label, **AQI (continuous)**, is a numeric value that indicates the level of air pollution on a given day. Higher AQI values correspond to worse air quality, posing greater health risks. Thus, the dataset has a multidimensional input with six continuous feature variables. The model uses all of these variables to predict the AQI, leveraging the relationships between environmental factors and pollution levels.

3) Methods

3.1) Dataset Preprocessing

To prepare the data for machine learning models, several preprocessing steps were performed:

- **Handling Missing Data**: The dataset had some missing values in the pollutant measurements. Rows containing missing data were removed to ensure the integrity of the dataset and to avoid introducing bias through imputation.
- **Normalisation**: The features in the dataset have different scales (e.g., temperature in °C, wind speed in m/s). To ensure that all features are comparable and to improve model performance, Min-Max scaling was applied to normalise the feature values between 0 and 1.

3.2) Feature Selection

All six features were retained as each has a known impact on air quality. We did not need any further reduction or feature engineering since the number of features is small enough and each provides useful information for the model. PM2.5 was kept as the most significant pollutant influencing AQI. Meteorological factors such as temperature, wind speed, and humidity play a key role in transferring pollutants and affecting air quality. Dew point and pressure were also included as they impact atmospheric stability and pollutants in the air.

3.3) Machine Learning Models

Random Forest Regressor:

- **Motivation**: Random Forest was chosen because of its ability to model complex, nonlinear relationships between features and the target variable. Given the complex interactions between environmental factors and air quality, Random Forest's ensemble approach reduces overfitting and provides insights into feature importance.
- **Hypothesis Space**: Random Forest combines multiple decision trees to predict the target variable. Each tree in the forest is built on a subset of the data, and the final prediction is the average of the predictions from individual trees.
- **Loss Function**: The model is trained using Mean Squared Error (MSE) as the loss function. MSE penalises larger errors more heavily, making it suitable for regression problems like AQI prediction where large deviations could have significant health implications.

Ridge Regression:

- **Motivation**: Ridge Regression was selected as a linear model to serve as a comparison to the more complex Random Forest. It is a regularised version of linear regression that helps reduce overfitting by adding a penalty on the magnitude of the coefficients, making it suitable for cases with potential multicollinearity among features.
- **Hypothesis Space**: Ridge Regression assumes a linear relationship between the input features and the target variable. The regularisation term prevents the coefficients from growing too large, thereby reducing the risk of overfitting.
- **Loss Function**: Like Random Forest, Ridge Regression also uses Mean Squared Error (MSE) as the loss function to measure the prediction accuracy by penalising larger errors. They have the same motivation.

3.4) Model Validation
To evaluate the performance of the models and avoid overfitting, the dataset was split into three subsets:

- **Training set (75%)**: This is the largest portion of the data used to train the machine learning models. The idea behind this split is to ensure that the models can learn from as much data as possible to understand the relationships between the input features and the output.
- **Validation set (10%)**: This smaller portion of the data is used to tune model hyperparameters, like the number of trees in the Random Forest or the regularisation strength in Ridge Regression. It helps avoid overfitting, which means preventing the model from becoming too specific to the training data and failing on unseen data.
- **Test set (15%)**: After the models are trained and validated, this set is used to evaluate how well the final model performs on new, unseen data. The goal is to get an unbiased estimate of the model's generalisation ability.

I used the 75/10/15 split to maintain a balance between having enough data for model training while reserving sufficient data for both hyperparameter tuning and final evaluation. This split provides a straightforward approach that ensures efficiency and avoids the computational overhead of more complex validation methods like k-fold cross-validation.

4. Results
4.1 Model Performance
The performance of the Random Forest Regressor and Ridge Regression models was evaluated using Mean Squared Error (MSE) on both the training, validation, and test sets. The results are as follows:

Random Forest Regressor:
- **Training Error (MSE)**: 0.0469
- **Validation Error (MSE)**: 1.1528
- **Test Error (MSE)**: 0.6670

The Random Forest model had a very low training error, which suggests it fit the training data well. However, the much higher validation error points to potential overfitting, meaning the model struggled to generalise to new data. The improvement in the test error indicates that the model performed better on the unseen test data, but the large gap between training and validation errors suggests some degree of overfitting.

