```
import numpy as np
import pandas as pd
```

```
dt=pd.read_csv("/content/drive/MyDrive/seattle-weather.csv")
dt.columns
```

```
Index(['date', 'precipitation', 'temp_max', 'temp_min', 'wind', 'weather'], dtype='object')
```

```
dt.head()
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain

```
dt['weather'].value_counts()
```

```
rain       641
sun        640
fog        101
drizzle     53
snow        26
Name: weather, dtype: int64
```

```
print(dt.dtypes)
```

```
date              object
precipitation    float64
temp_max         float64
temp_min         float64
wind             float64
weather           object
dtype: object
```

```
dt['date'] = pd.to_datetime(dt['date'])
dt['day_of_week'] = dt['date'].dt.dayofweek
dt['month'] = dt['date'].dt.month_name()
dt['year'] = dt['date'].dt.year.astype(int)
dt.drop('date', axis=1, inplace=True)
```

```
dt
```

	precipitation	temp_max	temp_min	wind	weather	day_of_week	month	year
0	0.0	12.8	5.0	4.7	drizzle	6	January	2012
1	10.9	10.6	2.8	4.5	rain	0	January	2012
2	0.8	11.7	7.2	2.3	rain	1	January	2012
3	20.3	12.2	5.6	4.7	rain	2	January	2012
4	1.3	8.9	2.8	6.1	rain	3	January	2012
...	...	...	...	...	...	...	...	...
1456	8.6	4.4	1.7	2.9	rain	6	December	2015
1457	1.5	5.0	1.7	1.3	rain	0	December	2015
1458	0.0	7.2	0.6	2.6	fog	1	December	2015
1459	0.0	5.6	-1.0	3.4	sun	2	December	2015
1460	0.0	5.6	-2.1	3.5	sun	3	December	2015

1461 rows × 8 columns

```
print(dt.dtypes)
```

```
precipitation    float64
temp_max         float64
temp_min         float64
wind             float64
weather           object
day_of_week        int64
month             object
year               int64
dtype: object
```

```
import pandas as pd
from sklearn.utils import resample

classes_to_oversample = {'drizzle': 641, 'snow': 641, 'fog': 641}

oversampled_data = pd.DataFrame()
for cls, desired_samples in classes_to_oversample.items():
    class_data = dt[dt['weather'] == cls]
    if len(class_data) < desired_samples:
        oversampled_class = resample(class_data, replace=True, n_samples=desired_samples - len(class_data))
    else:
        oversampled_class = class_data.sample(n=desired_samples, replace=False)
    oversampled_data = pd.concat([oversampled_data, oversampled_class])

dt = pd.concat([dt, oversampled_data])
print(dt['weather'].value_counts())
```

```
drizzle    641
rain       641
snow       641
fog        641
sun        640
Name: weather, dtype: int64
```

```
dt
```

	precipitation	temp_max	temp_min	wind	weather	day_of_week	month	year
0	0.0	12.8	5.0	4.7	drizzle	6	January	2012
1	10.9	10.6	2.8	4.5	rain	0	January	2012
2	0.8	11.7	7.2	2.3	rain	1	January	2012
3	20.3	12.2	5.6	4.7	rain	2	January	2012
4	1.3	8.9	2.8	6.1	rain	3	January	2012
...	...	...	...	...	...	...	...	...
1103	0.0	7.8	1.7	2.6	fog	3	January	2015
470	0.0	13.9	4.4	2.4	fog	0	April	2013
1011	0.0	20.6	12.8	1.8	fog	2	October	2014
593	0.0	28.9	16.1	2.2	fog	4	August	2013
433	0.0	12.8	1.1	1.3	fog	5	March	2013

3204 rows × 8 columns

