summer_six_conditions

December 7, 2022

1 Summer Season - 6 different weather conditions(long time frame)

1.1 Import libraries and dataset

[ ]:

```
import pandas as pd
import numpy as np
from datetime import datetime
date_format = "%Y-%m-%d"
```

[ ]:

```
summer = pd.read_csv('Datasets/summer.csv')
summer = summer[['datetime', 'conditions']]
```

[ ]:

```
summer.head()
```

[ ]:

```
     datetime        conditions
0  2000-07-01  Partially cloudy
1  2000-07-02  Partially cloudy
2  2000-07-03             Clear
3  2000-07-04  Partially cloudy
4  2000-07-05             Clear
```

1.2 Classify and separate data

[ ]:

```
classifier = {'Overcast':'overcast', 'Partially cloudy':'partially_cloudy',␣
 ↪'Clear':'clear', 'Rain, Partially cloudy':'rain_partially_cloudy', 'Rain':
 ↪'rain', 'Rain, Overcast':'rain_overcast'}

summer['condition'] = summer['conditions'].map(classifier)
```

[ ]:

```
summer.head()
```

[ ]:

```
     datetime        conditions          condition
0  2000-07-01  Partially cloudy  partially_cloudy
1  2000-07-02  Partially cloudy  partially_cloudy
2  2000-07-03             Clear             clear
3  2000-07-04  Partially cloudy  partially_cloudy
```

```
4  2000-07-05             Clear            clear
```

[ ]:
```
summer = summer[['datetime', 'condition']]
```

[ ]:
```
summer.head()
```

[ ]:
```
     datetime         condition
0  2000-07-01  partially_cloudy
1  2000-07-02  partially_cloudy
2  2000-07-03             clear
3  2000-07-04  partially_cloudy
4  2000-07-05             clear
```

[ ]:
```
train_start_date = '2002-01-01'
train_end_date = '2017-12-31'
summer_train = summer.loc[summer['datetime'].between(train_start_date,␣
 ↪train_end_date)]
summer_train = summer_train.reset_index()

test_start_date = '2018-01-01'
test_end_date = '2021-12-31'
summer_test = summer.loc[summer['datetime'].between(test_start_date,␣
 ↪test_end_date)]
summer_test = summer_test.reset_index()
```

1.3 Calculate proportions of conditions & Create transition matrix

[ ]:
```
# Initialize count variables

# 0: 'clear' - C
# 1: 'partially_cloudy' - PC
# 2: 'overcast' - OV
# 3: 'rain' - R
# 4: 'rain_partially_cloudy' - RPC
# 5: 'rain_overcast' - ROV

C_after_C_count = 0.0
PC_after_C_count = 0.0
OV_after_C_count = 0.0
R_after_C_count = 0.0
RPC_after_C_count = 0.0
ROV_after_C_count = 0.0

C_after_PC_count = 0.0
PC_after_PC_count = 0.0
OV_after_PC_count = 0.0
```

```
R_after_PC_count = 0.0
RPC_after_PC_count = 0.0
ROV_after_PC_count = 0.0

C_after_OV_count = 0.0
PC_after_OV_count = 0.0
OV_after_OV_count = 0.0
R_after_OV_count = 0.0
RPC_after_OV_count = 0.0
ROV_after_OV_count = 0.0

C_after_R_count = 0.0
PC_after_R_count = 0.0
OV_after_R_count = 0.0
R_after_R_count = 0.0
RPC_after_R_count = 0.0
ROV_after_R_count = 0.0

C_after_RPC_count = 0.0
PC_after_RPC_count = 0.0
OV_after_RPC_count = 0.0
R_after_RPC_count = 0.0
RPC_after_RPC_count = 0.0
ROV_after_RPC_count = 0.0

C_after_ROV_count = 0.0
PC_after_ROV_count = 0.0
OV_after_ROV_count = 0.0
R_after_ROV_count = 0.0
RPC_after_ROV_count = 0.0
ROV_after_ROV_count = 0.0
```

```
[ ]: summer_train
```

```
[ ]:       index    datetime          condition
     0       184  2002-07-01              clear
     1       185  2002-07-02  partially_cloudy
     2       186  2002-07-03  partially_cloudy
     3       187  2002-07-04  partially_cloudy
     4       188  2002-07-05  partially_cloudy
     …       …         …                …
     1467   1651  2017-09-26              clear
     1468   1652  2017-09-27              clear
     1469   1653  2017-09-28              clear
     1470   1654  2017-09-29              clear
     1471   1655  2017-09-30  partially_cloudy
```

```
[1472 rows x 3 columns]
```

