All Seasons - 6 different weather conditions(long time frame)

Import libraries and dataset

In [ ]:

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

In [ ]:

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

In [ ]:

```
all_seasons.head()
```

Out[ ]:

	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

Classify and separate data

In [ ]:

```
classifier = {'Overcast':'overcast', 'Partially cloudy':'partially_cloudy'

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

In [ ]:

```
all_seasons.head()
```

Out[ ]:

	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

In [ ]:

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

In [ ]:

```
all_seasons.head()
```

Out[ ]:

	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

In [ ]:

```
train_start_date = '2002-01-01'
train_end_date = '2017-12-31'
all_seasons_train = all_seasons.loc[all_seasons['datetime'].between(train_s
all_seasons_train = all_seasons_train.reset_index()

test_start_date = '2018-01-01'
test_end_date = '2021-12-31'
all_seasons_test = all_seasons.loc[all_seasons['datetime'].between(test_sta
all_seasons_test = all_seasons_test.reset_index()
```

Calculate proportions of conditions & Create transition matrix

We will refer to rain is 'R' and no rain as 'N'

In [ ]:

```
# 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
```

In [ ]:

```
all_seasons_train
```

Out[ ]:

	index	datetime	condition
0	731	2002-01-01	partially_cloudy
1	732	2002-01-02	rain_partially_cloudy
2	733	2002-01-03	rain_partially_cloudy
3	734	2002-01-04	partially_cloudy
4	735	2002-01-05	partially_cloudy
...	...	...	...
5839	6570	2017-12-27	clear
5840	6571	2017-12-28	clear
5841	6572	2017-12-29	clear
5842	6573	2017-12-30	partially_cloudy
5843	6574	2017-12-31	partially_cloudy

5844 rows × 3 columns

In [ ]:

```
# Count conditions

all_seasons_train['condition_shift'] = all_seasons_train['condition'].shif

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

In [ ]:

```
current_C_total = C_after_C_count + PC_after_C_count + OV_after_C_count + R
current_PC_total = C_after_PC_count + PC_after_PC_count + OV_after_PC_count
current_OV_total = C_after_OV_count + PC_after_OV_count + OV_after_OV_count
current_R_total =C_after_R_count + PC_after_R_count + OV_after_R_count + R_
current_RPC_total = C_after_RPC_count + PC_after_RPC_count + OV_after_RPC_c
current_ROV_total = C_after_ROV_count + PC_after_ROV_count + OV_after_ROV_c
```

In [ ]:

```
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
```

In [ ]:

```
# 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)
```

```
0.6896135265700483
0.19927536231884058
0.0008051529790660225
0.033816425120772944
0.07286634460547504
0.0036231884057971015
0.2502120441051739
0.6395250212044106
0.028413910093299407
0.008905852417302799
0.06658184902459711
0.006361323155216285
0.027522935779816515
0.7431192660550459
0.1559633027522936
0.0
0.045871559633027525
0.027522935779816515
0.36551724137931035
0.21379310344827587
0.0
0.14482758620689656
0.2482758620689655
0.027586206896551724
0.19032258064516128
0.3225806451612903
0.024193548387096774
0.02903225806451613
0.33548387096774196
0.09838709677419355
0.05511811023622047
0.33858267716535434
0.06299212598425197
0.007874015748031496
0.25984251968503935
0.2755905511811024
```

In [ ]:

```
# 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
print(C_after_PC_prob + PC_after_PC_prob + OV_after_PC_prob + R_after_PC_pr
print(C_after_OV_prob + PC_after_OV_prob + OV_after_OV_prob + R_after_OV_pr
print(C_after_R_prob + PC_after_R_prob + OV_after_R_prob + R_after_R_prob
print(C_after_RPC_prob + PC_after_RPC_prob + OV_after_RPC_prob + R_after_R
print(C_after_ROV_prob + PC_after_ROV_prob + OV_after_ROV_prob + R_after_R
```

```
1.0
1.0
1.0
1.0000000000000002
1.0
1.0
```

In [ ]:

```
# Creating the transition matrix:
transition_matrix = [[C_after_C_prob, PC_after_C_prob, OV_after_C_prob, R_a
                    [C_after_PC_prob, PC_after_PC_prob, OV_after_PC_prob, R
                    [C_after_OV_prob, PC_after_OV_prob, OV_after_OV_prob, R
                    [C_after_R_prob, PC_after_R_prob, OV_after_R_prob, R_a
                    [C_after_RPC_prob, PC_after_RPC_prob, OV_after_RPC_prob
                    [C_after_ROV_prob, PC_after_ROV_prob, OV_after_ROV_prob
print(transition_matrix)
```

```
[[0.6896135265700483, 0.19927536231884058, 0.0008051529790660225, 0.0338164
25120772944, 0.07286634460547504, 0.0036231884057971015], [0.25021204410517
39, 0.6395250212044106, 0.02841391009329407, 0.008905852417302799, 0.06658
184902459711, 0.006361323155216285], [0.027522935779816515, 0.7431192660550
459, 0.1559633027522936, 0.0, 0.045871559633027525, 0.027522935779816515],
[0.36551724137931035, 0.21379310344827587, 0.0, 0.14482758620689656, 0.2482
758620689655, 0.027586206896551724], [0.19032258064516128, 0.32258064516129
03, 0.024193548387096774, 0.02903225806451613, 0.33548387096774196, 0.09838
709677419355], [0.05511811023622047, 0.33858267716535434, 0.062992125984251
97, 0.007874015748031496, 0.25984251968503935, 0.2755905511811024]]
```

In [ ]:

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

```
[[0.68961353 0.19927536 0.00080515 0.03381643 0.07286634 0.00362319]
 [0.25021204 0.63952502 0.02841391 0.00890585 0.06658185 0.00636132]
 [0.02752294 0.74311927 0.1559633  0.         0.04587156 0.02752294]
 [0.36551724 0.2137931  0.         0.14482759 0.24827586 0.02758621]
 [0.19032258 0.32258065 0.02419355 0.02903226 0.33548387 0.0983871 ]
 [0.05511811 0.33858268 0.06299213 0.00787402 0.25984252 0.27559055]]
```

In [ ]:

```
all_seasons_test.head(1)
```

Out[ ]:

	index	datetime	condition
0	6575	2018-01-01	clear

First Day of 2018: clear

In [ ]:

```
def predict_weather_six_conditions(test_data):
    state = {0:'clear', 1:'partially_cloudy', 2:'overcast', 3:'rain', 4:'ra
    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]
        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]) #
    # 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,
            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
```

In [ ]:

```
# Sample prediction (for table graphic)

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

```
   index    datetime         condition predicted_condition
0   6575  2018-01-01             clear               clear
1   6576  2018-01-02             clear               clear
2   6577  2018-01-03             clear               clear
3   6578  2018-01-04  partially_cloudy    partially_cloudy
4   6579  2018-01-05  partially_cloudy                rain
0.3750855578370979
```

In [ ]:

```
run_predictions_return_avg_accuracy(all_seasons_test, 100)
```

Out[ ]:

```
0.36932238193018485
```