```
dt = dt.sort_index()
print(dt)
```

```
      precipitation  temp_max  temp_min  wind  weather  day_of_week     month  \
0               0.0      12.8       5.0   4.7  drizzle            6   January
0               0.0      12.8       5.0   4.7  drizzle            6   January
0               0.0      12.8       5.0   4.7  drizzle            6   January
0               0.0      12.8       5.0   4.7  drizzle            6   January
0               0.0      12.8       5.0   4.7  drizzle            6   January
...             ...       ...       ...   ...      ...          ...       ...
1456            8.6       4.4       1.7   2.9     rain            6  December
1457            1.5       5.0       1.7   1.3     rain            0  December
1458            0.0       7.2       0.6   2.6      fog            1  December
1459            0.0       5.6      -1.0   3.4      sun            2  December
1460            0.0       5.6      -2.1   3.5      sun            3  December

      year
```

```
0     2012
0     2012
0     2012
0     2012
0     2012
...    ...
1456  2015
1457  2015
1458  2015
1459  2015
1460  2015

[3204 rows x 8 columns]
```

```
dt.isnull().sum()
```

```
precipitation    0
temp_max         0
temp_min         0
wind             0
weather          0
day_of_week      0
month            0
year             0
dtype: int64
```

```
dt['weather'].value_counts()
```

```
drizzle    641
rain       641
snow       641
fog        641
sun        640
Name: weather, dtype: int64
```

```
from sklearn.preprocessing import LabelEncoder
label_encoder = LabelEncoder()
dt['day_of_week'] = label_encoder.fit_transform(dt['day_of_week'])
dt['month'] = label_encoder.fit_transform(dt['month'])
dt['weather'] = label_encoder.fit_transform(dt['weather'])
print(dt.head())
```

```
   precipitation  temp_max  temp_min  wind  weather  day_of_week  month  year
0            0.0      12.8       5.0   4.7        0            6      4  2012
0            0.0      12.8       5.0   4.7        0            6      4  2012
0            0.0      12.8       5.0   4.7        0            6      4  2012
0            0.0      12.8       5.0   4.7        0            6      4  2012
0            0.0      12.8       5.0   4.7        0            6      4  2012
```

```
print(dt.dtypes)
```

```
precipitation    float64
temp_max         float64
temp_min         float64
wind             float64
weather            int64
day_of_week        int64
month              int64
year               int64
dtype: object
```

```
dt
```

	precipitation	temp_max	temp_min	wind	weather	day_of_week	month	year
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
...	...	...	...	...	...	...	...	...
1456	8.6	4.4	1.7	2.9	2	6	2	2015
1457	1.5	5.0	1.7	1.3	2	0	2	2015
1458	0.0	7.2	0.6	2.6	1	1	2	2015
1459	0.0	5.6	-1.0	3.4	4	2	2	2015
1460	0.0	5.6	-2.1	3.5	4	3	2	2015

3204 rows × 8 columns

```
# Decision Tree Classifier
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.metrics import accuracy_score,confusion_matrix
X = dt.drop('weather', axis=1)
y = dt['weather']
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
clf = DecisionTreeClassifier()
clf.fit(X_train, y_train)
y_pred = clf.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9469578783151326
Confusion Matrix:
[[124   0   0   0   0]
 [  0 123   0   0   0]
 [  2   1 114   5   7]
 [  0   0   0 142   0]
 [  8   7   4   0 104]]
```

```
#Random Forest Classfier
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import classification_report
X = dt.drop('weather', axis=1)
y = dt['weather']
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
rf_classifier = RandomForestClassifier()
rf_classifier.fit(X_train, y_train)
y_pred = rf_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9516380655226209
Confusion Matrix:
[[124   0   0   0   0]
 [  0 123   0   0   0]
 [  1   0 112   3  13]
 [  0   0   0 142   0]
 [  9   5   0   0 109]]
```

```
#KNN
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.metrics import classification_report, accuracy_score, confusion_matrix
import pandas as pd

