Author : Part 2: Mohit Bansal (Manager, Advisory, PwC India)

Section 1: Import Data and read the file

```
#**Connect Google Colab to My google drive**
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

Adding 2 variables in the sheet from Assignment 1, formulating a **primary key with combination of Lead ID, bankid, bank account ID, Post date and transaction order to create unique record of each row** and month year column.

```
%matplotlib inline
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

bank_1 = pd.read_csv('/content/drive/MyDrive/Bank_Data/Bank_Data_v1.csv')
bank_1['month_year'] = pd.to_datetime(bank_1['post_date']).dt.to_period('M')
print(bank_1)
```

```
           Lead ID  bankid  ...                Primary_key month_year
                                08148_8535_12460_42439_1    2016-03
                                08148_8535_12460_42450_1    2016-03
                                08148_8535_12460_42450_3    2016-03
```

Saved successfully!

```
3       308148    8535  ...  308148_8535_12460_42450_2    2016-03
4       308148    8535  ...  308148_8535_12460_42451_1    2016-03
...        ...     ...  ...                        ...        ...
29024   330698    8545  ...  330698_8545_14374_42772_7    2017-02
29025   330698    8545  ...  330698_8545_14374_42773_2    2017-02
29026   330698    8545  ...  330698_8545_14374_42773_4    2017-02
29027   330698    8545  ...  330698_8545_14374_42773_3    2017-02
29028   330698    8545  ...  330698_8545_14374_42773_1    2017-02

[29029 rows x 13 columns]
```

```
pip install kneed
```

```
Collecting kneed
  Downloading https://files.pythonhosted.org/packages/c3/6b/e130913aaaad1373060e259ab22
Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: numpy>=1.14.2 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from kn
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packa
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/l
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-pa
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from cycl
Installing collected packages: kneed
Successfully installed kneed-0.7.0
```

```
bank_1.head(2)
```

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	descripti
0	308148	8535	12460	xxxx9928	Accommodation and Food Services	10-Mar-16	DEPOSIT NUMB xx68
					Accommodation		ATM CA DEPO

```
bank_3 = pd.read_csv('/content/drive/MyDrive/Bank_Data/Bank_data_v2.csv')
bank_3.tail(2)
```

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	descrip
29027	330698	8545	14374	1693	Health Care and Social Assistance	07-Feb-17	Withdr NEW LC (PREAUTH

Saved successfully!

merchants

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler


from sklearn.decomposition import PCA
from scipy.stats import zscore


sns.pairplot(bank_3, diag_kind='kde')
```

```
<seaborn.axisgrid.PairGrid at 0x7f5876e7fdd8>
```

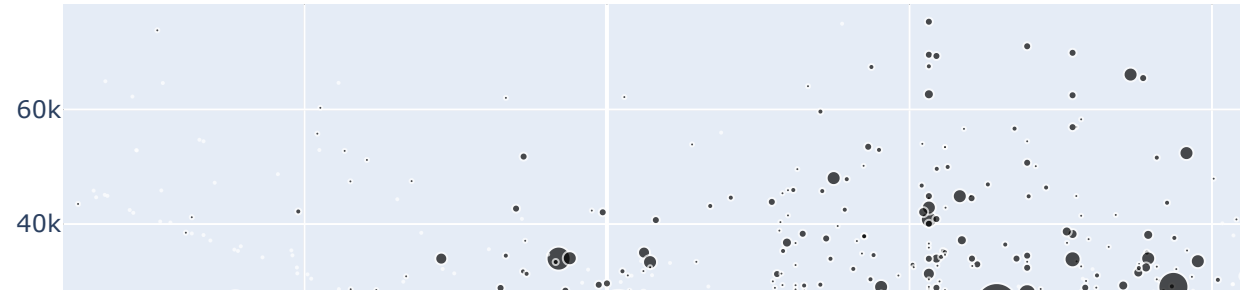
5a Sorted and Filtered for ordering the transactions for clustering patterns.

Saved successfully!

