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

# Generate synthetic air quality data
def generate_sample_data(n_samples=1000):
    np.random.seed(42)
    data = {
        'Temperature': np.random.uniform(10, 40, n_samples),
        'Humidity': np.random.uniform(20, 90, n_samples),
        'PM2.5': np.random.uniform(5, 150, n_samples),
        'PM10': np.random.uniform(10, 200, n_samples),
        'NO2': np.random.uniform(5, 100, n_samples),
        'SO2': np.random.uniform(2, 80, n_samples),
        'CO': np.random.uniform(0.1, 10, n_samples),
        'O3': np.random.uniform(10, 100, n_samples),
    }

    df = pd.DataFrame(data)
    # Create a synthetic AQI (target) using a weighted sum + noise
    df['AQI'] = (
        0.4 * df['PM2.5'] +
        0.2 * df['PM10'] +
        0.1 * df['NO2'] +
        0.1 * df['SO2'] +
        0.1 * df['CO'] +
        0.1 * df['O3'] +
        np.random.normal(0, 10, n_samples)
    )
    return df

# Preprocess data
def preprocess_data(df):
    X = df.drop(columns=['AQI'])
    y = df['AQI']
    scaler = StandardScaler()
    X_scaled = scaler.fit_transform(X)
    return X_scaled, y, X.columns

# Train and evaluate model
def train_and_evaluate(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    rmse = mean_squared_error(y_test, y_pred, squared=False)
    r2 = r2_score(y_test, y_pred)
    print(f"📊 RMSE: {rmse:.2f}")
    print(f"📈 R² Score: {r2:.2f}")

    return model

# Plot feature importance
def plot_feature_importance(model, feature_names):
    importances = model.feature_importances_
    indices = np.argsort(importances)[::-1]
    plt.figure(figsize=(10, 6))
    sns.barplot(x=importances[indices], y=np.array(feature_names)[indices])
    plt.title("Feature Importance")
    plt.xlabel("Importance")
    plt.ylabel("Features")
    plt.tight_layout()
    plt.show()

# Main execution
def main():
    print("🔄 Generating sample air quality data...")
    df = generate_sample_data()
    print("✅ Data generated.\n", df.head())
```

```
    X, y, feature_names = preprocess_data(df)
    print("🚀 Training model...")
    model = train_and_evaluate(X, y)

    print("📊 Plotting feature importance...")
    plot_feature_importance(model, feature_names)

if __name__ == "__main__":
    main()
```

```
🔄 Generating sample air quality data...
✅ Data generated.
   Temperature   Humidity       PM2.5        PM10        NO2        SO2  \
0    21.236204  32.959305   42.947324  137.813569  59.339608  32.703571
1    38.521429  57.933066   40.811926  161.369465  81.516071  38.927981
2    31.959818  81.106209  136.406914   57.588901  77.215288  68.654697
3    27.959755  71.255742   41.184199  128.726079  19.620491  28.520342
4    14.680559  76.459280   44.432710  118.631737  19.178700  69.832675

         CO         O3         AQI
0  6.517744  13.491951  65.535094
1  1.806625  26.809528  74.781547
2  8.736706  84.812122  70.549002
3  6.169851  79.009152  65.335457
4  1.656318  41.557842  46.616699
🚀 Training model...
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-2-e0680576c01d> in <cell line: 0>()
     83
     84 if __name__ == "__main__":
---> 85     main()

 4 frames
/usr/lib/python3.11/inspect.py in _bind(self, args, kwargs, partial)
   3182                 arguments[kwargs_param.name] = kwargs
   3183             else:
-> 3184                 raise TypeError(
   3185                     'got an unexpected keyword argument {arg!r}'.format(
   3186                         arg=next(iter(kwargs))))

TypeError: got an unexpected keyword argument 'squared'
```

Next steps: Explain error

```
# Train and evaluate model
def train_and_evaluate(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
    model = RandomForestRegressor(n_estimators=100, random_state=42)
    model.fit(X_train, y_train)
    y_pred = model.predict(X_test)

    # Calculate MSE first
    mse = mean_squared_error(y_test, y_pred)
    # Calculate RMSE by taking the square root of MSE
    rmse = np.sqrt(mse)

    r2 = r2_score(y_test, y_pred)
    print(f"📊 RMSE: {rmse:.2f}")
    print(f"📈 R² Score: {r2:.2f}")

    return model

# Plot feature importance
def plot_feature_importance(model, feature_names):
    importances = model.feature_importances_
    indices = np.argsort(importances)[::-1]
    plt.figure(figsize=(10, 6))
    sns.barplot(x=importances[indices], y=np.array(feature_names)[indices])
    plt.title("Feature Importance")
    plt.xlabel("Importance")
    plt.ylabel("Features")
    plt.tight_layout()
    plt.show()

# Main execution
def main():
```

```
    print("🔄 Generating sample air quality data...")
    df = generate_sample_data()
    print("✅ Data generated.\n", df.head())

    X, y, feature_names = preprocess_data(df)
    print("🚀 Training model...")
    model = train_and_evaluate(X, y)

    print("📊 Plotting feature importance...")
    plot_feature_importance(model, feature_names)

if __name__ == "__main__":
    main()
```

```
🔄 Generating sample air quality data...
✅ Data generated.
   Temperature   Humidity       PM2.5        PM10        NO2        SO2  \
0    21.236204  32.959305   42.947324  137.813569  59.339608  32.703571
1    38.521429  57.933066   40.811926  161.369465  81.516071  38.927981
2    31.959818  81.106209  136.406914   57.588901  77.215288  68.654697
3    27.959755  71.255742   41.184199  128.726079  19.620491  28.520342
4    14.680559  76.459280   44.432710  118.631737  19.178700  69.832675

         CO         O3        AQI
0  6.517744  13.491951  65.535094
1  1.806625  26.809528  74.781547
2  8.736706  84.812122  70.549002
3  6.169851  79.009152  65.335457
4  1.656318  41.557842  46.616699
🚀 Training model...
📊 RMSE: 12.63
📈 R² Score: 0.71
📊 Plotting feature importance...
```