Moloco_Takehome

April 16, 2020

Moloco Takehome Exam
1.Read Data

```
In [1]: import numpy as np
        import pandas as pd
        import warnings
        warnings.filterwarnings("ignore")
        User_data = pd.read_excel("Adops & Data Scientist Sample Data.xlsx", sheet_name="Q1 Ana
```

2.Transform timestamps

```
In [2]: import time
        import datetime
        reformed = []
        for date in User_data["ts"]:
            if isinstance(date,str):
                date = time.strptime(date, "%Y-%m-%d %H:%M:%S")
                reformed.append(date)
        User_data["ts"]=pd.Series(reformed)
```

3. Answer to Part 1 questions

   i. Site with most unique users
   Consider only the rows with country_id = "BDV" (there are 844 such rows). For each site_id, we can compute the number of unique user_id's found in these 844 rows. Which site_id has the largest number of unique users?

```
In [3]: bdv_data = User_data[User_data["country_id"]=="BDV"]
        bdv_data.groupby("site_id").count().sort_values("user_id", ascending=False)
```

```
Out[3]:           ts  user_id  country_id
        site_id
        5NPAU    717      717         717
        NOOTG    122      122         122
        3POLC      5        5           5
```

The answer is "5NPAU" with 717 unique users.

ii. Four users & which sites they (each) visited more than 10 times
Between 2019-02-03 00:00:00 and 2019-02-04 23:59:59, there are four users who visited a certain site more than 10 times. Find these four users & which sites they (each) visited more than 10 times. (Simply provides four triples in the form (user_id, site_id, number of visits) in the box below.)

```
In [4]: start = time.strptime("2019-02-03 00:00:00", "%Y-%m-%d %H:%M:%S")
        end = time.strptime("2019-02-04 23:59:59", "%Y-%m-%d %H:%M:%S")
        time_window = User_data[(User_data["ts"]>=start) & (User_data["ts"]<=end)]
        grouped = time_window.groupby(["user_id","site_id"]).count()
        grouped[grouped["ts"]>=10]
```

```
Out[4]:                 ts  country_id
        user_id site_id
        LC06C3  N0OTG    25          25
        LC3A59  N0OTG    26          26
        LC3C7E  3POLC    15          15
        LC3C9D  N0OTG    17          17
```

The answer is ("LC06C3", "N0OTG",25) , ("LC3A59", "N0OTG",26), ("LC3C7E", "3POLC",15), ("LC3C9D", "N0OTG",17)

iii. Top three last visit sites
For each site, compute the unique number of users whose last visit (found in the original data set) was to that site. For instance, user "LC3561"'s last visit is to "N0OTG" based on timestamp data. Based on this measure, what are top three sites? (hint: site "3POLC" is ranked at 5th with 28 users whose last visit in the data set was to 3POLC; simply provide three pairs in the form (site_id, number of users).)

```
In [5]: user_grouped = User_data.groupby("user_id")
```

```
In [6]: def getFirstVisit(user,group):
            group = group.sort_values("ts",ascending=True)
            return group.iloc[0,3]
        def getLastVisit(user,group):
            group = group.sort_values("ts",ascending=False)
            return group.iloc[0,3]
```

```
In [7]: first_sites = []
        last_sites = []
        users = []
        for user, group in user_grouped:
            users.append(user)
            first_sites.append(getFirstVisit(user,group))
            last_sites.append(getLastVisit(user,group))
        result = pd.DataFrame([pd.Series(users),pd.Series(first_sites),pd.Series(last_sites)],
```

```
In [8]: result.groupby("last visit site").count().sort_values("users",ascending=False)
```

```
Out[8]:                  users  first visit site
        last visit site
        5NPAU              992               992
        N0OTG              561               561
        QGO3G              289               289
        GVOFK               42                42
        3POLC               28                28
        RT9Z6                2                 2
        EUZ/Q                1                 1
        JSUUP                1                 1
```

The answer is ("5NPAU", 992), ("N0OTG", 561), ("QGO3G", 289)

iv. The number of users whose first/last visits are to the same website
For each user, determine the first site he/she visited and the last site he/she visited based on the timestamp data. Compute the number of users whose first/last visits are to the same website. What is the number?

```
In [9]: result[result["first visit site"]==result["last visit site"]].shape[0]
```

```
Out[9]: 1670
```

4. Part 2 Regression model

The data contains 300 rows and 3 columns (from the left, A, B, and C). Please build a good regression model which explains column C by a function of A and B.

Since column A, B and C are all numerical values, I used linear regresion with its loss fuction as MSE here.

```
In [10]: import numpy as np
         import pandas as pd
         import scipy
         from scipy import optimize
         import warnings
         warnings.filterwarnings("ignore")
         data = pd.read_excel("Adops & Data Scientist Sample Data.xlsx", sheet_name="Q2 Regres
         data.columns = ["A","B","C"]
         data.shape
```

```
Out[10]: (300, 3)
```

i. Detect Null values and outliers.
There's no Null values in our dataset.
My criterion for detecting outlier (high leverage point) is: outside the interval of mean ś (2.33*standard deviation).
If we assume all the 3 variables has enough samples(Central Limit Theorem), 1% of the samples of each variable would be considered as outliers. The threshold could also be adjusted.

```
In [11]: data = data.dropna()
```

```
In [12]: def outlierDetector(dataseries):
             mean = dataseries.mean()
             std = dataseries.std()
             lower_bond = mean - (2.33*std)
             upper_bond = mean + (2.33*std)
             outliers = []
             for record in dataseries:
                 if record < lower_bond or record > upper_bond:
                     outliers.append(record)
             return outliers

In [13]: data = data[~data["A"].isin(outlierDetector(data["A"])) & ~data["B"].isin(outlierDete
         data.shape

Out[13]: (297, 3)
```

ii. Do the linear regression
We have our dependant variable with 2 independant variables, the result of our regression should looks like a plane in the space.

```
In [14]: def MSE(b):
             x0 = np.array(data["A"])
             x1 = np.array(data["B"])
             y =  np.array(data["C"])
             pred =b[0] + b[1]*x0 + b[2]*x1
             mse = sum(list(map(lambda x: x**2, y - pred)))
             return mse

In [15]: result = scipy.optimize.minimize(MSE, [1,1,1])
         result.x

Out[15]: array([ 25.03925411,  -1.09164429, -15.0198942 ])

In [16]: print("The relationship between A, B and C is: C = % 5.2f + (% 5.2f * A) + (% 5.2f * B

The relationship between A, B and C is: C =  25.04 + (-1.09 * A) + (-15.02 * B)
```

iii. Visualization

```
In [17]: X, Y = np.meshgrid(data["A"], data["B"])
         Z = result.x[0] + result.x[1]*X +result.x[2]*Y

In [18]: import matplotlib.pyplot as plt
         from mpl_toolkits.mplot3d import Axes3D
         %matplotlib inline
         fig = plt.figure(figsize=[10,10])
         ax = Axes3D(fig)
         ax.scatter(data["A"],data["B"],data["C"],c="y")
         ax.plot_surface(X,Y,Z, cmap="Blues")
```

Out[18]: <mpl_toolkits.mplot3d.art3d.Poly3DCollection at 0x1175d3278>