Ridge Regression:
- **Training Error (MSE)**: 0.1512
- **Validation Error (MSE)**: 0.1669
- **Test Error (MSE)**: 0.1522

Ridge Regression demonstrated consistent performance across all datasets. The small gap between the training, validation, and test errors indicates that the model was able to generalise well to unseen data. Ridge Regression outperformed Random Forest, especially in terms of validation and test errors, indicating that Ridge Regression was better able to capture the relationships between the features and AQI without overfitting. The results also suggest that the linear nature of the Ridge Regression model provided a simpler and more effective solution for this problem.

4.2 Final Model Selection
In comparing the models, Ridge Regression significantly outperformed the Random Forest model in terms of both validation and test errors. Specifically, the Ridge Regression model achieved a validation error of 0.1669 and a test error of 0.1522, while the Random Forest model recorded a validation error of 1.1528 and a test error of 0.6670. These results clearly demonstrate that Ridge Regression was the more effective model for this particular task, as it produced consistently lower

errors across both the validation and test sets. Consequently, Ridge Regression was selected as the final model for this project. Although Random Forest is typically favoured for capturing complex nonlinear relationships, Ridge Regression performed better on both validation and test sets in this case. The lower errors achieved by Ridge Regression indicate that it was more effective at predicting AQI in this specific dataset.

5. Conclusion

In this project, we aimed to predict the AQI for Beijing using historical pollutant and meteorological data. Two machine learning models, Random Forest Regressor and Ridge Regression, were applied to this task. After comparing the performance of both models on the training, validation and test sets, Ridge Regression was chosen as the final model due to its superior performance in terms of MSE.

The results showed that Ridge Regression achieved a validation error of 0.1669 and a test error of 0.1522, significantly outperforming the Random Forest model, which had a validation error of 1.1528 and a test error of 0.6670. While Random Forest typically excels at modelling complex nonlinear relationships, in this case, the simpler linear model—Ridge Regression—proved to be more effective, likely due to the nature of the relationships between the features and the AQI.

5.1 Limitations

Despite the success of Ridge Regression in this project, there are several limitations to consider:

- **Model Complexity**: While Ridge Regression performed well, more complex models like Random Forest may perform better with additional feature engineering or hyperparameter tuning.
- **Feature Set**: The dataset used only six features related to meteorological conditions and pollutants. Including more features, such as real-time traffic data or industrial activity, could improve the model’s predictive accuracy.
- **Data Limitations**: The dataset only spans from 2010-2014. More recent data could provide better insights and account for changes in air quality trends due to recent changes.

5.2 Future Improvements

Several avenues for improvement and future research could be pursued:

- **Feature Engineering**: Investigating additional features or performing feature transformations could help improve the model’s performance, especially for more complex models like Random Forest.
- **Model Exploration**: Other machine learning models, such as Gradient Boosting Machines or neural networks, could be tested to optimise the performance further.
- **Long-term Forecasting**: Extending the project to predict AQI over a longer time horizon or using time-series models could enhance its usefulness for public health planning.

In conclusion, while Ridge Regression provided satisfactory results in this project, there is still room for improvement. Future work should focus on expanding the feature set, experimenting with more sophisticated models, and exploring different time horizons for prediction to address air quality prediction challenges better.

6. Bibliography

- Jung, A., 2022. Machine Learning: The Basics. Singapore: Springer.
- Chen, Song. "Beijing PM2.5." UCI Machine Learning Repository, 2015, https://doi.org/10.24432/C5JS49.