X = dt.drop('weather', axis=1)
y = dt['weather']
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)
knn_classifier = KNeighborsClassifier()
knn_classifier.fit(X_train, y_train)
y_pred = knn_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.8377535101404057
Confusion Matrix:
[[124   0   0   0   0]
 [  0 120   0   0   3]
 [  8  14  83   9  15]
 [  0   0   0 142   0]
 [ 18  33   3   1  68]]
```

```
# XGB classifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, confusion_matrix
import xgboost as xgb
X = dt.drop('weather', axis=1)
y = dt['weather']
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
X = pd.get_dummies(X)
X_train, X_test, y_train, y_test = train_test_split(X, y_encoded, test_size=0.2, random_state=42)
xgb_classifier = xgb.XGBClassifier()
xgb_classifier.fit(X_train, y_train)
y_pred = xgb_classifier.predict(X_test)
accuracy = accuracy_score(y_test, y_pred)
print(f"Accuracy: {accuracy}")
conf_matrix = confusion_matrix(y_test, y_pred)
print("Confusion Matrix:")
print(conf_matrix)
```

```
Accuracy: 0.9375975039001561
Confusion Matrix:
[[124   0   0   0   0]
 [  0 123   0   0   0]
 [  2   0 112   5  10]
 [  0   0   0 142   0]
 [ 10  11   2   0 100]]
```

Regression

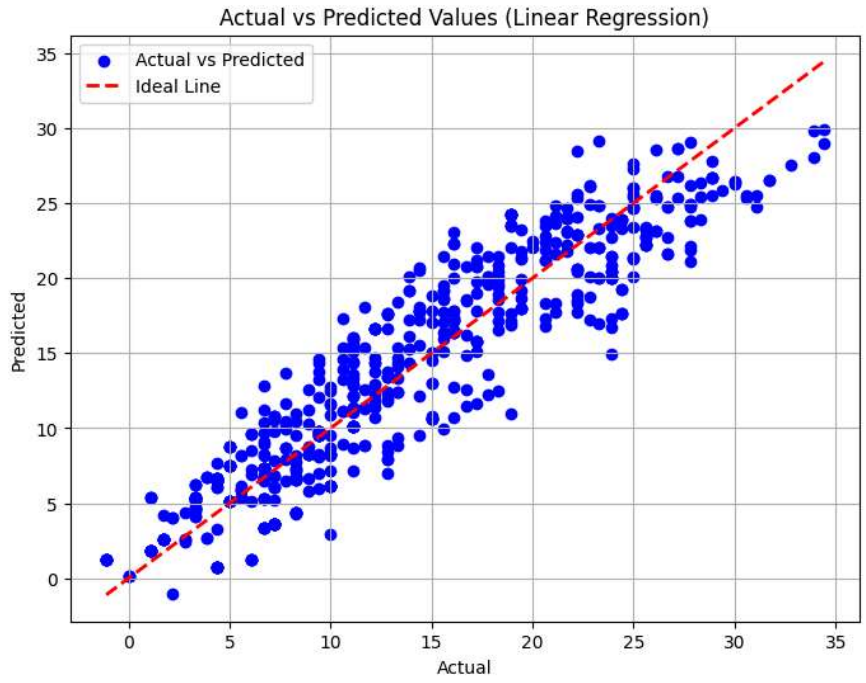
dt

	precipitation	temp_max	temp_min	wind	weather	day_of_week	month	year
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
...	...	...	...	...	...	...	...	...
1456	8.6	4.4	1.7	2.9	2	6	2	2015
1457	1.5	5.0	1.7	1.3	2	0	2	2015
1458	0.0	7.2	0.6	2.6	1	1	2	2015
1459	0.0	5.6	-1.0	3.4	4	2	2	2015
1460	0.0	5.6	-2.1	3.5	4	3	2	2015

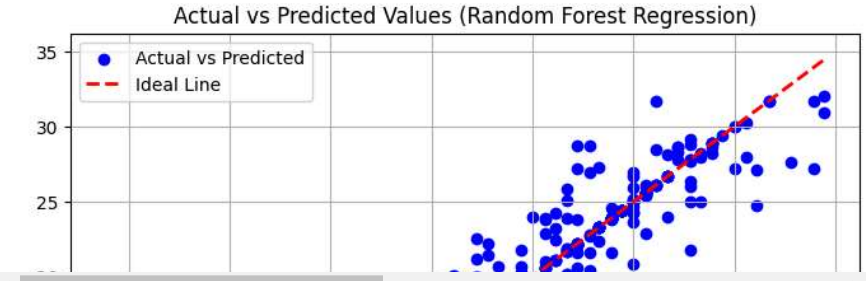
3204 rows × 8 columns

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LinearRegression, Lasso
from sklearn.ensemble import RandomForestRegressor, AdaBoostRegressor
from sklearn.tree import DecisionTreeRegressor
from xgboost import XGBRegressor
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
import matplotlib.pyplot as plt


X = dt.drop('temp_max', axis=1)
y = dt['temp_max']

# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Define models
models = {
    'Linear Regression': LinearRegression(),
    'Random Forest Regression': RandomForestRegressor(random_state=42),
