winter_six_conditions

December 7, 2022

1 Winter 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"
```

[ ]:

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

[ ]:

```
winter_seasons.head(200)
```

[ ]:

```
       datetime        conditions
0    2000-01-01  Partially cloudy
1    2000-01-02             Clear
2    2000-01-03             Clear
3    2000-01-04             Clear
4    2000-01-05             Clear
..          …                 …
195  2002-01-15  Partially cloudy
196  2002-01-16  Partially cloudy
197  2002-01-17             Clear
198  2002-01-18  Partially cloudy
199  2002-01-19  Partially cloudy

[200 rows x 2 columns]
```

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'}

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

[ ]:

```
winter_seasons.head()
```

[ ]:

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

[ ]:

```
winter_seasons = winter_seasons[['datetime', 'condition']]
```

[ ]:

```
winter_seasons.head()
```

[ ]:

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

[ ]:

```
train_start_date = '2002-01-01'
train_end_date = '2017-12-31'
winter_seasons_train = winter_seasons.loc[winter_seasons['datetime'].
 ↪between(train_start_date, train_end_date)]
winter_seasons_train = winter_seasons_train.reset_index()

test_start_date = '2018-01-01'
test_end_date = '2021-12-31'
winter_seasons_test = winter_seasons.loc[winter_seasons['datetime'].
 ↪between(test_start_date, test_end_date)]
winter_seasons_test = winter_seasons_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
```

[ ]:
```
winter_seasons_train
```

[ ]:
```
      index    datetime              condition
0       181  2002-01-01        partially_cloudy
1       182  2002-01-02   rain_partially_cloudy
2       183  2002-01-03   rain_partially_cloudy
3       184  2002-01-04        partially_cloudy
4       185  2002-01-05        partially_cloudy
…       …          …                    …
```

```
1439   1620  2017-03-27                     clear
1440   1621  2017-03-28                     clear
1441   1622  2017-03-29                     clear
1442   1623  2017-03-30                     clear
1443   1624  2017-03-31                     clear

[1444 rows x 3 columns]
```

[ ]:
```
winter_seasons_train['condition_shift'] = winter_seasons_train['condition'].
 ↪shift(-1)
winter_seasons_train.head(30)
```

[ ]:
```
    index    datetime             condition         condition_shift
0     181  2002-01-01      partially_cloudy  rain_partially_cloudy
1     182  2002-01-02  rain_partially_cloudy  rain_partially_cloudy
2     183  2002-01-03  rain_partially_cloudy       partially_cloudy
3     184  2002-01-04      partially_cloudy       partially_cloudy
4     185  2002-01-05      partially_cloudy       partially_cloudy
5     186  2002-01-06      partially_cloudy       partially_cloudy
6     187  2002-01-07      partially_cloudy       partially_cloudy
7     188  2002-01-08      partially_cloudy       partially_cloudy
8     189  2002-01-09      partially_cloudy                  clear
9     190  2002-01-10                 clear                  clear
10    191  2002-01-11                 clear                  clear
11    192  2002-01-12                 clear                  clear
12    193  2002-01-13                 clear       partially_cloudy
13    194  2002-01-14      partially_cloudy       partially_cloudy
14    195  2002-01-15      partially_cloudy       partially_cloudy
15    196  2002-01-16      partially_cloudy                  clear
16    197  2002-01-17                 clear       partially_cloudy
17    198  2002-01-18      partially_cloudy       partially_cloudy
18    199  2002-01-19      partially_cloudy                  clear
19    200  2002-01-20                 clear                  clear
20    201  2002-01-21                 clear       partially_cloudy
21    202  2002-01-22      partially_cloudy                  clear
22    203  2002-01-23                 clear                  clear
23    204  2002-01-24                 clear       partially_cloudy
24    205  2002-01-25      partially_cloudy       partially_cloudy
25    206  2002-01-26      partially_cloudy  rain_partially_cloudy
26    207  2002-01-27  rain_partially_cloudy  rain_partially_cloudy
27    208  2002-01-28  rain_partially_cloudy  rain_partially_cloudy
28    209  2002-01-29  rain_partially_cloudy                  clear
29    210  2002-01-30                 clear                  clear
```

[ ]:
```
# Count conditions
```

```
winter_seasons_train['condition_shift'] = winter_seasons_train['condition'].
↪shift(-1)

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

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)
```

```
0.6776611694152923
0.15892053973013492
0.0
0.043478260869565216
0.11694152923538231
0.0029985007496251873
0.34382566585956414
0.4915254237288136
0.01937046004842615
0.009685230024213076
0.1234866828087167
0.012106537530266344
0.07692307692307693
0.8461538461538461
```

```
0.07692307692307693
0.0
0.0
0.0
0.375
0.15
0.0
0.05
0.375
0.05
0.21011673151750973
0.2607003891050584
0.011673151750972763
0.019455252918287938
0.377431906614786
0.12062256809338522
0.03773584905660377
0.39622641509433965
0.018867924528301886
0.0
0.3018867924528302
0.24528301886792453
```

[ ]:

```
# 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)
```

```
1.0
1.0
1.0
1.0
1.0
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)
```

```
[[0.6776611694152923, 0.15892053973013492, 0.0, 0.043478260869565216,
0.11694152923538231, 0.0029985007496251873], [0.34382566585956414,
0.4915254237288136, 0.01937046004842615, 0.009685230024213076,
0.1234866828087167, 0.012106537530266344], [0.07692307692307693,
0.8461538461538461, 0.07692307692307693, 0.0, 0.0, 0.0], [0.375, 0.15, 0.0,
0.05, 0.375, 0.05], [0.21011673151750973, 0.2607003891050584,
0.011673151750972763, 0.019455252918287938, 0.377431906614786,
0.12062256809338522], [0.03773584905660377, 0.39622641509433965,
0.018867924528301886, 0.0, 0.3018867924528302, 0.24528301886792453]]
```

[ ]:

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

```
[[0.67766117 0.15892054 0.         0.04347826 0.11694153 0.0029985 ]
 [0.34382567 0.49152542 0.01937046 0.00968523 0.12348668 0.01210654]
 [0.07692308 0.84615385 0.07692308 0.         0.         0.        ]
 [0.375      0.15       0.         0.05       0.375      0.05      ]
 [0.21011673 0.26070039 0.01167315 0.01945525 0.37743191 0.12062257]
 [0.03773585 0.39622642 0.01886792 0.         0.30188679 0.24528302]]
```

[ ]:

```
winter_seasons_test.head(1)
```

[ ]:

```
   index    datetime condition
0   1625  2018-01-01     clear
```

First Day of spring 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 = []
    result.append(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(winter_seasons_test)
sample_accuracy = find_accuracy(sample_prediction)
print(sample_prediction.head())
print(sample_accuracy)
```

```
   index     datetime          condition predicted_condition
```

```
0   1625  2018-01-01             clear             clear
1   1626  2018-01-02             clear             clear
2   1627  2018-01-03             clear             clear
3   1628  2018-01-04  partially_cloudy  partially_cloudy
4   1629  2018-01-05  partially_cloudy  partially_cloudy
0.3878116343490305
```

```
[ ]: run_predictions_return_avg_accuracy(winter_seasons_test, 100)
```

```
[ ]: 0.35728531855955675
```