```
#objective :predicting mean temperature using existing data from Dailydelhiclimate using machine learning

import pandas as pd
import matplotlib.pyplot as plt
from sklearn.model_selection import TimeSeriesSplit
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_squared_error
testdata = pd.read_csv("/content/DailyDelhiClimateTest.csv",header=0)
testdata.head()
# here we display the first few rows of the dataset Dailydelhiclimate.
```

	date	meantemp	humidity	wind_speed	meanpressure
0	2017-01-01	15.913043	85.869565	2.743478	59.000000
1	2017-01-02	18.500000	77.222222	2.894444	1018.277778
2	2017-01-03	17.111111	81.888889	4.016667	1018.333333
3	2017-01-04	18.700000	70.050000	4.545000	1015.700000
4	2017-01-05	18.388889	74.944444	3.300000	1014.333333

```
testdata.shape
#the function returns tuple representing the number of rows and columns in the testData.
```

```
(114, 5)
```

```
traindata = pd.read_csv("/content/DailyDelhiClimateTrain.csv",header=0)
traindata.head()
```

	date	meantemp	humidity	wind_speed	meanpressure
0	2013-01-01	10.000000	84.500000	0.000000	1015.666667
1	2013-01-02	7.400000	92.000000	2.980000	1017.800000
2	2013-01-03	7.166667	87.000000	4.633333	1018.666667
3	2013-01-04	8.666667	71.333333	1.233333	1017.166667
4	2013-01-05	6.000000	86.833333	3.700000	1016.500000

```
traindata.shape
#the function returns tuple representing the number of rows and columns in the trainData.
```

```
(1462, 5)
```

```
traindata.describe()
#summary statistics of numerical columns
```

	meantemp	humidity	wind_speed	meanpressure
count	1462.000000	1462.000000	1462.000000	1462.000000
mean	25.495521	60.771702	6.802209	1011.104548
std	7.348103	16.769652	4.561602	180.231668
min	6.000000	13.428571	0.000000	-3.041667
25%	18.857143	50.375000	3.475000	1001.580357
50%	27.714286	62.625000	6.221667	1008.563492
75%	31.305804	72.218750	9.238235	1014.944901
max	38.714286	100.000000	42.220000	7679.333333

```
testdata.describe()
```

	meantemp	humidity	wind_speed	meanpressure
count	114.000000	114.000000	114.000000	114.000000
mean	21.713079	56.258362	8.143924	1004.035090
std	6.360072	19.068083	3.588049	89.474692
min	11.000000	17.750000	1.387500	59.000000
25%	16.437198	39.625000	5.563542	1007.437500
50%	19.875000	57.750000	8.069444	1012.739316
75%	27.705357	71.902778	10.068750	1016.739583
max	34.500000	95.833333	19.314286	1022.809524

```
traindata.isnull().sum()
#There is no missing values in both traindata and testdata
```

```
date            0
meantemp        0
humidity        0
wind_speed      0
meanpressure    0
dtype: int64
```

```
testdata.isnull().sum()
```

```
date            0
meantemp        0
humidity        0
wind_speed      0
meanpressure    0
dtype: int64
```

```
#Feature Engineering:
#Converting the 'date' column to datetime format and Extracting 'year', 'month', and 'day' from the 'date' column.
#Time Series Split:
#Splitting the training data using TimeSeriesSplit into 5 folds.
#Model Training:
for dataset in [traindata, testdata]:
    dataset['date'] = pd.to_datetime(dataset['date'])
    dataset['year'] = dataset['date'].dt.year
    dataset['month'] = dataset['date'].dt.month
    dataset['day'] = dataset['date'].dt.day

# Time Series Split
tscv = TimeSeriesSplit(n_splits=5)
for train_index, val_index in tscv.split(traindata):
    train_set, val_set = traindata.iloc[train_index], traindata.iloc[val_index]
```

```
features = ['year', 'month', 'day', 'humidity', 'wind_speed', 'meanpressure']

# Train Random Forest Regressor for Mean Temperature
X_train_temp, y_train_temp = train_set[features], train_set['meantemp']
X_val_temp, y_val_temp = val_set[features], val_set['meantemp']
model_temp = RandomForestRegressor(n_estimators=100, random_state=42)
model_temp.fit(X_train_temp, y_train_temp)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
X_train_humidity, y_train_humidity = train_set[features], train_set['humidity']
X_val_humidity, y_val_humidity = val_set[features], val_set['humidity']
model_humidity = RandomForestRegressor(n_estimators=100, random_state=42)
model_humidity.fit(X_train_humidity, y_train_humidity)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
#Train Random Forest Regressor for Wind Speed
X_train_wind, y_train_wind = train_set[features], train_set['wind_speed']
X_val_wind, y_val_wind = val_set[features], val_set['wind_speed']
model_wind = RandomForestRegressor(n_estimators=100, random_state=42)
model_wind.fit(X_train_wind, y_train_wind)
```

```
▾ RandomForestRegressor
RandomForestRegressor(random_state=42)
```

```
y_pred_temp = model_temp.predict(X_val_temp)
y_pred_humidity = model_humidity.predict(X_val_humidity)
y_pred_wind = model_wind.predict(X_val_wind)
```

```
mse_temp = mean_squared_error(y_val_temp, y_pred_temp)
mse_humidity = mean_squared_error(y_val_humidity, y_pred_humidity)
```

```
mse_wind = mean_squared_error(y_val_wind, y_pred_wind)

print(f'Mean Squared Error on Validation Set (Mean Temperature): {mse_temp}')
print(f'Mean Squared Error on Validation Set (Humidity): {mse_humidity}')
print(f'Mean Squared Error on Validation Set (Wind Speed): {mse_wind}')
#1.This means, on average, the squared difference between the predicted mean temperature values and the actual mean temperature values on the validation set.
#A higher MSE indicates larger deviations between the predicted and actual mean temperature values. In this case, an MSE of 5.4807 suggests some level of discrepancy between the predicted
#mean temp is my main focus
```

