Problem Statement :

Perform clustering (K Means clustering and DBSCAN) for the airlines data to obtain optimum number of clusters and Compare.

Draw the inferences from the clusters obtained.

1. Import Necessory Libraries

```
from google.colab import drive
drive.mount('/content/drive')
```

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

```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import matplotlib.cm as cm
from yellowbrick.cluster import KElbowVisualizer
from sklearn.cluster import KMeans, AgglomerativeClustering, DBSCAN
from sklearn.preprocessing import MinMaxScaler, StandardScaler
from sklearn.metrics import silhouette_score, calinski_harabasz_score, silhouette_samples

import warnings
warnings.filterwarnings('ignore')
```

2. Import Dataset

```
airline_data = pd.read_excel('/content/drive/MyDrive/Colab Notebooks/EastWestAirlines.xlsx',sheet_name='data')
airline_data
```

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_tra
0	1	28143	0	1	1	1	174	
1	2	19244	0	1	1	1	215	
2	3	41354	0	1	1	1	4123	
3	4	14776	0	1	1	1	500	
4	5	97752	0	4	1	1	43300	
...	...	...	...	...	...	...	...	
3994	4017	18476	0	1	1	1	8525	
3995	4018	64385	0	1	1	1	981	
3996	4019	73597	0	3	1	1	25447	
3997	4020	54899	0	1	1	1	500	
3998	4021	3016	0	1	1	1	0	

3999 rows × 12 columns

Data Description:

- ID - Unique ID
- Balance - Number of miles eligible for award travel
- Qual_mile - Number of miles counted as qualifying for Topflight status

- cc1_miles - Number of miles earned with freq. flyer credit card in the past 12 months:
- cc2_miles - Number of miles earned with Rewards credit card in the past 12 months:
- cc3_miles - Number of miles earned with Small Business credit card in the past 12 months:

1 = under 5,000 2 = 5,000 - 10,000 3 = 10,001 - 25,000 4 = 25,001 - 50,000 5 = over 50,000

- Bonus_miles - Number of miles earned from non-flight bonus transactions in the past 12 months
- Bonus_trans - Number of non-flight bonus transactions in the past 12 months
- Flight_miles_12mo - Number of flight miles in the past 12 months
- Flight_trans_12 - Number of flight transactions in the past 12 months
- Days_since_enrolled - Number of days since enrolled in flier program
- Award - whether that person had award flight (free flight) or not

3. Data Understanding

```
airline_data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3999 entries, 0 to 3998
Data columns (total 12 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   ID#                3999 non-null   int64
 1   Balance            3999 non-null   int64
 2   Qual_miles         3999 non-null   int64
 3   cc1_miles          3999 non-null   int64
 4   cc2_miles          3999 non-null   int64
 5   cc3_miles          3999 non-null   int64
 6   Bonus_miles        3999 non-null   int64
 7   Bonus_trans        3999 non-null   int64
 8   Flight_miles_12mo  3999 non-null   int64
 9   Flight_trans_12    3999 non-null   int64
 10  Days_since_enroll  3999 non-null   int64
 11  Award?             3999 non-null   int64
dtypes: int64(12)
memory usage: 375.0 KB
```

```
airline_data.describe()
```

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bo
count	3999.000000	3.999000e+03	3999.000000	3999.000000	3999.000000	3999.000000	3!
mean	2014.819455	7.360133e+04	144.114529	2.059515	1.014504	1.012253	17
std	1160.764358	1.007757e+05	773.663804	1.376919	0.147650	0.195241	24
min	1.000000	0.000000e+00	0.000000	1.000000	1.000000	1.000000	
25%	1010.500000	1.852750e+04	0.000000	1.000000	1.000000	1.000000	1:
50%	2016.000000	4.309700e+04	0.000000	1.000000	1.000000	1.000000	7
75%	3020.500000	9.240400e+04	0.000000	3.000000	1.000000	1.000000	23
max	4021.000000	1.704838e+06	11148.000000	5.000000	3.000000	5.000000	263

```
airline_data.head()
```

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans
0	1	28143	0	1	1	1	174	1

4. Exploratory Data Analysis

```
airline_data.isnull().sum()
```

```
ID#                 0
Balance             0
Qual_miles          0
cc1_miles           0
cc2_miles           0
cc3_miles           0
Bonus_miles         0
Bonus_trans         0
Flight_miles_12mo   0
Flight_trans_12     0
Days_since_enroll   0
Award?              0
dtype: int64
```

```
airline_data.dtypes
```

```
ID#                 int64
Balance             int64
Qual_miles          int64
cc1_miles           int64
cc2_miles           int64
cc3_miles           int64
Bonus_miles         int64
Bonus_trans         int64
Flight_miles_12mo   int64
Flight_trans_12     int64
Days_since_enroll   int64
Award?              int64
dtype: object
```

```
airline_data
```

