PROJECT - 7

PYTHON: HEALTH CARE CASE STUDY - CLOUDERA CHALLENGE

ABSTRACT:

First introduced in 2013, the Cloudera Data Science Challenge is a rigorous competition in which candidates must provide a solution to a real-world big data problem that surpasses a benchmark specified by some of the world's elite data scientists.

The Cloudera Data Science Challenge 2 (in 2014) involved detecting anomalies in the United States Medicare insurance system. Finding anomalous patients, procedures, providers, and regions in the competition's large, complex, and intertwined data sets required industrial-strength tools for data wrangling and machine learning. This paper shows how I did it with python.

SUMMARIES OF PROBLEM, DATA, METHODS, AND TECHNOLOGIES:

❖ PROBLEM SUMMARY

The Challenge was divided into the following three parts, each of which had specific requirements that pertained to identifying anomalous entities in different aspects of the Medicare system:

- **Part 1**: Identify providers that overcharge for certain procedures or regions where procedures are too expensive.
- **Part 2**: Identify the three providers that are least similar to other providers and the three regions that are least similar to other regions.
- **Part 3**: Identify 10,000 Medicare patients who are involved in anomalous activities.

The Challenge rules mandated that solutions for each part be packaged into specific deliverables and submitted to Cloudera by a specified deadline.

❖ DATA SUMMARY

Parts 1 and 2 were based on financial summary data from 2011 that were made available by the Centers for Medicare and Medicaid Services (CMS) in both comma-separated value (CSV) format and Microsoft Excel format

- Medicare_Charge_Inpatient_DRG100_DRG_Summary_by_DRG_FY2011.csv
- Medicare_Charge_Outpatient_APC30_Summary_by_APC_CY2011.csv
- Medicare_Provider_Charge_Inpatient_DRG100_FY2011.csv
- Medicare_Provider_Charge_Outpatient_APC30_CY2011_v2.csv

Part 3 of the Challenge contain patient demographic information and patient-procedure transaction information

- Patient_history_samp.csv
- Review_patient_history_samp.csv
- Rreview_transaction_coo.csv
- Transaction_coo.csv

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

❖ METHODS SUMMARY

Table shows the wide variety of data pre-processing, analysis, and visualization techniques that I applied to complete the tasks as part of the project –

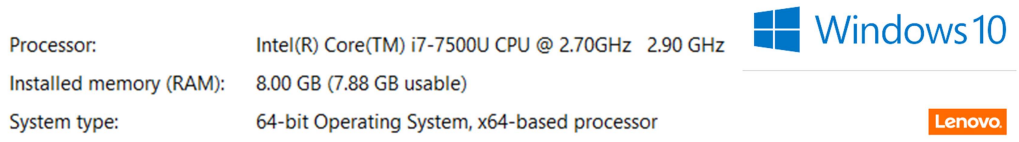
PART	Analytical Techniques	Visualization Techniques
1	Descriptive statistics Straightforward data manipulation	Box plots Pie charts Scatter plots Heat map
2	Data manipulation K MEANS Clustering Euclidean distances	
3	Data manipulation SMOTE Random Forest K MEANS Clustering	

❖ TECHNOLOGIES SUMMARY

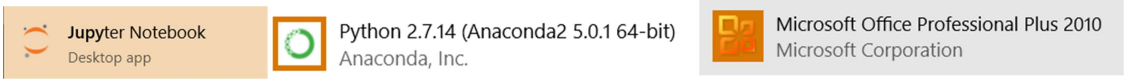
The size and variability of data possess lots of challenge in this project.

The following list summarizes the technology that I used:

Computing platforms:



Microsoft Excel 2010 on the single-machine platform
Jupyter Notebook on the single-machine platform
Python 2.7.14 (Anaconda2 5.0.1 64bit) on the single-machine platform



Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

PART ONE: IDENTIFY PROVIDER AND REGIONS WHERE COSTS ARE HIGH

❖ METHODS SUMMARY

The data provided was mostly clean for analysis; there were not much data cleaning required.