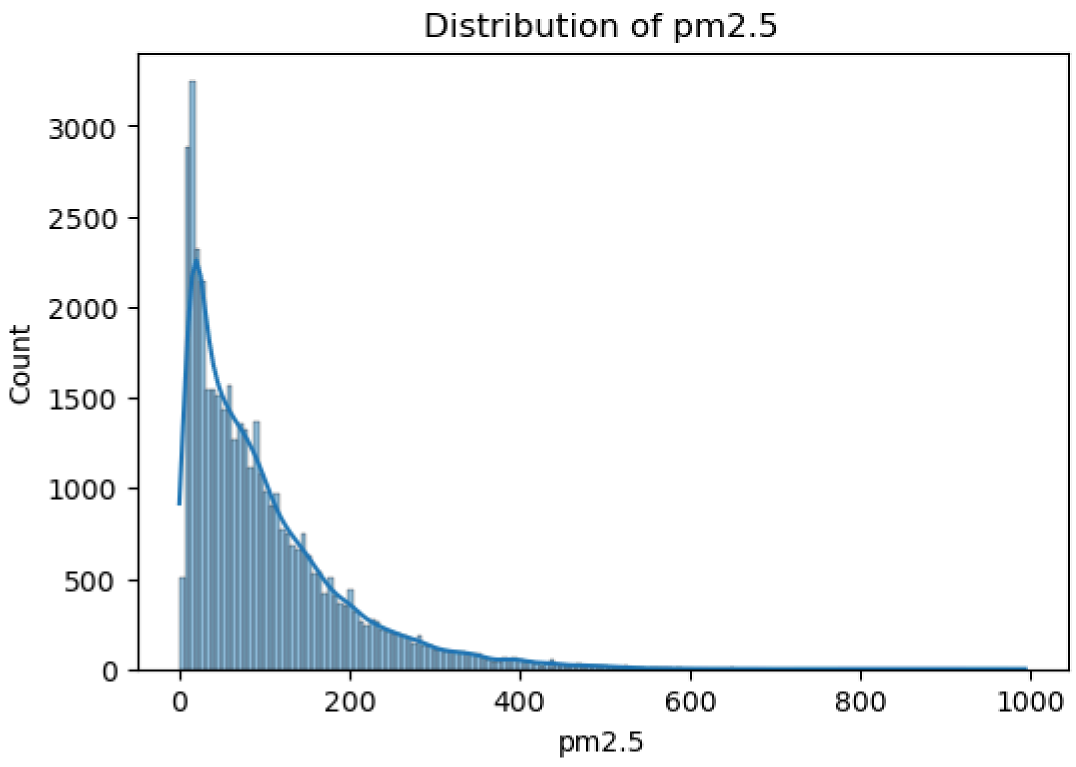
In [4]:
```
import pandas as pd
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split

# Load the dataset and drop rows with missing values
data = pd.read_csv('Beijing_PM2.5.csv')
data_clean = data.dropna()
print(data_clean.head())

# Apply min-max scaling to the selected features
scaler = MinMaxScaler()
features = ['pm2.5', 'DEWP', 'TEMP', 'PRES', 'Iws']
data_clean.loc[:, features] = scaler.fit_transform(data_clean[features])
print(data_clean[features].head())

# Split the dataset (75% training, 10% validation, 15% test)
X = data_clean[features]
y = data_clean['pm2.5']

X_train, X_temp, y_train, y_temp = train_test_split(X, y, test_size=0.25, random
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.6, r

print("Training set size:", len(X_train))
print("Validation set size:", len(X_val))
print("Test set size:", len(X_test))
```