	Lead ID	bankid	bank_account_id	amount	running_balance	tr
count	29029.000000	29029.000000	29029.000000	29029.000000	29029.000000	290
mean	320714.718282	8823.888491	14052.030556	1927.453475	10247.851975	
std	5051.317064	683.963276	703.164708	7450.638452	20055.683105	
min	308148.000000	6192.000000	12460.000000	0.000000	-84727.980000	
max	330698.000000	10656.000000	15148.000000	419000.000000	437942.290000	

5b Plotting the data for amount vs Running Balance to analyse and understand the pattern of payments

```
# Understand the pattern for a particular merchant 313082
# bubble size - transaction no for the particular day
# x axis - month year
# y ais - Amount

import plotly.graph_objects as go

fig = go.Figure(data=[go.Scatter(
    x=ML_data3['running_balance'], y= ML_data3['amount'],
    mode='markers',
    marker=dict(
        size = ML_data3['trans_order'],
        color=['rgb(9, 164, 214)', 'rgb(255, 144, 14)',
               'rgb(44, 131, 101)', 'rgb(255, 65, 54)'],

    )
)])


fig.show()
```



5c Developing a basic cluster model initially using the numerical data along with Merchant and Bank IDs. Using **k-means clustering with ELBOW Method** to study the number of clusters

```
bank_3['trans_type'] = bank_3['transaction_type'].apply(lambda x: 1 if x == 'credit' else 2)

ML_data3 = ML_data3.iloc[range(0, len(ML_data3), 1),[0,8,10,13,14,15]]
ML_data3.head(4)
```

	Lead ID	amount	trans_order	year	month	trans_type
113	308148	113.02	2	2016	8	1
114	308148	229.09	5	2016	8	1
115	308148	360.75	4	2016	8	1
116	308148	10.22	1	2016	8	2

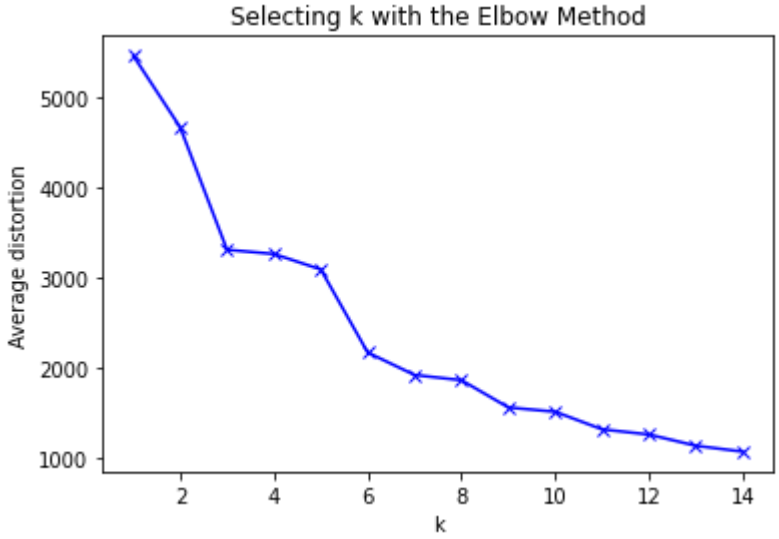
```
from sklearn.model_selection  import train_test_split
from sklearn.cluster import KMeans
#import sklearn.metrics

#Finding optimal no. of clusters

from scipy.spatial.distance import cdist

ML_data3a = ML_data3
clusters=range(1,15)
```

Saved successfully!

```
    model=KMeans(n_clusters=k)
    model.fit(ML_data3a)
    prediction=model.predict(ML_data3a)
    meanDistortions.append(sum(np.min(cdist(ML_data3a, model.cluster_centers_, 'euclidean'),
```

```
plt.plot(clusters, meanDistortions, 'bx-')
plt.xlabel('k')
plt.ylabel('Average distortion')
plt.title('Selecting k with the Elbow Method')
```

```
Text(0.5, 1.0, 'Selecting k with the Elbow Method')
```




```
ML_data4 = bank_3.iloc[range(0, len(bank_3), 1),[4,6,7,8,11]]
ML_data4.head(4)
```

	Industry	description	transaction_type	amount	Primary_ke
113	Accommodation and Food Services	BANKCARD-8779 BTOT DEP xxxxxxxxxx4599 CCD ID:...	credit	113.02	308148_8535_12460_42583_
114	Accommodation	BANKCARD-8779 BTOT DEP	credit	220.00	308148_8535_12460_42583

5d Pre-Processing of the data for analysis on banking transaction description.