[ ]:

```
# Count conditions

summer_train['condition_shift'] = summer_train['condition'].shift(-1)

for i in range(len(summer_train)):
    # Current 'clear'
    if summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'clear':
        C_after_C_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'clear':
        PC_after_C_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'clear':
        OV_after_C_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'clear':
        R_after_C_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'clear':
        RPC_after_C_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
 ↪loc[i, 'condition_shift'] == 'clear':
        ROV_after_C_count += 1
    # Current 'partially_cloudy'
    elif summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'partially_cloudy':
        C_after_PC_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'partially_cloudy':
        PC_after_PC_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'partially_cloudy':
        OV_after_PC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'partially_cloudy':
        R_after_PC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'partially_cloudy':
        RPC_after_PC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
 ↪loc[i, 'condition_shift'] == 'partially_cloudy':
        ROV_after_PC_count += 1
    # Current 'overcast'
    elif summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
↪'condition_shift'] == 'overcast':
        C_after_OV_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
↪summer_train.loc[i, 'condition_shift'] == 'overcast':
        PC_after_OV_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
↪'condition_shift'] == 'overcast':
        OV_after_OV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
↪'condition_shift'] == 'overcast':
        R_after_OV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
↪summer_train.loc[i, 'condition_shift'] == 'overcast':
        RPC_after_OV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
↪loc[i, 'condition_shift'] == 'overcast':
        ROV_after_OV_count += 1
    # Current 'rain'
    elif summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
↪'condition_shift'] == 'rain':
        C_after_R_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
↪summer_train.loc[i, 'condition_shift'] == 'rain':
        PC_after_R_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
↪'condition_shift'] == 'rain':
        OV_after_R_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
↪'condition_shift'] == 'rain':
        R_after_R_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
↪summer_train.loc[i, 'condition_shift'] == 'rain':
        RPC_after_R_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
↪loc[i, 'condition_shift'] == 'rain':
        ROV_after_R_count += 1
    # Current 'rain_partially_cloudy'
    elif summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
↪'condition_shift'] == 'rain_partially_cloudy':
        C_after_RPC_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
↪summer_train.loc[i, 'condition_shift'] == 'rain_partially_cloudy':
        PC_after_RPC_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
↪'condition_shift'] == 'rain_partially_cloudy':
        OV_after_RPC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'rain_partially_cloudy':
        R_after_RPC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'rain_partially_cloudy':
        RPC_after_RPC_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
 ↪loc[i, 'condition_shift'] == 'rain_partially_cloudy':
        ROV_after_RPC_count += 1
    # Current 'rain_overcast'
    elif summer_train.loc[i, 'condition'] == 'clear' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'rain_overcast':
        C_after_ROV_count += 1
    elif summer_train.loc[i, 'condition'] == 'partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'rain_overcast':
        PC_after_ROV_count += 1
    elif summer_train.loc[i, 'condition'] == 'overcast' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'rain_overcast':
        OV_after_ROV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain' and summer_train.loc[i,␣
 ↪'condition_shift'] == 'rain_overcast':
        R_after_ROV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_partially_cloudy' and␣
 ↪summer_train.loc[i, 'condition_shift'] == 'rain_overcast':
        RPC_after_ROV_count += 1
    elif summer_train.loc[i, 'condition'] == 'rain_overcast' and summer_train.
 ↪loc[i, 'condition_shift'] == 'rain_overcast':
        ROV_after_ROV_count += 1
```

