

## ✓ XGBoost Model for Hourly 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

```
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 = 5
dataset = create_lag_features(dataset, lags)
dataset.dropna(inplace=True)
```

### ✓ Train Test Split

```
# Separate features (X) and target variable (y)
X = dataset.drop(columns=['PM2.5'])
y = dataset['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 Hourly Predictions

```
# XGBoost model parameters
params = {
    'objective': 'reg:squarederror', # Regression
    'eval_metric': 'rmse',           # Evaluation metric
    'max_depth': 3,                  # Maximum Depth
    'n_estimators': 100,             # Trees
}
# 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
num_rounds = 100
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

```
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): 3.9615976292360826  
Root Mean Squared Error (RMSE): 1.9903762531833227  
Mean Absolute Error (MAE): 1.3808883666992187  
R-squared: 0.9825394415533497

## ▼ Visual Plot for Actual and Predicted Data Points

```
n_points = len(y_test)
n_points_20_percent = int(n_points * 0.2)
# Select only the last 20% of the data
x_indices = y_test.index[-n_points_20_percent:]
y_test_actual_20_percent = y_test[-n_points_20_percent:]
y_pred_actual_20_percent = y_pred[-n_points_20_percent:]
plt.figure(figsize=(14, 4))
plt.plot(x_indices, y_test_actual_20_percent, color='blue', label='Actual PM2.5')
plt.plot(x_indices, y_pred_actual_20_percent, color='red', label='Predicted PM2.5')
plt.title('Actual vs Predicted PM2.5 (Test Set - Last 20%)')
plt.xlabel('Time')
plt.ylabel('PM2.5')
plt.legend()
plt.show()
```