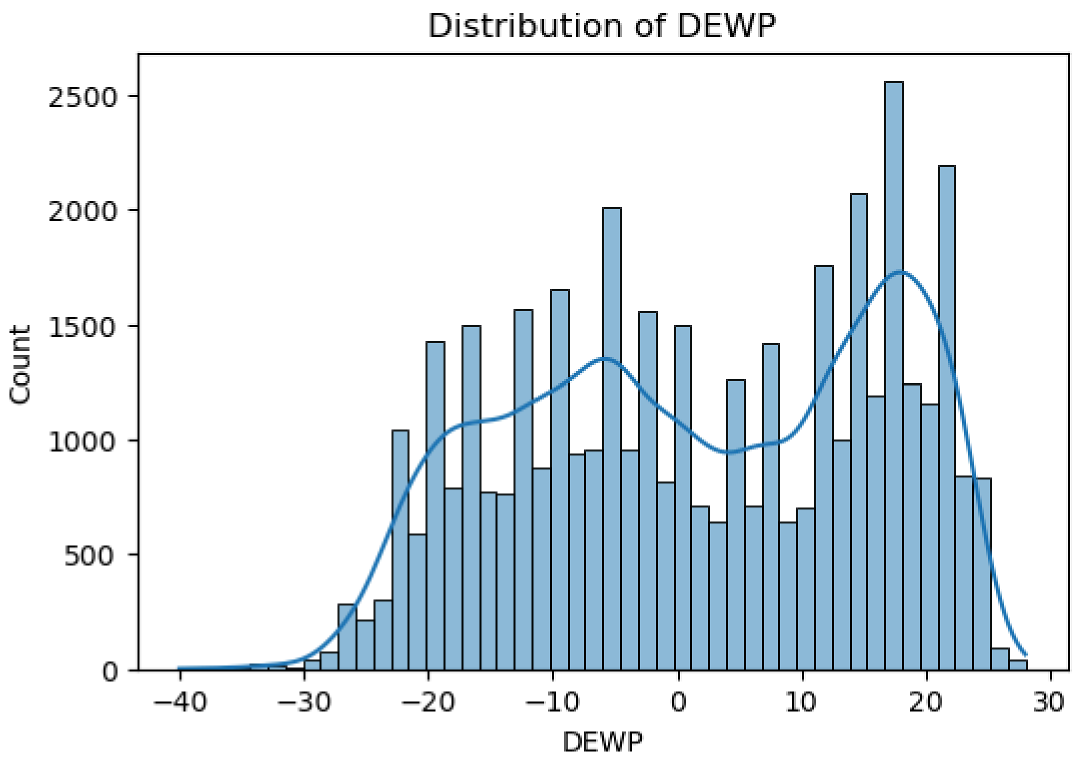
```
    No  year  month  day  hour  pm2.5  DEWP  TEMP    PRES cbwd   Iws  Is  Ir
24  25  2010      1    2     0  129.0   -16  -4.0  1020.0   SE  1.79   0   0
25  26  2010      1    2     1  148.0   -15  -4.0  1020.0   SE  2.68   0   0
26  27  2010      1    2     2  159.0   -11  -5.0  1021.0   SE  3.57   0   0
27  28  2010      1    2     3  181.0    -7  -5.0  1022.0   SE  5.36   1   0
28  29  2010      1    2     4  138.0    -7  -5.0  1022.0   SE  6.25   2   0
       pm2.5      DEWP      TEMP      PRES       Iws
24  0.129779  0.352941  0.245902  0.527273  0.002372
25  0.148893  0.367647  0.245902  0.527273  0.003947
26  0.159960  0.426471  0.229508  0.545455  0.005522
27  0.182093  0.485294  0.229508  0.563636  0.008690
28  0.138833  0.485294  0.229508  0.563636  0.010265
Training set size: 31317
Validation set size: 4176
Test set size: 6264
```