[ ]:

```
current_C_total = C_after_C_count + PC_after_C_count + OV_after_C_count +␣
 ↪R_after_C_count + RPC_after_C_count + ROV_after_C_count
current_PC_total = C_after_PC_count + PC_after_PC_count + OV_after_PC_count +␣
 ↪R_after_PC_count + RPC_after_PC_count + ROV_after_PC_count
current_OV_total = C_after_OV_count + PC_after_OV_count + OV_after_OV_count +␣
 ↪R_after_OV_count + RPC_after_OV_count + ROV_after_OV_count
current_R_total =C_after_R_count + PC_after_R_count + OV_after_R_count +␣
 ↪R_after_R_count + RPC_after_R_count + ROV_after_R_count
current_RPC_total = C_after_RPC_count + PC_after_RPC_count + OV_after_RPC_count␣
 ↪+ R_after_RPC_count + RPC_after_RPC_count + ROV_after_RPC_count
current_ROV_total = C_after_ROV_count + PC_after_ROV_count + OV_after_ROV_count␣
 ↪+ R_after_ROV_count + RPC_after_ROV_count + ROV_after_ROV_count
```

[ ]:

```
C_after_C_prob = C_after_C_count / current_C_total
PC_after_C_prob = PC_after_C_count / current_C_total
OV_after_C_prob = OV_after_C_count / current_C_total
```

```
R_after_C_prob = R_after_C_count / current_C_total
RPC_after_C_prob = RPC_after_C_count / current_C_total
ROV_after_C_prob = ROV_after_C_count / current_C_total

C_after_PC_prob = C_after_PC_count / current_PC_total
PC_after_PC_prob = PC_after_PC_count / current_PC_total
OV_after_PC_prob = OV_after_PC_count / current_PC_total
R_after_PC_prob = R_after_PC_count / current_PC_total
RPC_after_PC_prob = RPC_after_PC_count / current_PC_total
ROV_after_PC_prob = ROV_after_PC_count / current_PC_total

C_after_OV_prob = C_after_OV_count / current_OV_total
PC_after_OV_prob = PC_after_OV_count / current_OV_total
OV_after_OV_prob = OV_after_OV_count / current_OV_total
R_after_OV_prob = R_after_OV_count / current_OV_total
RPC_after_OV_prob = RPC_after_OV_count / current_OV_total
ROV_after_OV_prob = ROV_after_OV_count / current_OV_total

C_after_R_prob = C_after_R_count / current_R_total
PC_after_R_prob = PC_after_R_count / current_R_total
OV_after_R_prob = OV_after_R_count / current_R_total
R_after_R_prob = R_after_R_count / current_R_total
RPC_after_R_prob = RPC_after_R_count / current_R_total
ROV_after_R_prob = ROV_after_R_count / current_R_total

C_after_RPC_prob = C_after_RPC_count / current_RPC_total
PC_after_RPC_prob = PC_after_RPC_count / current_RPC_total
OV_after_RPC_prob = OV_after_RPC_count / current_RPC_total
R_after_RPC_prob = R_after_RPC_count / current_RPC_total
RPC_after_RPC_prob = RPC_after_RPC_count / current_RPC_total
ROV_after_RPC_prob = ROV_after_RPC_count / current_RPC_total

C_after_ROV_prob = C_after_ROV_count / current_ROV_total
PC_after_ROV_prob = PC_after_ROV_count / current_ROV_total
OV_after_ROV_prob = OV_after_ROV_count / current_ROV_total
R_after_ROV_prob = R_after_ROV_count / current_ROV_total
RPC_after_ROV_prob = RPC_after_ROV_count / current_ROV_total
ROV_after_ROV_prob = ROV_after_ROV_count / current_ROV_total
```

