```python
import numpy as np
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_squared_error, r2_score

# Step 1: Sample data (replace this with your real air quality data)
# Features: PM10, SO2, NO2, temperature, humidity, wind_speed
X = np.array([
    [50, 10, 20, 30, 60, 3.5],
    [60, 12, 25, 28, 55, 4.0],
    [70, 15, 18, 32, 65, 3.2],
    [80, 20, 22, 29, 70, 4.5],
    [90, 25, 30, 26, 75, 5.0],
    [100, 30, 35, 25, 80, 5.5],
    [110, 35, 28, 33, 85, 6.0],
    [120, 40, 40, 27, 90, 6.5]
])

# Target: PM2.5 (air quality level)
y = np.array([25, 30, 28, 35, 40, 45, 50, 55])

# Step 2: Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.3, random_state=42)

# Step 4: Train the Random Forest model
model = RandomForestRegressor(n_estimators=100, random_state=42)
model.fit(X_train, y_train)

# Step 5: Make predictions
y_pred = model.predict(X_test)

# Step 6: Evaluate the model
mse = mean_squared_error(y_test, y_pred)
r2 = r2_score(y_test, y_pred)

# Step 7: Print the results
print(f"Mean Squared Error: {mse}")
print(f"R² Score: {r2}")
```

```
# Step 8: Predict PM2.5 for a new sample input (example)
sample_input = np.array([[85, 22, 27, 30, 78, 4.8]])  # Example new data
sample_scaled = scaler.transform(sample_input)  # Scale the sample input
predicted_pm25 = model.predict(sample_scaled)
print(f"Predicted PM2.5 for the sample input: {predicted_pm25[0]}")

# Step 9: Plot Actual vs Predicted PM2.5 values
plt.figure(figsize=(8, 6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--', label='Perfect Prediction')
plt.xlabel('Actual PM2.5')
plt.ylabel('Predicted PM2.5')
plt.title('Actual vs Predicted PM2.5')
plt.legend()
plt.show()

# Step 10: Plot Feature Importance
feature_importance = model.feature_importances_
plt.figure(figsize=(8, 6))
plt.barh(range(len(feature_importance)), feature_importance,
tick_label=['PM10', 'SO2', 'NO2', 'Temperature', 'Humidity', 'Wind
Speed'])
plt.xlabel('Feature Importance')
plt.title('Feature Importance for PM2.5 Prediction')
plt.show()
```

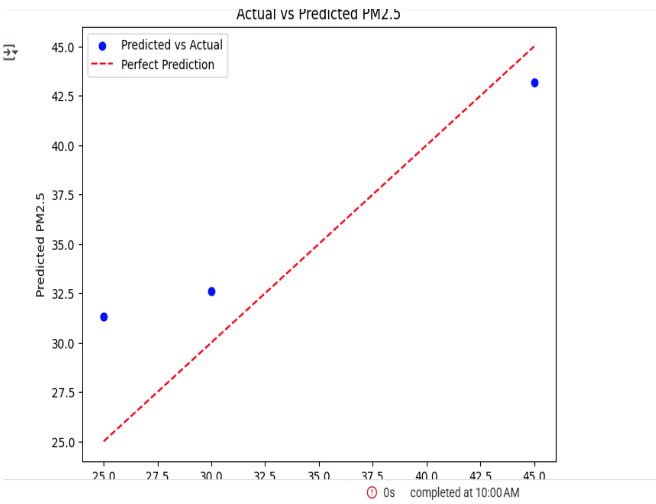
Second program :