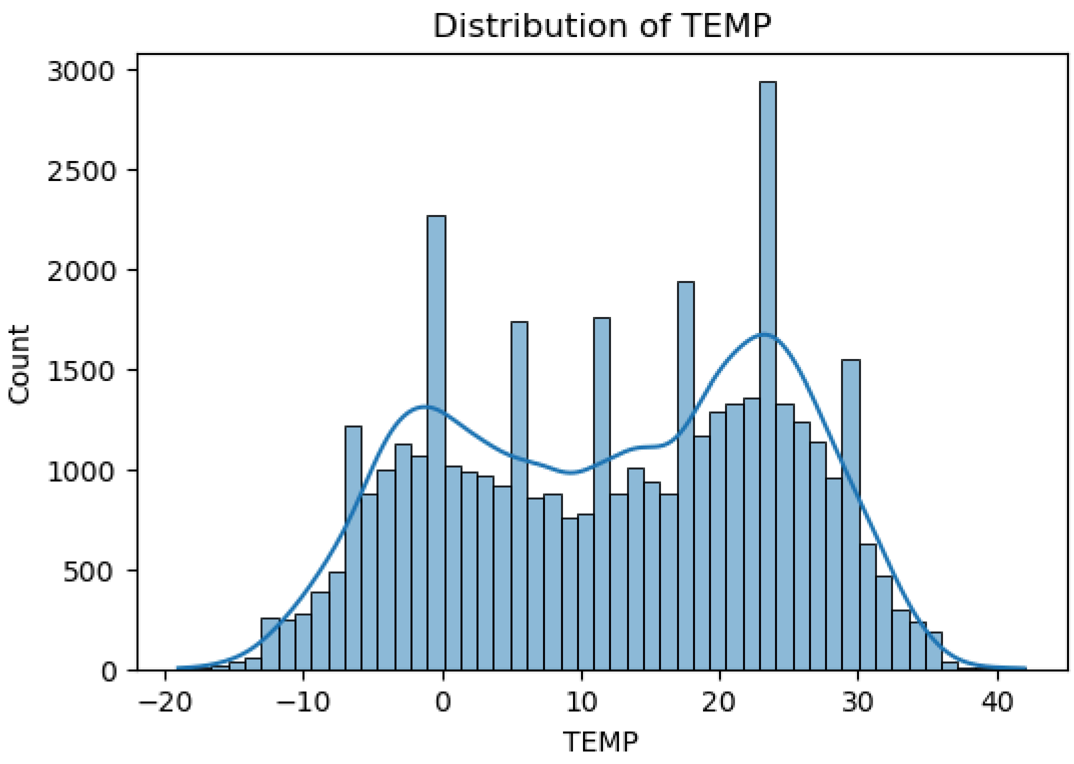
In [1]:
```
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import RandomForestRegressor
from sklearn.linear_model import Ridge
from sklearn.metrics import mean_squared_error

# Load and preprocess your dataset
data = pd.read_csv('Beijing_PM2.5.csv')
data_clean = data.dropna()

# Visualize the distribution of each feature
print("Visualizing the data distribution:")
features = ['pm2.5', 'DEWP', 'TEMP', 'PRES', 'Iws']
for feature in features:
    plt.figure(figsize=(6, 4))
    sns.histplot(data_clean[feature], kde=True)
    plt.title(f'Distribution of {feature}')
    plt.show()

# Correlation Matrix
print("Checking correlations with pm2.5:")

# Drop non-numeric columns like 'cbwd' before calculating the correlation matrix
numeric_data = data_clean.drop(columns=['cbwd'])

# Calculate and visualize the correlation matrix
correlation_matrix = numeric_data.corr()
sns.heatmap(correlation_matrix, annot=True, cmap="coolwarm", fmt=".2f")
plt.title('Correlation Matrix')
plt.show()

# Plot the correlation of the features with pm2.5
correlation_with_target = correlation_matrix['pm2.5'].sort_values(ascending=Fals
print("Correlation with pm2.5 (target):\n", correlation_with_target)

# Features and label
features = ['pm2.5', 'DEWP', 'TEMP', 'PRES', 'Iws']
X = data_clean[features]
y = data_clean['pm2.5']

# Apply Min-Max Scaling
scaler = MinMaxScaler()
X_scaled = scaler.fit_transform(X)

# Split into training, validation, and test sets
X_train, X_temp, y_train, y_temp = train_test_split(X_scaled, y, test_size=0.25,
X_val, X_test, y_val, y_test = train_test_split(X_temp, y_temp, test_size=0.6, r

# Random Forest Regressor
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)

# Validation and test predictions
rf_train_pred = rf_model.predict(X_train)
rf_val_pred = rf_model.predict(X_val)
rf_test_pred = rf_model.predict(X_test)

# Mean Squared Error (MSE)
rf_train_mse = mean_squared_error(y_train, rf_train_pred)
rf_val_mse = mean_squared_error(y_val, rf_val_pred)
rf_test_mse = mean_squared_error(y_test, rf_test_pred)

print("Random Forest Training MSE:", rf_train_mse)
print("Random Forest Validation MSE:", rf_val_mse)
print("Random Forest Test MSE:", rf_test_mse)

# Ridge Regression
ridge_model = Ridge(alpha=1.0)
ridge_model.fit(X_train, y_train)

# Validation and test predictions
ridge_train_pred = ridge_model.predict(X_train)
ridge_val_pred = ridge_model.predict(X_val)
ridge_test_pred = ridge_model.predict(X_test)
```

