XGBoost Model for Daily Predictions

Importing Libraries

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import xgboost as xgb
from sklearn.metrics import mean_squared_error
from sklearn.metrics import mean_absolute_error
from sklearn.metrics import r2_score
```

Importing Dataset

```
dataset = pd.read_csv('final-dataset.csv',
                      parse_dates=['Time'],
                      index_col='Time')
dataset.sort_index(inplace=True)
```

Preprocessing Dataset

```
# Resample data to daily frequency
daily_data = dataset.resample('D').mean()
daily_data['PM2.5'] = daily_data['PM2.5']
```

```
# lag features
def create_lag_features(data, lags):
    for lag in range(1, lags + 1):
        data[f'lag_{lag}'] = data['PM2.5'].shift(lag)
    return data
lags = 15
daily_data = create_lag_features(daily_data, lags)
daily_data.dropna(inplace=True)
```

Train Test Split

```
# Separate features (X) and target variable (y)
X = daily_data.drop(columns=['PM2.5'])
y = daily_data['PM2.5']
train_size = int(len(X) * 0.80)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

Building XGBoost Model for Daily Predictions

```
# XGBoost model parameters
params = {
    'objective': 'reg:squarederror',  # Regression
    'eval_metric': 'rmse',            # Evaluation metric
    'max_depth': 3 ,                  # Maximum Depth
    'n_estimators': 100,              # Trees
}
num_rounds = 100
# DMatrix format for XGBoost
dtrain = xgb.DMatrix(X_train, label=y_train)
dtest = xgb.DMatrix(X_test, label=y_test)
```

Training Model

```
# Train the XGBoost model
xgb_model = xgb.train(params, dtrain, num_rounds)
```

Show hidden output

Making Predictions

```
# Make predictions on the test set
y_pred = xgb_model.predict(dtest)
```

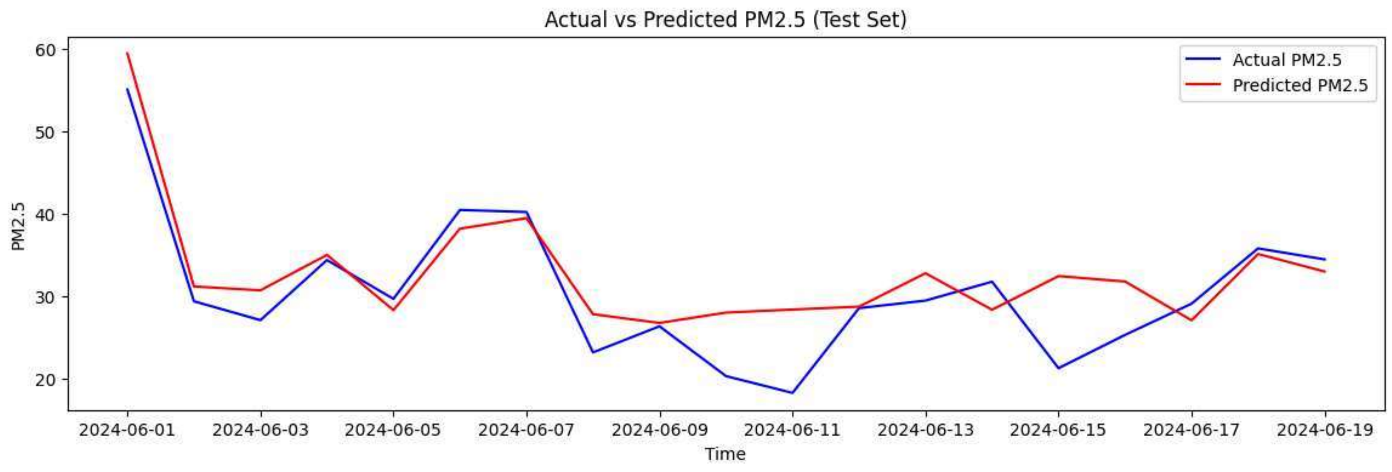
Evaluating Model Performance

```
# Evaluate the model
mse = mean_squared_error(y_test, y_pred)
print("Mean Squared Error (MSE):", mse)
rmse = np.sqrt(mse)
print("Root Mean Squared Error (RMSE):", rmse)
mae = mean_absolute_error(y_test, y_pred)
print("Mean Absolute Error (MAE):", mae)
r2 = r2_score(y_test, y_pred)
print("R-squared:", r2)
```

```
Mean Squared Error (MSE): 22.275255000816824
Root Mean Squared Error (RMSE): 4.719666831548263
Mean Absolute Error (MAE): 3.495050196062055
R-squared: 0.6785582283172543
```

Visual Plot for Actual and Predicted Data Points

```
  plt.figure(figsize=(14, 4))
  plt.plot(y_test.index, y_test, color='blue', label='Actual PM2.5')
  plt.plot(y_test.index, y_pred, color='red', label='Predicted PM2.5')
  plt.title('Actual vs Predicted PM2.5 (Test Set)')
  plt.xlabel('Time')
  plt.ylabel('PM2.5')
  plt.legend()
  plt.show()
```


Actual vs Predicted PM2.5 (Test Set)
Actual PM2.5
Predicted PM2.5
PM2.5
Time