[ ]:

```
# Printing our probabilities for 6x6 transition matrix:
print(C_after_C_prob)
print(PC_after_C_prob)
print(OV_after_C_prob)
print(R_after_C_prob)
print(RPC_after_C_prob)
print(ROV_after_C_prob)
```

```python
print(C_after_PC_prob)
print(PC_after_PC_prob)
print(OV_after_PC_prob)
print(R_after_PC_prob)
print(RPC_after_PC_prob)
print(ROV_after_PC_prob)

print(C_after_OV_prob)
print(PC_after_OV_prob)
print(OV_after_OV_prob)
print(R_after_OV_prob)
print(RPC_after_OV_prob)
print(ROV_after_OV_prob)

print(C_after_R_prob)
print(PC_after_R_prob)
print(OV_after_R_prob)
print(R_after_R_prob)
print(RPC_after_R_prob)
print(ROV_after_R_prob)

print(C_after_RPC_prob)
print(PC_after_RPC_prob)
print(OV_after_RPC_prob)
print(R_after_RPC_prob)
print(RPC_after_RPC_prob)
print(ROV_after_RPC_prob)

print(C_after_ROV_prob)
print(PC_after_ROV_prob)
print(OV_after_ROV_prob)
print(R_after_ROV_prob)
print(RPC_after_ROV_prob)
print(ROV_after_ROV_prob)
```

```
0.693069306930693
0.2623762376237624
0.0016501650165016502
0.02145214521452145
0.02145214521452145
0.0
0.21106821106821108
0.7348777348777349
0.021879021879021878
0.006435006435006435
0.02574002574002574
0.0
```

```
0.13636363636363635
0.7272727272727273
0.09090909090909091
0.0
0.0
0.045454545454545456
0.6190476190476191
0.2857142857142857
0.0
0.047619047619047616
0.047619047619047616
0.0
0.16279069767441862
0.5348837209302325
0.046511627906976744
0.046511627906976744
0.18604651162790697
0.023255813953488372
0.0
0.5
0.0
0.0
0.5
0.0
```

[ ]:

```
# Checking that each row in the transition matrix adds up to 1:
print(C_after_C_prob + PC_after_C_prob + OV_after_C_prob + R_after_C_prob +␣
 ↪RPC_after_C_prob + ROV_after_C_prob)
print(C_after_PC_prob + PC_after_PC_prob + OV_after_PC_prob + R_after_PC_prob +␣
 ↪RPC_after_PC_prob + ROV_after_PC_prob)
print(C_after_OV_prob + PC_after_OV_prob + OV_after_OV_prob + R_after_OV_prob +␣
 ↪RPC_after_OV_prob + ROV_after_OV_prob)
print(C_after_R_prob + PC_after_R_prob + OV_after_R_prob + R_after_R_prob +␣
 ↪RPC_after_R_prob + ROV_after_R_prob)
print(C_after_RPC_prob + PC_after_RPC_prob + OV_after_RPC_prob +␣
 ↪R_after_RPC_prob + RPC_after_RPC_prob + ROV_after_RPC_prob)
print(C_after_ROV_prob + PC_after_ROV_prob + OV_after_ROV_prob +␣
 ↪R_after_ROV_prob + RPC_after_ROV_prob + ROV_after_ROV_prob)
```

```
0.9999999999999999
1.0
1.0
1.0
0.9999999999999999
1.0
```

[ ]:

```
# Creating the transition matrix:
transition_matrix = [[C_after_C_prob, PC_after_C_prob, OV_after_C_prob,␣
 ↪R_after_C_prob, RPC_after_C_prob, ROV_after_C_prob],
                     [C_after_PC_prob, PC_after_PC_prob, OV_after_PC_prob,␣
 ↪R_after_PC_prob, RPC_after_PC_prob, ROV_after_PC_prob],
                     [C_after_OV_prob, PC_after_OV_prob, OV_after_OV_prob,␣
 ↪R_after_OV_prob, RPC_after_OV_prob, ROV_after_OV_prob],
                     [C_after_R_prob, PC_after_R_prob, OV_after_R_prob,␣
 ↪R_after_R_prob, RPC_after_R_prob, ROV_after_R_prob],
                     [C_after_RPC_prob, PC_after_RPC_prob, OV_after_RPC_prob,␣
 ↪R_after_RPC_prob, RPC_after_RPC_prob, ROV_after_RPC_prob],
                     [C_after_ROV_prob, PC_after_ROV_prob, OV_after_ROV_prob,␣
 ↪R_after_ROV_prob, RPC_after_ROV_prob, ROV_after_ROV_prob]]
print(transition_matrix)
```

```
[[0.693069306930693, 0.2623762376237624, 0.0016501650165016502,
0.02145214521452145, 0.02145214521452145, 0.0], [0.21106821106821108,
0.7348777348777349, 0.021879021879021878, 0.006435006435006435,
0.02574002574002574, 0.0], [0.13636363636363635, 0.7272727272727273,
0.09090909090909091, 0.0, 0.0, 0.045454545454545456], [0.6190476190476191,
0.2857142857142857, 0.0, 0.047619047619047616, 0.047619047619047616, 0.0],
[0.16279069767441862, 0.5348837209302325, 0.046511627906976744,
0.046511627906976744, 0.18604651162790697, 0.023255813953488372], [0.0, 0.5,
0.0, 0.0, 0.5, 0.0]]
```