The data was provided in csv format and was imported to python using "read_csv" method of pandas.

```
Medicare_Provider_Charge_In =
pd.read_csv('Medicare_Provider_Charge_Inpatient_DRG100_FY2011.csv')

Medicare_Provider_Charge_Out =
pd.read_csv('Medicare_Provider_Charge_Outpatient_APC30_CY2011_v2.csv')
```

The data frame for inpatient was expressed in terms of 'DRG Definition' and for outpatient; it was expressed in terms of 'APC'

```
Medicare_Provider_Charge_In.isnull().any()
```

```
DRG Definition                                False
Provider Id                                   False
Provider Name                                 False
Provider Street Address                       False
Provider City                                 False
Provider State                                False
Provider Zip Code                             False
Hospital Referral Region (HRR) Description    False
Total Discharges                              False
Average Covered Charges                       False
Average Total Payments                        False
Average Medicare Payments                     False
dtype: bool
```

```
Medicare_Provider_Charge_Out.isnull().any()
```

```
APC                                           False
Provider Id                                   False
Provider Name                                 False
Provider Street Address                       False
Provider City                                 False
Provider State                                False
Provider Zip Code                             False
Hospital Referral Region (HRR) Description    False
Outpatient Services                           False
Average  Estimated Submitted Charges          False
Average Total Payments                        False
dtype: bool
```

The **PART 1** problem seems straight forward; I used simple data manipulation and statistics to get to the desired outcome. Following steps have been repeated for both **Inpatient** and **Outpatient** data.

For **PART1A** - For cost variation calculation

Step 1- Finding Standard deviation and mean for InpatientDRG/OutpatientAPC
Step 2 - Coefficient of Variance, the standard deviation divided by the mean

$$c_v = \frac{\sigma}{\mu}$$

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

For **PART1B** - Highest Cost Claims by provider

Step 1 - Finding the max of Average Covered Charges for InpatientDRG/OutpatientAPC

Step 2 - Merging the max value with InpatientDRG/OutpatientAPC dataset

Step 3 – Aggregating the data at provider level to get number of times a provider has charge max value for procedure

For **PART1C** - Highest Cost Claims by Region

Step 1 - Finding the mean of Average Covered Charges for InpatientDRG/OutpatientAPC by DRG/APC Definition and Region

Step2 – Getting the max by region for each DRG/APC Definition

Step3 - Getting number of times a region has charge max value for procedure

For **Part 1D**: Highest Number of Procedures and Largest Differences between Claims and Reimbursements

Step 1 – Get the claim difference by (Charge – Payment) for each provider and DRG/APC Definition

Step2 – Getting the max by DRG/APC Definition

Step3 – Join the data frame and count how many times a provider has hit MAX.

- **RESULTS**
  - **Part 1A: Highest Cost Variation**

```
0604 - Level 1 Hospital Clinic Visits
0698 - Level II Eye Tests & Treatments
0019 - Level I Excision/ Biopsy
```

  - **Parts 1B : Highest-Cost Claims by Provider**

```
BAYONNE HOSPITAL CENTER       - NJ - Newark
CROZER CHESTER MEDICAL CENTER - PA - Philadelphia
STANFORD HOSPITAL             - CA - San Mateo County
```

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

- **Part 1C: Highest-Cost Claims by Region**

```
Hospital Referral Region (HRR) Description
CA - Contra Costa County
CA - San Mateo County
CA - Santa Cruz
```

- **Part 1D: Highest Number of Procedures and Largest Differences between Claims and Reimbursements**

```
BAYONNE HOSPITAL CENTER, NJ          29
CROZER CHESTER MEDICAL CENTER, PA    12
HAHNEMANN UNIVERSITY HOSPITAL, PA     8
```

Solution: solution**Clouera_sol1_for _PART1.ipynb**

PART TWO: IDENTIFY THE LEAST SIMILAR PROVIDERS AND REGIONS

- **METHODS SUMMARY:**

The purpose of this part is to identify the **three providers** that are least similar to other providers and the **three regions** that are least similar to other region.

