All Seaons - 6 different weather conditions(short 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 = '2017-01-01'
train_end_date = '2020-12-31'
all_seasons_train = all_seasons.loc[all_seasons['datetime'].between(train_s
all_seasons_train = all_seasons_train.reset_index()

test_start_date = '2021-01-01'
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	6210	2017-01-01	partially_cloudy
1	6211	2017-01-02	partially_cloudy
2	6212	2017-01-03	partially_cloudy
3	6213	2017-01-04	rain_overcast
4	6214	2017-01-05	rain_partially_cloudy
...	...	...	...
1456	7666	2020-12-27	partially_cloudy
1457	7667	2020-12-28	rain_partially_cloudy
1458	7668	2020-12-29	clear
1459	7669	2020-12-30	clear
1460	7670	2020-12-31	clear

1461 rows × 3 columns

In [ ]:

```
# Count conditions

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

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_
        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_seaso
        OV_after_OV_count += 1
    elif all_seasons_train.loc[i, 'condition'] == 'rain' and all_seasons_t
        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_
        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_seaso
        OV_after_R_count += 1
    elif all_seasons_train.loc[i, 'condition'] == 'rain' and all_seasons_t
        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_
        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_seaso
        OV_after_RPC_count += 1
    elif all_seasons_train.loc[i, 'condition'] == 'rain' and all_seasons_t
        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_
        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_seaso
        OV_after_ROV_count += 1
    elif all_seasons_train.loc[i, 'condition'] == 'rain' and all_seasons_t
        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 + I
current_PC_total = C_after_PC_count + PC_after_PC_count + OV_after_PC_coun
current_OV_total = C_after_OV_count + PC_after_OV_count + OV_after_OV_coun
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)
```

```
0.7429775280898876
0.1853932584269663
0.0
0.008426966292134831
0.06320224719101124
0.0
0.27706422018348625
0.6220183486238532
0.03669724770642202
0.0
0.05871559633027523
0.005504587155963303
0.03225806451612903
0.7741935483870968
0.1935483870967742
0.0
0.0
0.0
0.2857142857142857
0.2857142857142857
0.0
0.0
0.42857142857142855
0.0
0.18
0.26666666666666666
0.02
0.006666666666666667
0.44666666666666666
0.08
0.06666666666666667
0.6
0.13333333333333333
0.0
0.2
0.0
```

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_p
print(C_after_OV_prob + PC_after_OV_prob + OV_after_OV_prob + R_after_OV_p
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
```

```
0.9999999999999999
1.0
1.0
1.0
0.9999999999999999
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, I
                    [C_after_OV_prob, PC_after_OV_prob, OV_after_OV_prob, I
                    [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_prol
                    [C_after_ROV_prob, PC_after_ROV_prob, OV_after_ROV_prol
print(transition_matrix)
```

```
[[0.7429775280898876, 0.1853932584269663, 0.0, 0.008426966292134831, 0.0632
0224719101124, 0.0], [0.27706422018348625, 0.6220183486238532, 0.0366972477
0642202, 0.0, 0.05871559633027523, 0.005504587155963303], [0.03225806451612
903, 0.7741935483870968, 0.1935483870967742, 0.0, 0.0, 0.0], [0.28571428571
42857, 0.2857142857142857, 0.0, 0.0, 0.42857142857142855, 0.0], [0.18, 0.26
666666666666666, 0.02, 0.006666666666666667, 0.44666666666666666, 0.08], [0
.06666666666666667, 0.6, 0.13333333333333333, 0.0, 0.2, 0.0]]
```

In [ ]:

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

```
[[0.74297753 0.18539326 0.         0.00842697 0.06320225 0.        ]
 [0.27706422 0.62201835 0.03669725 0.         0.0587156  0.00550459]
 [0.03225806 0.77419355 0.19354839 0.         0.         0.        ]
 [0.28571429 0.28571429 0.         0.         0.42857143 0.        ]
 [0.18       0.26666667 0.02       0.00666667 0.44666667 0.08      ]
 [0.06666667 0.6        0.13333333 0.         0.2        0.        ]]
```

In [ ]:

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

Out[ ]:

	index	datetime	condition
0	7671	2021-01-01	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   7671  2021-01-01              clear                  clear
1   7672  2021-01-02              clear  rain_partially_cloudy
2   7673  2021-01-03  partially_cloudy  rain_partially_cloudy
3   7674  2021-01-04  partially_cloudy               overcast
4   7675  2021-01-05  partially_cloudy       partially_cloudy
0.3698630136986301
```

In [ ]:

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

Out[ ]:

```
0.38753424657534247
```