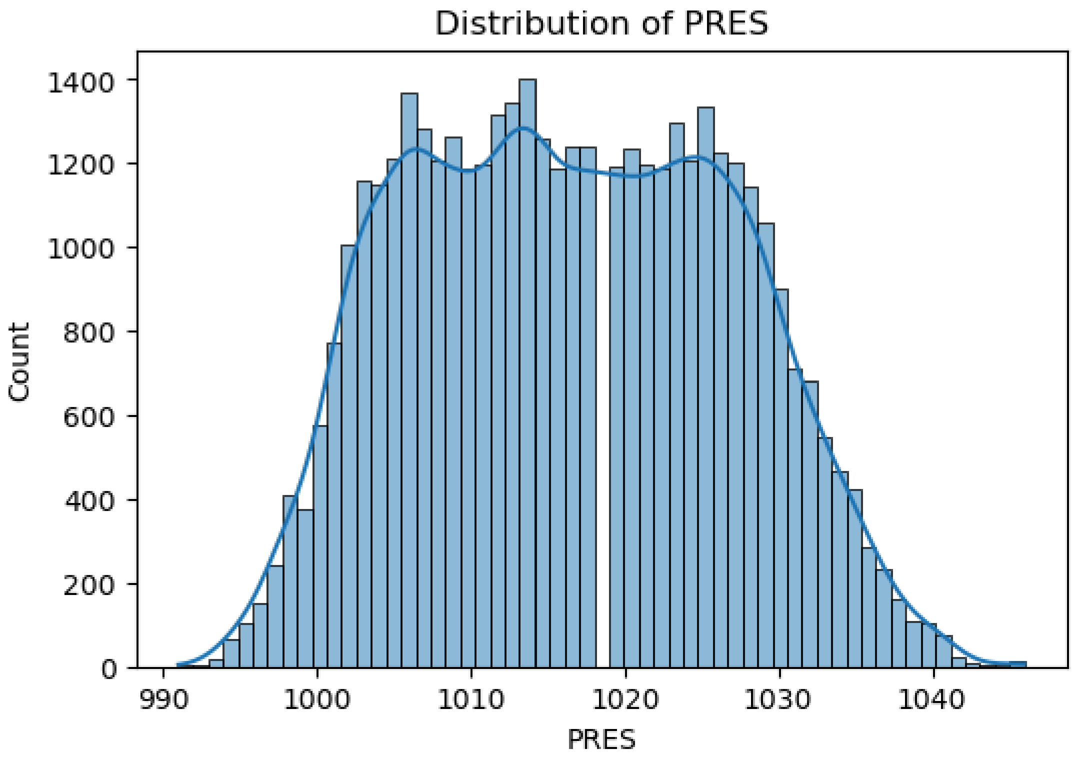
```python
# Mean Squared Error (MSE)
ridge_train_mse = mean_squared_error(y_train, ridge_train_pred)
ridge_val_mse = mean_squared_error(y_val, ridge_val_pred)
ridge_test_mse = mean_squared_error(y_test, ridge_test_pred)

print("Ridge Regression Training MSE:", ridge_train_mse)
print("Ridge Regression Validation MSE:", ridge_val_mse)
print("Ridge Regression Test MSE:", ridge_test_mse)
```