[ ]:

```
t_array = np.array(transition_matrix)
print(t_array)
```

```
[[0.69306931 0.26237624 0.00165017 0.02145215 0.02145215 0.        ]
 [0.21106821 0.73487773 0.02187902 0.00643501 0.02574003 0.        ]
 [0.13636364 0.72727273 0.09090909 0.         0.         0.04545455]
 [0.61904762 0.28571429 0.         0.04761905 0.04761905 0.        ]
 [0.1627907  0.53488372 0.04651163 0.04651163 0.18604651 0.02325581]
 [0.         0.5        0.         0.         0.5        0.        ]]
```

[ ]:

```
summer_test.head(1)
```

[ ]:

```
   index    datetime          condition
0   1656  2018-07-01  partially_cloudy
```

First day of summer 2018: partially_cloudy

[ ]:

```
def predict_weather_six_conditions(test_data):
    state = {0:'clear', 1:'partially_cloudy', 2:'overcast', 3:'rain', 4:
 ↪'rain_partially_cloudy', 5:'rain_overcast'}
    n = len(test_data) # how many steps to test
    start_state = 0 # 0 = clear
    test_result = test_data.copy()

    prev_state = start_state
    result = [state[start_state]]
    while n-1:
        curr_state = np.random.choice([0,1,2,3,4,5], p=t_array[prev_state])␣
 ↪#taking the probability from the transition matrix
        result.append(state[curr_state])
        prev_state = curr_state
        n -= 1

    # curr_state = np.random.choice([0,1,2,3,4,5], p=t_array[prev_state])␣
 ↪#taking the probability from the transition matrix
    # result.append(state[curr_state])

    test_result['predicted_condition'] = result

    return test_result

def find_accuracy(predicted_result):
    correct_count = 0.0

    for i in range(len(predicted_result)):
        if predicted_result.loc[i, 'condition'] == predicted_result.loc[i,␣
 ↪'predicted_condition']:
            correct_count += 1

    correct_prop = correct_count / len(predicted_result)

    return correct_prop

def run_predictions_return_avg_accuracy(test_data, trial_count):
    accuracy_sum = 0.0
    for i in range(trial_count):
        predicted_result = predict_weather_six_conditions(test_data)
        accuracy = find_accuracy(predicted_result)
        accuracy_sum += accuracy
    avg_accuracy = accuracy_sum / trial_count

    return avg_accuracy
```

[ ]:

```
# Sample prediction (for table graphic)

sample_prediction = predict_weather_six_conditions(summer_test)
sample_accuracy = find_accuracy(sample_prediction)
print(sample_prediction.head())
print(sample_accuracy)
```

```
   index    datetime         condition predicted_condition
0   1656  2018-07-01  partially_cloudy               clear
1   1657  2018-07-02  partially_cloudy               clear
2   1658  2018-07-03  partially_cloudy               clear
3   1659  2018-07-04  partially_cloudy               clear
4   1660  2018-07-05             clear               clear
0.46195652173913043
```

[ ]:

```
run_predictions_return_avg_accuracy(summer_test, 100)
```

[ ]:

```
0.45853260869565204
```