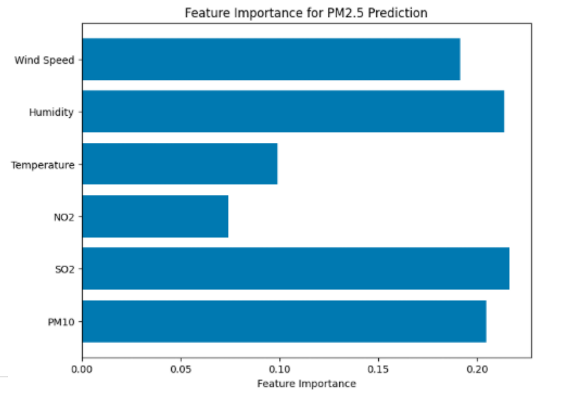
```
import numpy as np
import matplotlib.pyplot as plt
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from tensorflow.keras.utils import plot_model

# Step 1: Create synthetic data (features: PM10, SO2, NO2, temperature,
humidity, wind_speed)
```

```
X = np.array([
    [50, 10, 20, 30, 60, 3.5],
    [60, 12, 25, 28, 55, 4.0],
    [70, 15, 18, 32, 65, 3.2],
    [80, 20, 22, 29, 70, 4.5],
    [90, 25, 30, 26, 75, 5.0],
    [100, 30, 35, 25, 80, 5.5],
    [110, 35, 28, 33, 85, 6.0],
    [120, 40, 40, 27, 90, 6.5]
])

# Target: PM2.5 (air quality level)
y = np.array([25, 30, 28, 35, 40, 45, 50, 55])

# Step 2: Scale the features
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)

# Step 3: Split data into train and test (no validation for simplicity
here)
X_train, X_test, y_train, y_test = train_test_split(X_scaled, y,
test_size=0.3, random_state=42)

# Step 4: Reshape input for LSTM (LSTM expects 3D data)
X_train_reshaped = X_train.reshape((X_train.shape[0], 1,
X_train.shape[1]))  # (samples, time_steps, features)
X_test_reshaped = X_test.reshape((X_test.shape[0], 1, X_test.shape[1]))  #
(samples, time_steps, features)

# Step 5: Build the LSTM model
model_lstm = Sequential([
    LSTM(units=64, activation='relu', input_shape=(1, X_train.shape[1])),
    Dense(units=1)
])

# Step 6: Compile the model
model_lstm.compile(optimizer='adam', loss='mse')

# Step 7: Model summary
model_lstm.summary()

# Step 8: Train the model
history_lstm = model_lstm.fit(X_train_reshaped, y_train, epochs=50,
batch_size=32, validation_data=(X_test_reshaped, y_test))
```

```python
# Step 9: Evaluate the model
loss = model_lstm.evaluate(X_test_reshaped, y_test)
print(f"Test Loss (MSE): {loss}")

# Step 10: Make predictions
y_pred = model_lstm.predict(X_test_reshaped)
print("Predicted PM2.5 values:", y_pred)

# Step 11: Plot the training & validation loss over epochs
plt.figure(figsize=(8,6))
plt.plot(history_lstm.history['loss'], label='Training Loss')
plt.plot(history_lstm.history['val_loss'], label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Model Training & Validation Loss')
plt.legend()
plt.show()

# Step 12: Plot Actual vs Predicted PM2.5 values
plt.figure(figsize=(8,6))
plt.scatter(y_test, y_pred, color='blue', label='Predicted vs Actual')
plt.plot([min(y_test), max(y_test)], [min(y_test), max(y_test)],
color='red', linestyle='--', label='Perfect Prediction')
plt.xlabel('Actual PM2.5')
plt.ylabel('Predicted PM2.5')
plt.title('Actual vs Predicted PM2.5')
plt.legend()
plt.show()

# Step 13: Plot the Model Architecture (graph)
plot_model(model_lstm, to_file='model_lstm.png', show_shapes=True,
show_layer_names=True)
```

Output for first program:





Output for |Second program:

Layer (type)	Output Shape	Param #
lstm_1 (LSTM)	(None, 64)	18,176
dense_1 (Dense)	(None, 1)	65

```
 Total params: 18,241 (71.25 KB)
 Trainable params: 18,241 (71.25 KB)
 Non-trainable params: 0 (0.00 B)
Epoch 1/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 3s 3s/step - loss: 1822.7426 - val_loss: 1181.3257
Epoch 2/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 136ms/step - loss: 1821.9955 - val_loss: 1180.8495
Epoch 3/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - loss: 1821.2471 - val_loss: 1180.3724
Epoch 4/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - loss: 1820.4965 - val_loss: 1179.9008
Epoch 5/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 152ms/step - loss: 1819.7516 - val_loss: 1179.4301
Epoch 6/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 189ms/step - loss: 1819.0045 - val_loss: 1178.9564
Epoch 7/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 182ms/step - loss: 1818.2562 - val_loss: 1178.4833
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 303ms/step - loss: 1814.4500 - val_loss: 1176.0614
Epoch 13/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 297ms/step - loss: 1813.6790 - val_loss: 1175.5668
Epoch 14/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 176ms/step - loss: 1812.9016 - val_loss: 1175.0688
Epoch 15/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 153ms/step - loss: 1812.1097 - val_loss: 1174.5675
Epoch 16/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 107ms/step - loss: 1811.3093 - val_loss: 1174.0623
Epoch 17/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - loss: 1810.5088 - val_loss: 1173.5492
Epoch 18/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 97ms/step - loss: 1809.6879 - val_loss: 1173.0270
Epoch 19/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 1808.8539 - val_loss: 1172.4960
Epoch 20/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 1808.0078 - val_loss: 1171.9524
Epoch 21/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 137ms/step - loss: 1807.1472 - val_loss: 1171.4015
Epoch 22/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 1806.2705 - val_loss: 1170.8444
Epoch 23/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step - loss: 1805.3757 - val_loss: 1170.2806
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 143ms/step - loss: 1791.9270 - val_loss: 1161.8525
Epoch 37/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 105ms/step - loss: 1790.7279 - val_loss: 1161.1222
Epoch 38/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - loss: 1789.5125 - val_loss: 1160.3705
Epoch 39/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - loss: 1788.2565 - val_loss: 1159.6039
Epoch 40/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 113ms/step - loss: 1786.9740 - val_loss: 1158.8259
Epoch 41/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 100ms/step - loss: 1785.6692 - val_loss: 1158.0297
Epoch 42/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 96ms/step - loss: 1784.3369 - val_loss: 1157.2161
Epoch 43/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 99ms/step - loss: 1782.9697 - val_loss: 1156.3893
Epoch 44/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 142ms/step - loss: 1781.5707 - val_loss: 1155.5483
Epoch 45/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 177ms/step - loss: 1780.1393 - val_loss: 1154.6903
Epoch 46/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 103ms/step - loss: 1778.6621 - val_loss: 1153.8087
Epoch 47/50
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 141ms/step - loss: 1777.1481 - val_loss: 1152.9097
```

```
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 45ms/step - loss: 1150.0952
Test Loss (MSE): 1150.09521484375
1/1 ━━━━━━━━━━━━━━━━━━━━ 0s 234ms/step
Predicted PM2.5 values: [[0.37071055]
 [0.5649445 ]
 [0.54766464]]
```