Based on the profiling report and cardinality of data following columns will be used for processing –

Columns to be used to process inpatient data-

'DRG Definition', 'Provider Name', 'Provider State', 'Hospital Referral Region (HRR) Description', 'Total Discharges', 'Average Covered Charges' and 'Average Total Payments'

Columns to be used to process outpatient data -

'APC', 'Provider Name', 'Provider State', 'Hospital Referral Region (HRR) Description', 'Outpatient Services', 'Average Estimated Submitted Charges' and 'Average Total Payments'

Sub-setting & renaming columns so that we have identical columns for Inpatient and Outpatient as:-

```
Procedures
Provider Name
Provider State
Region
Count Of Services
Charges
Payment
```

This was a cluster analysis problem; but multiple regression needs to be performed to get the 3 odds for providers and region:-

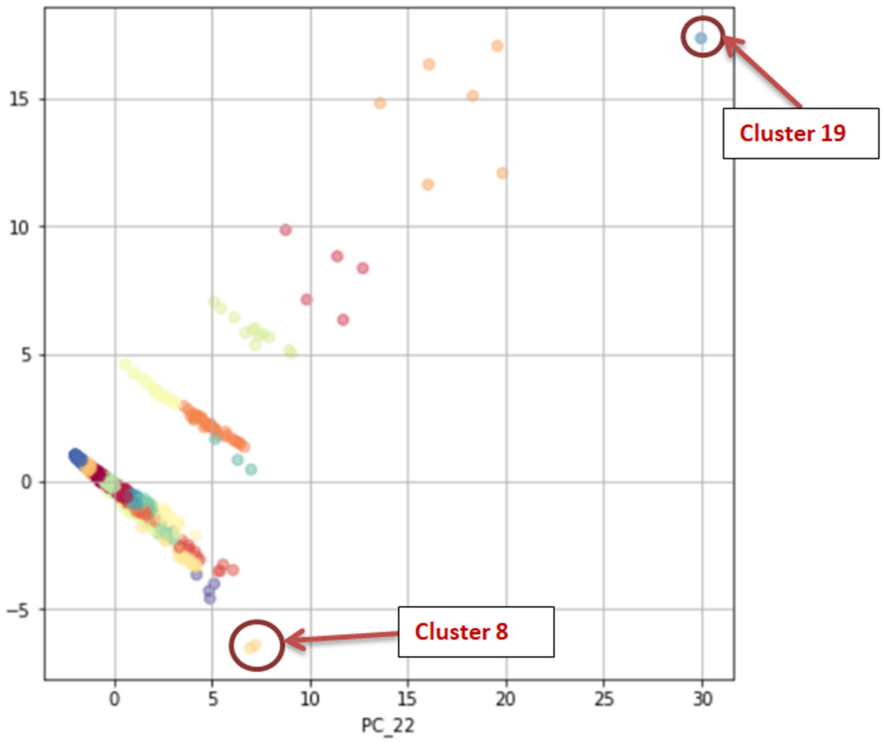
a. Before putting data for cluster analysis I scaled the data and applied PCA to get optimal components.
b. Using Silhouette Coefficient – optimal cluster.

*My purpose was to create **most scattered cluster** to get the odd one out; so I used cluster having least Silhouette Coefficient.*

c. Based on **K-MEANS** cluster analysis we can see clusters with low population as outliers –

(Same process was applied for provider and region)

From the scatter plot we can clearly **see cluster 19 with 1 dot** and **cluster 8 with 2 dots** are outliers