    'Lasso Regression': Lasso(alpha=0.1, random_state=42),
    'Decision Tree Regression': DecisionTreeRegressor(random_state=42),
    'AdaBoost Regression': AdaBoostRegressor(random_state=42),
    'XGBoost Regression': XGBRegressor(random_state=42)
}

# Train and evaluate each model
for name, model in models.items():
    # Fit the model
    model.fit(X_train, y_train)

    # Make predictions
    y_pred = model.predict(X_test)

    # Calculate evaluation metrics
    mse = mean_squared_error(y_test, y_pred)
    mae = mean_absolute_error(y_test, y_pred)
    r2 = r2_score(y_test, y_pred)

    # Print the evaluation metrics
    print(f'{name}:')
    print(f'Mean Squared Error (MSE): {mse:.4f}')
    print(f'Mean Absolute Error (MAE): {mae:.4f}')
    print(f'R-squared (R2) Score: {r2:.4f}\n')

    # Plot actual vs predicted values
    plt.figure(figsize=(8, 6))
    plt.scatter(y_test, y_pred, color='blue', label='Actual vs Predicted')
    plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, color='red', label='Ideal Line')
    plt.xlabel('Actual')
    plt.ylabel('Predicted')
    plt.title(f'Actual vs Predicted Values ({name})')
    plt.legend()
    plt.grid(True)
    plt.show()
```

```
Linear Regression:
Mean Squared Error (MSE): 8.9683
Mean Absolute Error (MAE): 2.4559
R-squared (R2) Score: 0.8615

<ipython-input-22-a42182666e21>:51: UserWarning: color is redundantly defined by the
  plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, c
```




```
Random Forest Regression:
Mean Squared Error (MSE): 3.1469
Mean Absolute Error (MAE): 0.8813
R-squared (R2) Score: 0.9514

<ipython-input-22-a42182666e21>:51: UserWarning: color is redundantly defined by the
  plt.plot([y_test.min(), y_test.max()], [y_test.min(), y_test.max()], 'k--', lw=2, c
```



```
dt
```

	precipitation	temp_max	temp_min	wind	weather	day_of_week	month	year
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
0	0.0	12.8	5.0	4.7	0	6	4	2012
...	...	...	...	...	...	...	...	...
1456	8.6	4.4	1.7	2.9	2	6	2	2015
1457	1.5	5.0	1.7	1.3	2	0	2	2015
1458	0.0	7.2	0.6	2.6	1	1	2	2015
1459	0.0	5.6	-1.0	3.4	4	2	2	2015
1460	0.0	5.6	-2.1	3.5	4	3	2	2015