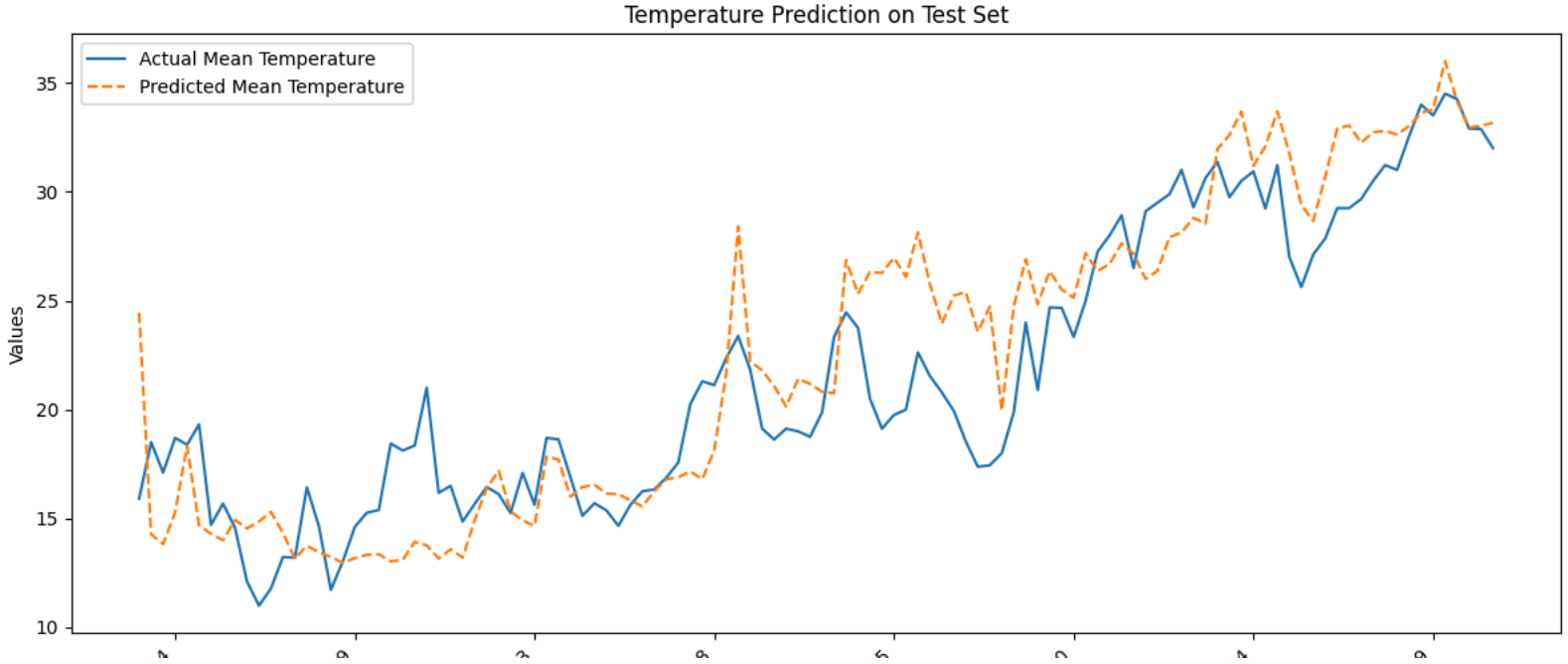
```
Mean Squared Error on Validation Set (Mean Temperature): 5.480743862871555
Mean Squared Error on Validation Set (Humidity): 0.04361086002605141
Mean Squared Error on Validation Set (Wind Speed): 0.0016501579434270843
```

```
X_test = testdata[features]

# Make predictions on the test sets
testdata['predicted_meantemp'] = model_temp.predict(X_test)
testdata['predicted_humidity'] = model_humidity.predict(X_test)
testdata['predicted_wind_speed'] = model_wind.predict(X_test)
```

```
plt.figure(figsize=(12, 6))
plt.plot(testdata['date'], testdata['meantemp'], label='Actual Mean Temperature')
plt.plot(testdata['date'], testdata['predicted_meantemp'], label='Predicted Mean Temperature', linestyle='dashed')
plt.xticks(rotation=45, ha='right')
plt.gca().xaxis.set_major_locator(plt.MaxNLocator(prune='both'))
plt.gca().xaxis.set_major_formatter(plt.matplotlib.dates.DateFormatter('%Y-%m-%d'))

plt.title('Temperature Prediction on Test Set')
plt.xlabel('Date')
plt.ylabel('Values')
plt.legend()
plt.tight_layout()
plt.show()
#Temperature Prediction plot  is shown below with  predicted mean temp coming out high around 18th february  to 20th march  and some days of april ,predicted mean temp is higher in most da
#hence in conclusion predicted mean temp  and actual mean temp increase and decrease trend is evident.
```


Temperature Prediction on Test Set
Actual Mean Temperature
Predicted Mean Temperature
Values
35
30
25
20
15
10