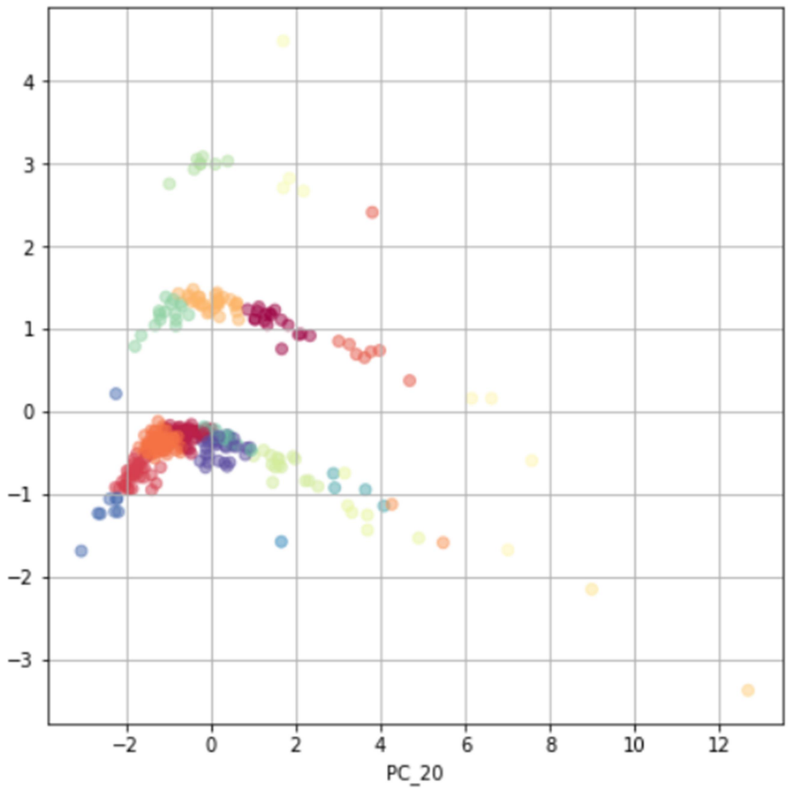


Least like clusters were identified by provider

Provider Name	Count Of Services	Provider State	Charges	Region	Procedures	Payment	Cluster_22
CLEVELAND CLINIC	377234	1	4653233.86	1	123	1198120.16	8
GOOD SAMARITAN HOSPITAL	69164	6	28175444.35	9	124	6991828.73	19
SCOTT & WHITE MEMORIAL HOSPITAL	402799	1	2823332.26	1	122	1061028.55	8

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

We see cluster **7, 8 and 17** has only one region each making it least similar to other regions



Least Like clusters were identified by region

Region	Count Of Servic	Provider State	Charges	Provider Nar	Procedur	Payment	Cluster_20
CA - Los Angeles	505020	1	251071326	80	130	47115949.39	7
MA - Boston	1494212	1	61870396.92	41	130	30781725.53	8
MD - Baltimore	94192	1	25449498.82	23	100	23992800.17	17

Solutions: - solution/**Clouera_sol1_for _PART2.ipynb**

Output of cluster analysis - output/

- PART2_PROVIDER_CLUSTER.xlsx
- PART2_REGION_CLUSTER.xlsx

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

PART THREE: IDENTIFY PATIENTS INVOLVED IN ANOMALOUS ACTIVITIES

❖ DATA PRE-PROCESSING AND DATA MODELLING:

This is the most critical of all parts; since we are working with imbalanced data.

For this problem the dataset used were:

- Patient_history_samp.csv
- Review_patient_history_samp.csv
- Rreview_transaction_coo.csv
- Transaction_coo.csv

Generally for anomaly detection, we aim to learn what is most representative of 'normal', and then anything outside of some threshold can be considered an anomaly.

Review_patient_history_samp.csv and **Rreview_transaction_coo.csv** datasets are the details for reviewed patient.

Patient_history_samp.csv and **Transaction_coo.csv** contained **unclassified/unmarked** patient.

We have to identify the cause/pattern for the review patient and based on that the unmarked patients' needs to be marked.

This is a highly imbalanced data; since only **5000** participants were marked for review; while unmarked participant count is **500000**

#Step1: As a part of data analysis I transformed categorical variables to numerical variables:

(Code snippet)

```
#function to get the income group indicator
def incometyp(patient):
    if (patient['income']  == '16000-23999'):
        return 2
    elif (patient['income']  == '24000-31999'):
        return 3
    elif (patient['income']  == '32000-47999'):
        return 4
    elif (patient['income']  == '48000+'):
        return 5
    else:
        return 1
```

```
#function to get the age group indicator
def agetyp(patient):
    if (patient['age']  == '65-74'):
        return 2
    elif (patient['age']  == '75-84'):
        return 3
    elif (patient['age']  == '85+'):
        return 4
    else:
        return 1
```

```
#function to get the gender group indicator
def gendertyp(patient):
    if (patient['gender']  == 'M'):
        return 1
    else:
        return 0
```

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

#Step2: Then I created dummies for each proc and counted by patient id for both review patient and unlabelled transaction data.

Added flag '**REVIEW_IND**' to both data frame and merge.