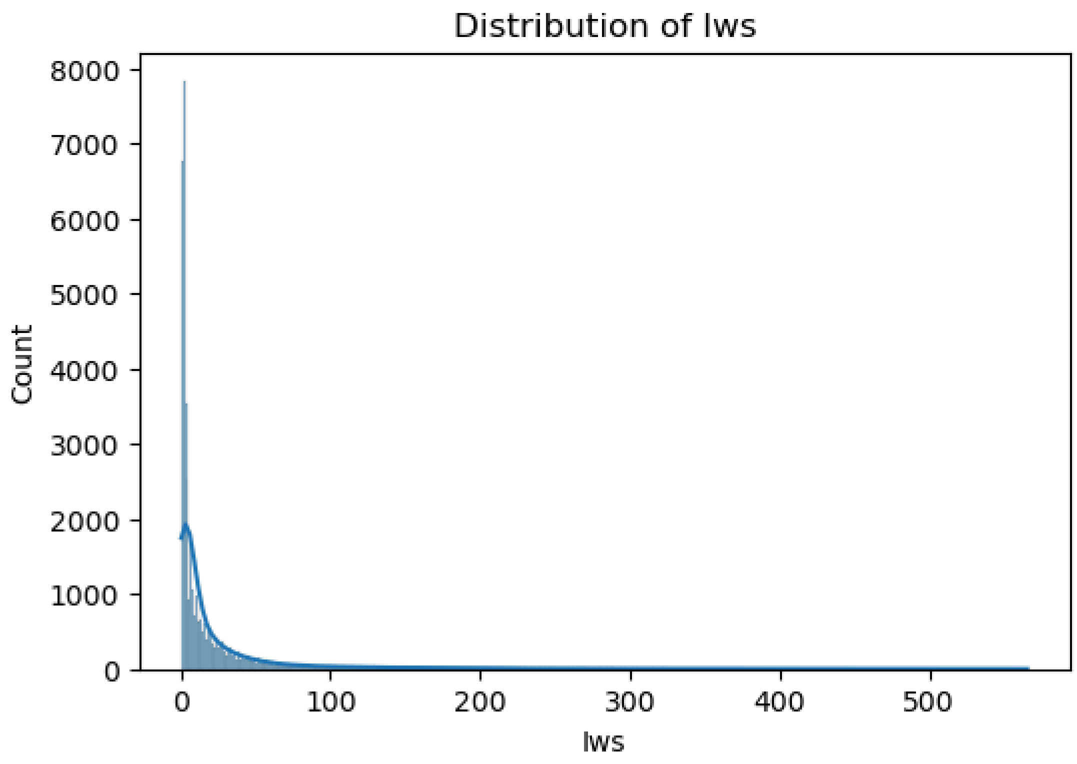
```
Visualizing the data distribution:
```