We would develop cluster on Description and transaction

***text and develop an NLP model**

Saved successfully!

using Tokenization/stop word removal, TF-IDF algorithm along with clusters to study and segment the transaction description *for daily details of merchants.

```
ML_data5 = ML_data4.replace('\n','',regex=True)
ML_data5 = ML_data5.replace(('[^A-z ]',''),regex=True)
ML_data5 = ML_data5.replace(('/',''),regex=True)
#JD_data2.rename(columns={0:'Decription'},inplace = True)
ML_data5["Combined_data"] = ML_data5["description"] + "_" + ML_data5["transaction_type"]
print(ML_data5.head(5))
```

```
                            Industry  ...                                    Combined
113  Accommodation and Food Services  ...  BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD I
114  Accommodation and Food Services  ...  BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD I
115  Accommodation and Food Services  ...  BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD I
116  Accommodation and Food Services  ...            CHICK-FIL-A #x1303 KEMAH TX07/29_
112  Accommodation and Food Services  ...            ARLANS MARKET # SEABROOK TX07/30_

[5 rows x 6 columns]
```

```
import nltk
nltk.download('stopwords')
nltk.download('punkt') # download
nltk.download('averaged_perceptron_tagger') # Download the tagger
from nltk.tokenize import punkt
import re, pprint
from nltk import word_tokenize
from nltk.corpus import stopwords

from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.decomposition import PCA
from sklearn.preprocessing import normalize
from sklearn.metrics import pairwise_distances
from nltk.stem.porter import PorterStemmer
from nltk.corpus import stopwords
import bs4
from scipy.stats import multivariate_normal as mvn
import nltk
import os
from os import listdir
from os.path import isfile, join
nltk.download('stopwords')
import matplotlib.pyplot as plt
import pandas as pd
                                    t import TfidfVectorizer
```

Saved successfully!

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Unzipping tokenizers/punkt.zip.
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping taggers/averaged_perceptron_tagger.zip.
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
```

```
def clean_column1(data):
       if data is not None:
        stopwords_list = stopwords.words('english')
        #exclusions = ['RE:', 'Re:', 're:']
        #exclusions = '|'.join(exclusions)
        data =  data.lower()
        data = re.sub('re:', '', data)
        data = re.sub('-', '', data)
        data = re.sub('_', '', data)
        # Remove data between square brackets
        data =re.sub('\[[^]]*\]', '', data)
        # removes punctuation
        data = re.sub(r'[^\w\s]','',data)
        data = re.sub(r'\n',' ',data)
        data = re.sub(r'/',' ',data)
        data = re.sub(r'[0-9]+','',data)
        # strip html
        p = re.compile(r'<.*?>')
        data = re.sub(r"\'ve", " have ", data)
        data = re.sub(r"can't", "cannot ", data)
        data = re.sub(r"n't", " not ", data)
        data = re.sub(r"I'm", "I am", data)
        data = re.sub(r" m ", " am ", data)
        data = re.sub(r"\'re", " are ", data)
        data = re.sub(r"\'d", " would ", data)
        data = re.sub(r"\'ll", " will ", data)

        data = p.sub('', data)
        if 'forwarded by:' in data:
            data = data.split('subject')[1]
        data = data.strip()
        return data
       return 'No Subject'
ML_data5["Combined_data_new"] = ML_data5["Combined_data"].apply(clean_column1)
print(ML_data5)
#ML_data5.to_excel("./CV_data_check_2.xlsx", index=False)
```

```
                                  Industry  ...                                  Combined
                                  d Services  ...   bankcard btot dep xxxxxxxxxxx ccd id xx
                                  d Services  ...   bankcard btot dep xxxxxxxxxxx ccd id xx
                                  d Services  ...   bankcard btot dep xxxxxxxxxxx ccd id xx
```

Saved successfully!