3204 rows × 8 columns

```
import numpy as np
import pandas as pd
df=pd.read_csv("/content/drive/MyDrive/seattle-weather.csv")
df
```

	date	precipitation	temp_max	temp_min	wind	weather
0	2012-01-01	0.0	12.8	5.0	4.7	drizzle
1	2012-01-02	10.9	10.6	2.8	4.5	rain
2	2012-01-03	0.8	11.7	7.2	2.3	rain
3	2012-01-04	20.3	12.2	5.6	4.7	rain
4	2012-01-05	1.3	8.9	2.8	6.1	rain
...	...	...	...	...	...	...
1456	2015-12-27	8.6	4.4	1.7	2.9	rain
1457	2015-12-28	1.5	5.0	1.7	1.3	rain
1458	2015-12-29	0.0	7.2	0.6	2.6	fog
1459	2015-12-30	0.0	5.6	-1.0	3.4	sun
1460	2015-12-31	0.0	5.6	-2.1	3.5	sun

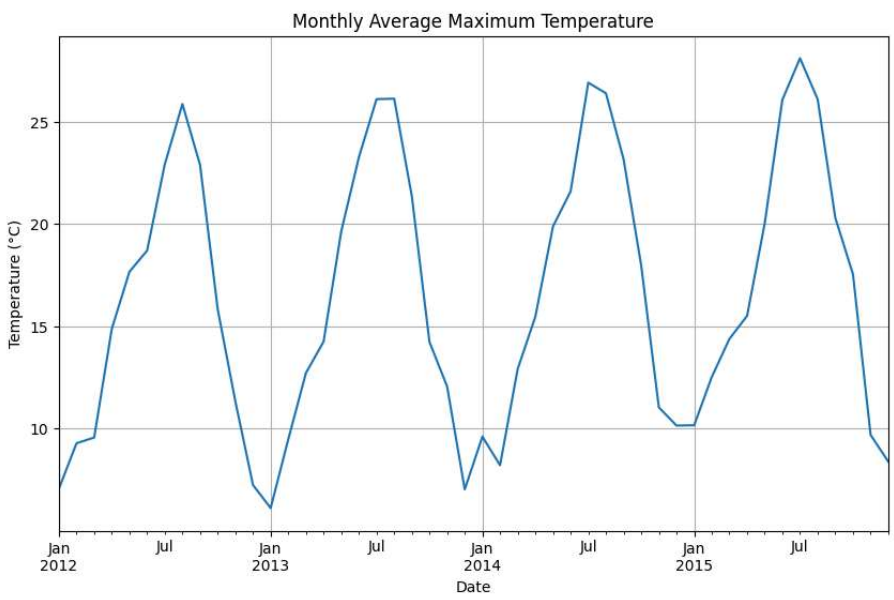
1461 rows × 6 columns

```
df['date'] = pd.to_datetime(df['date'])
df.set_index('date', inplace=True)
avg_temp = df['temp_max'].resample('M').mean()
print(avg_temp)
```

```
date
2012-01-31     7.054839
2012-02-29     9.275862
2012-03-31     9.554839
2012-04-30    14.873333
2012-05-31    17.661290
2012-06-30    18.693333
2012-07-31    22.906452
2012-08-31    25.858065
2012-09-30    22.880000
2012-10-31    15.829032
2012-11-30    11.326667
2012-12-31     7.235484
2013-01-31     6.106452
2013-02-28     9.467857
2013-03-31    12.709677
2013-04-30    14.243333
2013-05-31    19.625806
2013-06-30    23.253333
2013-07-31    26.093548
2013-08-31    26.119355
2013-09-30    21.360000
2013-10-31    14.229032
```

```
2013-11-30    12.053333
2013-12-31     7.022581
2014-01-31     9.600000
2014-02-28     8.200000
2014-03-31    12.906452
2014-04-30    15.460000
2014-05-31    19.870968
2014-06-30    21.590000
2014-07-31    26.900000
2014-08-31    26.383871
2014-09-30    23.163333
2014-10-31    17.961290
2014-11-30    11.030000
2014-12-31    10.138710
2015-01-31    10.154839
2015-02-28    12.517857
2015-03-31    14.377419
2015-04-30    15.503333
2015-05-31    20.025806
2015-06-30    26.063333
2015-07-31    28.093548
2015-08-31    26.087097
2015-09-30    20.293333
2015-10-31    17.538710
2015-11-30     9.683333
2015-12-31     8.380645
Freq: M, Name: temp_max, dtype: float64
```

```
import matplotlib.pyplot as plt

# Plot the monthly average maximum temperature
plt.figure(figsize=(10, 6))
avg_temp.plot()
plt.title('Monthly Average Maximum Temperature')
plt.xlabel('Date')
plt.ylabel('Temperature (°C)')
plt.grid(True)
plt.show()
```




```
!pip install pmdarima
```

```
Collecting pmdarima
  Downloading pmdarima-2.0.4-cp310-cp310-manylinux_2_17_x86_64.manylinux2014_x86_64.manylinux_2_28_x86_64.whl (2.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 2.1/2.1 MB 8.3 MB/s eta 0:00:00
Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.3.2)
