PHASE 4- BIG DATA ANALYSIS USING IBM CLOUD COMPUTING

PROGRAM:

```
import numpy as np
import pandas as pd



import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

df = pd.read_csv('/kaggle/input/weather-prediction/seattle-weather.csv')

df.head()


df.groupby('weather').count()

df.loc[(df['weather']=='drizzle') & (df.precipitation != 0)]

df.loc[(df.weather == 'sun') & (df.precipitation)!=0]

df.loc[df['weather'] == 'snow']

df.loc[(df['weather'] == 'fog') & (df['precipitation'] != 0)]

m = len(df['weather'])

encode_weather = np.zeros(m)

for i in range(m):

    weather = -1

    if df['weather'][i] == 'sun':

        weather = 0

    if df['weather'][i] == 'rain':

        weather = 1

    if df['weather'][i] == 'snow':

        weather = 2

    if df['weather'][i] == 'drizzle':
        weather = 3
    if df['weather'][i] == 'fog':
        weather = 4
    encode_weather[i] = weather
df['Encode Weather'] = encode_weather
from datetime import datetime
arr_date = df['date']
season = np.zeros(df.shape[0])
duration = len(arr_date)
for i in range(duration):
    date = datetime.strptime(arr_date[i], '%Y-%m-%d')
    month = date.month
    if month == 12 or month == 1 or month == 2 or month == 3:
        season[i] = 0 #winter
    if month == 4 or month == 5 or month == 6:
        season[i] = 1 #spring
    if month == 7 or month == 8:
        season[i] = 2 #summer
    if month == 9 or month == 10 or month == 11:
        season[i] = 3 #autumn
df['Encode Season'] = season
max_temp = df['temp_min'].max()
min_temp = df['temp_min'].min()
max_wind = df['wind'].max()
min_wind = df['wind'].min()
print("max of temp_min:", max_temp)
print("min of temp_min:", min_temp)
print("max of wind:", max_wind)
print("min of wind:", min_wind)
count_rain_below_zero = 0
count_snow_above_zero = 0
m = df.shape[0]
count_snow = 0
count_rain = 0
for i in range(m):
    if df['weather'][i] == 'snow':
        count_snow += 1
    if df['weather'][i] == 'rain':
        count_rain += 1
for i in range(m):
    if df['temp_min'][i] <= 0.0:
        if df['weather'][i] == 'rain':
            count_rain_below_zero += 1
            print("Min temp when it rains:", df['temp_min'][i])
    else:
        if df['weather'][i] == 'snow':
            print("Min temp when it snows above zero:", df['temp_min'][i])
            count_snow_above_zero += 1

print(count_snow_above_zero)
print(count_rain_below_zero)
print(count_snow)
print(count_rain)
m = df.shape[0]
```

```
encode_temp = np.zeros(m)
count_very_cold = 0
count_cold = 0
count_cool = 0
count_warm = 0
for i in range(m):
    if df['temp_min'][i] <= 2.8:
        encode_temp[i] = 0
        count_very_cold += 1
    if df['temp_min'][i] > 2.8 and df['temp_min'][i] <=10.0:
        encode_temp[i] = 1
        count_cold += 1
    if df['temp_min'][i] > 10 and df['temp_min'][i] <= 15:
        encode_temp[i] = 2
        count_cool += 1
    if df['temp_min'][i] > 15:
        encode_temp[i] = 3
        count_warm += 1
df['Encode Temp'] = encode_temp
print(count_very_cold)
print(count_cold)
print(count_cool)
print(count_warm)
```

```
encode_pre = np.zeros(m)
none_pre = 0
pre = 0
for i in range(m):
    pre_value = df['precipitation'][i]
    if pre_value == 0.0:
        encode_pre[i] = 0
        none_pre += 1
    else:
        encode_pre[i] = 1
        pre += 1
df['Encode Precipitation'] = encode_pre
print(none_pre)
print(pre)
```

```
encode_wind = np.zeros(m)
wind_1 = 0
wind_2 = 0
wind_3 = 0
wind_4 = 0
for i in range(m):
    wind_value = df['wind'][i]
    if wind_value < 2.5:
        encode_wind[i] = 0
        wind_1 += 1
    if wind_value >= 2.5 and wind_value < 5.0:
        encode_wind[i] = 1
        wind_2 += 1
    if wind_value >= 5.0 and wind_value < 7.5:
        encode_wind[i] = 2
        wind_3 += 1
    if wind_value >= 7.5:
        encode_wind[i] = 3
        wind_4 += 1
df['Encode Wind'] = encode_wind
print(wind_1)
print(wind_2)
print(wind_3)
print(wind_4)
```

```
df.head()
```

```
train_size = int(df.shape[0]*0.8)
train_data = df[:train_size]
test_data = df[train_size:]
```

```
def augment_features(df):
    return pd.DataFrame({'Season':df['Encode Season'],'Temp':df['Encode
Temp'],'Precipitation':df['Encode Precipitation'],'Wind':df['Encode
Wind'],'Weather': df['Encode Weather']})
```

```
def extract_features(df):
    return np.column_stack((df['Season'],df['Temp'], df['Precipitation'],
df['Wind'], df['Weather']))
```

```
features = extract_features(augment_features(train_data))
features
```

```
!pip install hmmlearn
```

```
from hmmlearn.hmm import GaussianHMM
```

```
model  = GaussianHMM(n_components = 5,random_state = 123
```

```
model.fit(features)
```

```
test_data.reset_index(inplace=True, drop=True)
test_data.shape
```

```
num_previous_days = 2
```

```
test_data['Encode Precipitation'].unique()
```

```
from tqdm import tqdm
```

```
sample_weather = np.linspace(0.0,4.0,5)
test_size = test_data.shape[0]
predicted_weather = []
for i in tqdm(range(test_size)):
    previous_days_start_index = max(0, i-num_previous_days)
    previous_days_end_index = max(0, i)
    previous_days =
extract_features(augment_features(test_data.iloc[previous_days_start_index
:previous_days_end_index]))

    likelihood_scores = []
    for weather in sample_weather:
        current_day = [test_data['Encode Season'][i], test_data['Encode
Temp'][i], test_data['Encode Precipitation'][i], test_data['Encode
Wind'][i],weather]
        sequence = np.row_stack((previous_days, current_day))
        likelihood_scores.append(model.score(sequence))
    most_probable_weather = sample_weather[np.argmax(likelihood_scores)]
    predicted_weather.append(most_probable_weather)

print(predicted_weather)
```

```
count = 0
for i in range(test_size):
    if test_data['Encode Weather'][i] == predicted_weather[i]:
        count += 1
print("accuracy:", float(count/test_size))
```

```
def score(predicted_weather):
    count = 0
    for i in range(test_size):
        if test_data['Encode Weather'][i] == predicted_weather[i]:
            count += 1
    return float(count/test_size)
```

```
import matplotlib.pyplot as plt

n_component_values = [1,2,3,4,5,6,7,8,9,10]
arr_previous_days = [1,2,3,4,5,6,7]

for n in n_component_values:
    tmp_model = GaussianHMM(n_components = n, random_state = 123)
    tmp_model.fit(features)
    for tmp_num_previous_days in arr_previous_days:
        sample_weather = np.linspace(0.0,4.0,5)
        test_size = test_data.shape[0]
        tmp_predicted_weather = []
        for i in range(test_size):
            previous_days_start_index = max(0, i-tmp_num_previous_days)
            previous_days_end_index = max(0, i)
            previous_days =
extract_features(augment_features(test_data.iloc[previous_days_start_index
:previous_days_end_index]))

            likelihood_scores = []
            for weather in sample_weather:
                current_day = [test_data['Encode Season'][i],
test_data['Encode Temp'][i], test_data['Encode Precipitation'][i],
test_data['Encode Wind'][i],weather]
                sequence = np.row_stack((previous_days, current_day))
                likelihood_scores.append(tmp_model.score(sequence))
            most_probable_weather =
sample_weather[np.argmax(likelihood_scores)]
            tmp_predicted_weather.append(most_probable_weather)
        print("(n_components, num_previous_days, score)", n,

tmp_num_previous_days,score(tmp_predicted_weather))
```