```
116      Accommodation and Food Services  ...                        chickfila x kema
112      Accommodation and Food Services  ...                 arlans market  seabroo
...                                   ...  ...
28518  Health Care and Social Assistance  ...  ach withdrawal new logic  preauthpmt per
28515  Health Care and Social Assistance  ...  ach withdrawal amex epayment ach pmt she
28517  Health Care and Social Assistance  ...  ach withdrawal hsbc bank usa na hsbc ban
28516  Health Care and Social Assistance  ...  ach withdrawal amex epayment ach pmt she
28519  Health Care and Social Assistance  ...  wire transfer withdrawal janabi associat

[29029 rows x 7 columns]
```

```
# Preprocessing and tokenizing
import string

def preprocessing(line):
    line = line.lower()
    line = re.sub(r"[{}]".format(string.punctuation), " ", line)
    return line


ML_data6 = ML_data5['Combined_data_new']
ML_data6.head(2)

tfidf_vectorizer = TfidfVectorizer(preprocessor=preprocessing)
tfidf = tfidf_vectorizer.fit_transform(ML_data6)

kmeans = KMeans(n_clusters=5).fit(tfidf)

#ML_data3 = ML_data3.iloc[range(0, len(ML_data3), 1),[0,8,10,13,14,15]]


print("Top terms per cluster:")
order_centroids = kmeans.cluster_centers_.argsort()[:, ::-1]
terms = tfidf_vectorizer.get_feature_names()
for i in range(5):
    print("Cluster %d:" % i),
    for ind in order_centroids[i, :15]:
        print(' %s' % terms[ind]),
    print
```

```
Top terms per cluster:
Cluster 0:
 desdep
 idlkxx
 trf
 worldpay
 idcredit
 xx
 co
 indncitgo
```

Saved successfully!

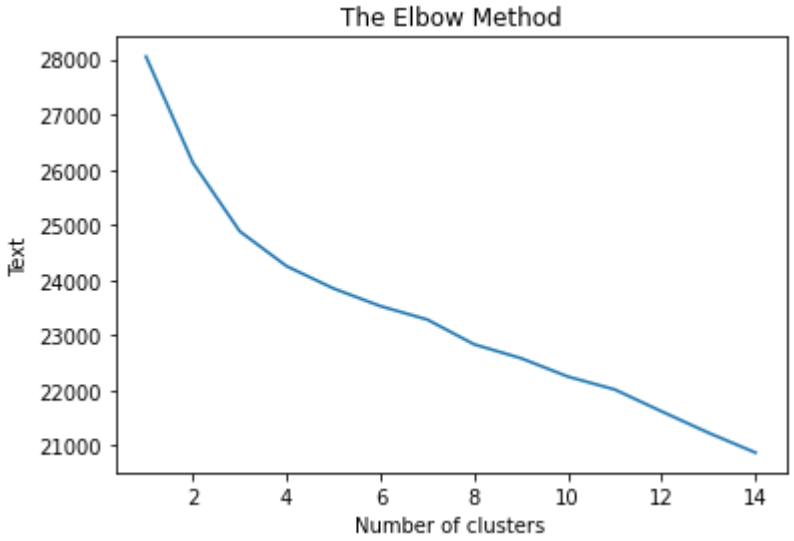
```
 party
 station
 sho
 lake
Cluster 1:
 check
 debit
 xxdebit
 xxxxxxxdebit
 cashed
 deposited
 or
 gift
 gordmans
 goodgo
 goodgobellevue
 goodgocall
 goodgoivr
 goods
 goodyear
Cluster 2:
 debit
 card
 xxxxxx
 purchase
 withdrawal
 checkdebit
 pos
 in
 xxxx
 authorized
 on
 checkcard
 xxxxxxxxxxxxxxxxxxxdebit
 co
 ach
Cluster 3:
 deposit
 id
 ccd
 dep
 xxcredit
 xxxxxxcredit
 lkxx
 bankcard
 fdmssettlement
```

5e The analysis using ***ELBOW method looks like the optimum cluster point is 4 ***observing a sudden dip from 1 to 4 and flatening comparatively.

Saved successfully!

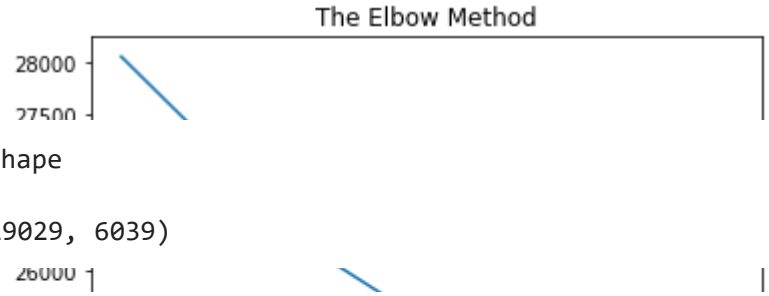
```
# Using the elbow method to find the optimal number of clusters
```