Requirement already satisfied: Cython!=0.29.18,!=0.29.31,>=0.29 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (3.0.9)
```

```
Requirement already satisfied: numpy>=1.21.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.25.2)
Requirement already satisfied: pandas>=0.19 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.5.3)
Requirement already satisfied: scikit-learn>=0.22 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.2.2)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (1.11.4)
Requirement already satisfied: statsmodels>=0.13.2 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (0.14.1)
Requirement already satisfied: urllib3 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (2.0.7)
Requirement already satisfied: setuptools!=50.0.0,>=38.6.0 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (67.7.2)
Requirement already satisfied: packaging>=17.1 in /usr/local/lib/python3.10/dist-packages (from pmdarima) (24.0)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.10/dist-packages (from pandas>=0.19->pmdarima) (2023.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from scikit-learn>=0.22->pmdarima) (3.4.
Requirement already satisfied: patsy>=0.5.4 in /usr/local/lib/python3.10/dist-packages (from statsmodels>=0.13.2->pmdarima) (0.5.6)
Requirement already satisfied: six in /usr/local/lib/python3.10/dist-packages (from patsy>=0.5.4->statsmodels>=0.13.2->pmdarima) (1.16.0
Installing collected packages: pmdarima
Successfully installed pmdarima-2.0.4
```

```
import pandas as pd
import matplotlib.pyplot as plt
from statsmodels.tsa.arima.model import ARIMA
from sklearn.metrics import mean_squared_error
from pmdarima import auto_arima
from sklearn.metrics import r2_score
train_data = avg_temp[:-12]  # Using all except the last year for training
test_data = avg_temp[-12:]  # Using the last year for testing

# Automated ARIMA hyperparameter tuning
model = auto_arima(train_data, seasonal=True, m=12, suppress_warnings=True)

# Fit ARIMA model with tuned hyperparameters
model_fit = model.fit(train_data)

# Forecast
forecast = model_fit.predict(n_periods=len(test_data))

# Evaluate model
mse = mean_squared_error(test_data, forecast)
rmse = mse ** 0.5
print(f"Root Mean Squared Error (RMSE): {rmse}")
r2 = r2_score(test_data, forecast)
print(f"R-squared (R2) Score: {r2}")
# Plot results
plt.figure(figsize=(10, 6))
plt.plot(train_data.index, train_data, label='Training Data')
plt.plot(test_data.index, test_data, label='Test Data')
plt.plot(test_data.index, forecast, label='ARIMA Forecast', color='red')
plt.title('ARIMA Temperature Forecasting')
plt.xlabel('Date')
plt.ylabel('Temperature')
plt.legend()
plt.grid(True)
plt.show()
```