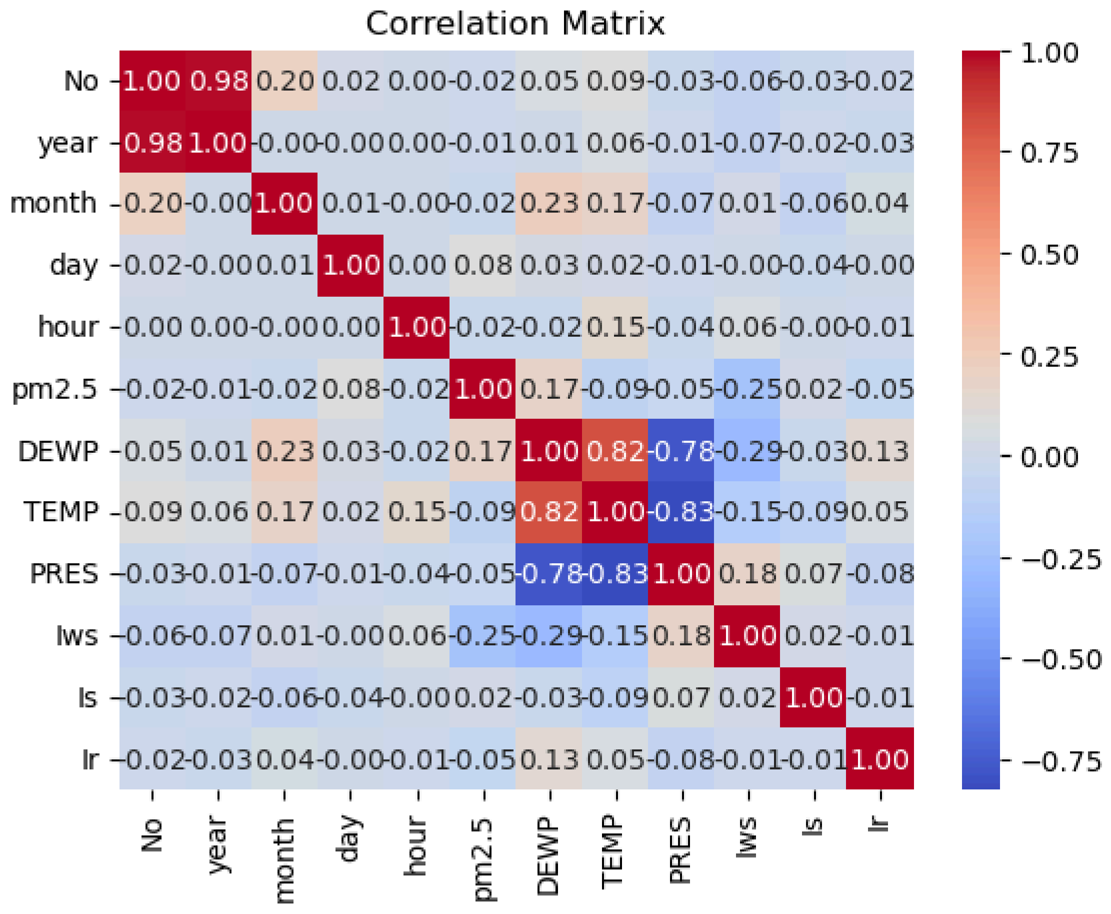



Distribution of TEMP
Count
3000
2500
2000
1500
1000
500
0
−20
−10
0
10
20
30
40
TEMP


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



```
Checking correlations with pm2.5:
```



	No	year	month	day	hour	pm2.5	DEWP	TEMP	PRES	Iws	Is	Ir
No	1.00	0.98	0.20	0.02	0.00	-0.02	0.05	0.09	-0.03	-0.06	-0.03	-0.02
year	0.98	1.00	-0.00	-0.00	0.00	-0.01	0.01	0.06	-0.01	-0.07	-0.02	-0.03
month	0.20	-0.00	1.00	0.01	-0.00	-0.02	0.23	0.17	-0.07	0.01	-0.06	0.04
day	0.02	-0.00	0.01	1.00	0.00	0.08	0.03	0.02	-0.01	-0.00	-0.04	-0.00
hour	0.00	0.00	-0.00	0.00	1.00	-0.02	-0.02	0.15	-0.04	0.06	-0.00	-0.01
pm2.5	-0.02	-0.01	-0.02	0.08	-0.02	1.00	0.17	-0.09	-0.05	-0.25	0.02	-0.05
DEWP	0.05	0.01	0.23	0.03	-0.02	0.17	1.00	0.82	-0.78	-0.29	-0.03	0.13
TEMP	0.09	0.06	0.17	0.02	0.15	-0.09	0.82	1.00	-0.83	-0.15	-0.09	0.05
PRES	-0.03	-0.01	-0.07	-0.01	-0.04	-0.05	-0.78	-0.83	1.00	0.18	0.07	-0.08
Iws	-0.06	-0.07	0.01	-0.00	0.06	-0.25	-0.29	-0.15	0.18	1.00	0.02	-0.01
Is	-0.03	-0.02	-0.06	-0.04	-0.00	0.02	-0.03	-0.09	0.07	0.02	1.00	-0.01
Ir	-0.02	-0.03	0.04	-0.00	-0.01	-0.05	0.13	0.05	-0.08	-0.01	-0.01	1.00

```
Correlation with pm2.5 (target):
 pm2.5    1.000000
DEWP     0.171423
day      0.082788
Is       0.019266
year    -0.014690
No      -0.017706
hour    -0.023116
month   -0.024069
PRES    -0.047282
Ir      -0.051369
TEMP    -0.090534
Iws     -0.247784
Name: pm2.5, dtype: float64
Random Forest Training MSE: 0.04693733116198871
Random Forest Validation MSE: 1.1527902777777768
Random Forest Test MSE: 0.6669782407407417
Ridge Regression Training MSE: 0.15124994408256184
Ridge Regression Validation MSE: 0.16694970912244458
Ridge Regression Test MSE: 0.15221073835187093
```

```
In [ ]:
```