```
sse = []
for i in range(1, 15):
    kmeans_2 = KMeans(n_clusters = i, init = 'k-means++', random_state = 5)
    kmeans_2.fit(tfidf)
    sse.append(kmeans_2.inertia_)
plt.plot(range(1, 15), sse)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Text')
plt.show()
```




```
# Using the elbow method to find the optimal number of clusters
sse = []
for i in range(1, 5):
    kmeans_2 = KMeans(n_clusters = i, init = 'k-means++', random_state = 5)
    kmeans_2.fit(tfidf)
    sse.append(kmeans_2.inertia_)
plt.plot(range(1, 5), sse)
plt.title('The Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Text')
plt.show()
```

Saved successfully!




```
tfidf.shape
```

```
(29029, 6039)
```

```
# Fitting K-Means to the dataset
X=tfidf
kmeans_3 = KMeans(n_clusters = 4, init = 'k-means++', random_state = 42)
y_kmeans = kmeans_3.fit_predict(tfidf)
```

```
y_kmeans
```

```
array([0, 0, 0, ..., 0, 0, 0], dtype=int32)
```

OUTPUT of the Model: 4 Clusters with grouped banking daily transactions descriptions for merchants

```
bank_3["Cluster_ID"] = y_kmeans
#ML_data4 = ML_data4[["Lead ID","Industry","post_date","Primary_key","Combined_data_new", "Cl
bank_3.head(4)
```

ankid	bank_account_id	account_number	Industry	post_date	description	transac
8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD ID:...	
8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD ID:...	
8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BANKCARD-8779 BTOT DEP xxxxxxxxxxx4599 CCD ID:...	
8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	CHICK-FIL-A #x1303 KEMAH TX07/29	

Saved successfully!

```
]==0)]
```

```
bank_3b.head(3)
```

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	desc
113	308148	8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BAN 8779 B xxxxxxx C
114	308148	8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BAN 8779 B xxxxxxx C
115	308148	8535	12460	xxxx9928	Accommodation and Food Services	01-Aug-16	BAN 8779 B xxxxxxx C

```
bank_3b=bank_3[(bank_3['Cluster_ID']==1)]
bank_3b.head(3)
```

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	descriptic
13473	321380	8534	14636	xxxx4207	Finance and Insurance	01-Apr-16	WorldPa DES:DE TR ID:LKxx587 xx311 INDN:R
13470	321380	8534	14636	xxxx4207	Finance and Insurance	01-Apr-16	WorldPa DES:DE TR ID:LKxx114 xx311 INDN:C
13474	321380	8534	14636	xxxx4207	Finance and Insurance	01-Apr-16	WorldPa DES:DE TR ID:LKxx781 xx311 INDN:C

```
bank_3b=bank_3[(bank_3['Cluster_ID']==2)]
bank_3b.head(3)
```

Saved successfully!

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	descrip
62	308148	8535	12460	xxxx9928	Accommodation and Food Services	01-Jul-16	CHECI
120	308148	8535	12460	xxxx9928	Accommodation and Food Services	03-Aug-16	CHECI

```
bank_3b=bank_3[(bank_3['Cluster_ID']==3)]
bank_3b.head(3)
```

	Lead ID	bankid	bank_account_id	account_number	Industry	post_date	descript
1033	312745	8544	12839	3582	Professional, Scientific, and Technical Services	02-Mar-16	ONL TRANSF FR TRIVEI PREM CHECK
784	312745	8544	12835	2549	Professional, Scientific, and Technical Services	02-Mar-16	ONL TRANSF TO RANG BUSINE CHEC
804	312745	8544	12838	8423	Professional, Scientific, and Technical Services	03-Feb-16	ONL TRANSF FR RANG LLC F #IBETZVJ

```
bank_3.to_csv('/content/drive/MyDrive/Bank_Data/ML_data4.csv', index=False)
```

Saved successfully!