Code Snippet

```
Transaction_grouped = pd.DataFrame(Transaction_cat[['id', 'count','proc_1' ,'proc_2' ,'proc_3' ,'proc_4' ,'proc_5' ,
                                                    'proc_6' ,'proc_7' ,'proc_8' ,'proc_9' ,'proc_10' ,'proc_11' ,
                                                    'proc_12' ,'proc_13' ,'proc_14' ,'proc_15' ,'proc_16' ,'proc_17' ,
                                                    'proc_18' ,'proc_19' ,'proc_20' ,'proc_21' ,'proc_22' ,'proc_23' ,
                                                    'proc_24' ,'proc_25' ,'proc_26' ,'proc_27' ,'proc_28' ,'proc_29' ,
                                                    'proc_30' ,'proc_31' ,'proc_32' ,'proc_33' ,'proc_34' ,'proc_35' ,
                                                    'proc_36' ,'proc_37' ,'proc_38' ,'proc_39' ,'proc_40' ,'proc_41' ,
                                                    'proc_42' ,'proc_43' ,'proc_44' ,'proc_45' ,'proc_46' ,'proc_47' ,
                                                    'proc_48' ,'proc_49' ,'proc_50' ,'proc_51' ,'proc_52' ,'proc_53' ,
                                                    'proc_54' ,'proc_55' ,'proc_56' ,'proc_57' ,'proc_58' ,'proc_59' ,
                                                    'proc_60' ,'proc_61' ,'proc_62' ,'proc_63' ,'proc_64' ,'proc_65' ,
                                                    'proc_66' ,'proc_67' ,'proc_68' ,'proc_69' ,'proc_70' ,'proc_71' ,
                                                    'proc_72' ,'proc_73' ,'proc_74' ,'proc_75' ,'proc_76' ,'proc_77' ,
                                                    'proc_78' ,'proc_79' ,'proc_80' ,'proc_81' ,'proc_82' ,'proc_83' ,
                                                    'proc_84' ,'proc_85' ,'proc_86' ,'proc_87' ,'proc_88' ,'proc_89' ,
                                                    'proc_90' ,'proc_91' ,'proc_92' ,'proc_93' ,'proc_94' ,'proc_95' ,
                                                    'proc_96' ,'proc_97' ,'proc_98' ,'proc_99' ,'proc_100' ,'proc_101' ,
                                                    'proc_102' ,'proc_103' ,'proc_104' ,'proc_105' ,'proc_106' ,
                                                    'proc_107' ,'proc_108' ,'proc_109' ,'proc_110' ,'proc_111' ,
                                                    'proc_112' ,'proc_113' ,'proc_114' ,'proc_115' ,'proc_116' ,
                                                    'proc_117' ,'proc_118' ,'proc_119' ,'proc_120' ,'proc_121' ,
                                                    'proc_122' ,'proc_123' ,'proc_124' ,'proc_125' ,'proc_126' ,
                                                    'proc_127' ,'proc_128' ,'proc_129' ,'proc_130']].groupby(['id']).agg
                                                                                        'proc_1' :'sum' ,
                                                                                        'proc_2' :'sum' ,
                                                                                        'proc_3' :'sum' ,
                                                                                        'proc_4' :'sum' ,
                                                                                        'proc_5' :'sum' ,
                                                                                        'proc_6' :'sum' ,
                                                                                        'proc_7' :'sum' ,
                                                                                        'proc_8' :'sum' ,
                                                                                        'proc_9' :'sum' ,
                                                                                        'proc_10' :'sum' ,
                                                                                        'proc_11' :'sum' ,
                                                                                        'proc_12' :'sum' ,
                                                                                        'proc_13' :'sum' ,
                                                                                        'proc_14' :'sum' ,
                                                                                        'proc_15' :'sum' ,
                                                                                        'proc_16' :'sum' ,
                                                                                        'proc_17' :'sum' ,
```

Thus we have 2 datasets; unmarked dataset (**full dataset**) and marked for review (**review dataset**)

```
full_dataset['id'].count()
```

```
500000
```

```
review_dataset['id'].count()
```

```
5000
```

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

❖ **METHODS TO GET THE FINAL 10K**

The major hurdle was we had 500000 unlabelled and 5000 review labelled patients, which is a highly imbalanced data.

So I took a sample of full dataset and added 5000 review labelled patients to create the train dataset.

Then I used **SMOTE** to oversample the train dataset to make it a balanced data.

Model was **fit** on the oversampled train data and tested on test data(unmarked datset)

Code Snippet -

```
sample_dataset = full_dataset.sample(frac=0.1)
```

```
frames = [sample_dataset, review_dataset]
train_df = pd.concat(frames)
```

```
cols = train_df.columns.difference( ['REVIEW_IND', 'id'] )
```

```
df_train = train_df[cols ]
df_test = full_dataset[cols ]
```

```
from sklearn.cross_validation import train_test_split


train_X, test_X, train_y, test_y = train_test_split( train_df[cols ],
                                                     train_df['REVIEW_IND'],
                                                     test_size = 0.2,
                                                     random_state = 42 )
```

```
    Overfitting the data using SMOTE
```

```
from imblearn.over_sampling import SMOTE
sm = SMOTE(random_state=123, ratio = 0.5)
x_train_res, y_train_res = sm.fit_sample(train_X,  train_y)
```

I performed association analysis for getting "**REVIEW probability**" based on train dataset – I used **Random Forest** for the purpose.

I got **99**% Accuracy on train dataset (only 46 out of 20006 were misclassified where REVIEW_IND =1)

I used the model on **test dataset** to add the **prediction probability** and predicted label to each unlabelled patient.

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

13787 patient were found having prediction probability > 5 i.e. (predicted label =1).

We have our suspicious unmarked patients.

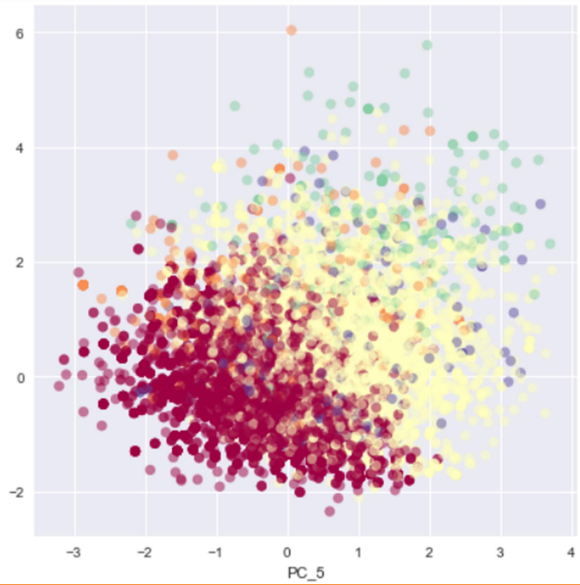
- **TESTING AND VALIDATION**

For testing the result I used cluster analysis to see if anomalous patients are clustering together.

I added **suspicious** un-labelled **13787** patients found from random forest and added the review data frame of 5000 patients.

I used PCA to get optimal components out of long list of columns for better cluster analysis.

To my amazement I found; the **10830** patients were getting clustered in **Cluster 0** for a 5 cluster solution.



Note:-

For **Cluster 0**: There are **10830** patients; among them **2091** have already been marked for review. Left **8739** patients need to be reviewed.

For **Cluster 1**: There are **7007** patients clustered; among them **2450** patients have already been marked for review. Left **4557** patients need to be reviewed.

Cluster Number	Count of id
0	10830
1	414
2	7007
3	298
4	238

ALREADY MARKED AS REVIEW	
Cluster Number	Count of id
0	2091
1	185
2	2450
3	117
4	157

NEED TO BE MARKED FOR REVIEW	
Cluster Number	Count of id
0	8739
1	229
2	4557
3	181
4	81

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90

- **RESULTS AND RECOMMENDATIONS**

Unmarked Patient IDs at **cluster 0** need to be reviewed with high priority followed by Cluster 2

Cluster 0 have **8739** and Cluster 2 have **4557** anomalous patients detected for review.

Solutions: - solution/ **Clouera_sol1_for _PART3.ipynb**

Output of cluster analysis - output/**Clouera_sol1_for _PART3.ipynb**

REFERENCES

https://github.com/

https://www.kaggle.com/

Analytixlabs classes

Email: banerjee.kaustav@outlook.com
Github: https://github.com/KBanerjee90
Linkedin: https://www.linkedin.com/in/kaustav-banerjee-584525149