	ID#	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_tra
0	1	28143	0	1	1	1	174	
1	2	19244	0	1	1	1	215	
2	3	41354	0	1	1	1	4123	
3	4	14776	0	1	1	1	500	
4	5	97752	0	4	1	1	43300	
...	...	...	...	...	...	...	...	
3994	4017	18476	0	1	1	1	8525	
3995	4018	64385	0	1	1	1	981	
3996	4019	73597	0	3	1	1	25447	
3997	4020	54899	0	1	1	1	500	
3998	4021	3016	0	1	1	1	0	

3999 rows × 12 columns

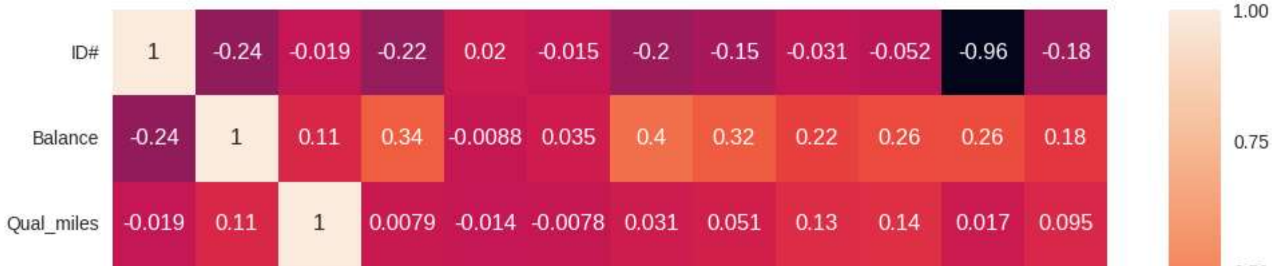
```
sns.heatmap(airline_data.isnull(),yticklabels=False,cmap='viridis')
```

```
<Axes: >
```


```
airline_data.columns
```

```
Index(['ID#', 'Balance', 'Qual_miles', 'cc1_miles', 'cc2_miles', 'cc3_miles',
       'Bonus_miles', 'Bonus_trans', 'Flight_miles_12mo', 'Flight_trans_12',
       'Days_since_enroll', 'Award?'],
      dtype='object')
```

```
### correlation
plt.figure(figsize=(12, 10))
correlation = airline_data.corr()
sns.heatmap(correlation, annot=True)
plt.show()
```




```
plt.figure(figsize=(20, 20))

columns = ['ID#', 'Balance', 'Qual_miles', 'cc1_miles', 'cc2_miles', 'cc3_miles',
       'Bonus_miles', 'Bonus_trans', 'Flight_miles_12mo', 'Flight_trans_12',
       'Days_since_enroll', 'Award?']
airline_data.boxplot()
plt.show()
```

5. Data Preprocessing

```
df = airline_data.drop('ID#', axis=1)
df.head()
```

	Balance	Qual_miles	cc1_miles	cc2_miles	cc3_miles	Bonus_miles	Bonus_trans	Fligh
0	28143	0	1	1	1	174	1	
1	19244	0	1	1	1	215	2	
2	41354	0	1	1	1	4123	4	
3	14776	0	1	1	1	500	1	
4	97752	0	4	1	1	43300	26	

```
X_numerics = df[['Balance', 'Qual_miles', 'cc1_miles', 'cc2_miles', 'cc3_miles',
       'Bonus_miles', 'Bonus_trans', 'Flight_miles_12mo', 'Flight_trans_12',
       'Days_since_enroll', 'Award?']]
```

```
from sklearn.preprocessing import StandardScaler

sc = StandardScaler()
sc.fit(X_numerics)

X_scaled = sc.transform(X_numerics)
```

```
X_scaled
```

```
array([[-4.51140783e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.39545434e+00, -7.66919299e-01],
       [-5.39456874e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.37995704e+00, -7.66919299e-01],
       [-3.20031232e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01,  1.41192021e+00, -7.66919299e-01],
       ...,
       [-4.29480975e-05, -1.86298687e-01,  6.83121167e-01, ...,
        -3.62167870e-01, -1.31560393e+00,  1.30391816e+00],
       [-1.85606976e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -9.85033311e-02, -1.31608822e+00, -7.66919299e-01],
       [-7.00507951e-01, -1.86298687e-01, -7.69578406e-01, ...,
        -3.62167870e-01, -1.31754109e+00, -7.66919299e-01]])
```

```
std_df = pd.DataFrame(X_scaled)
std_df
```

	0	1	2	3	4	5	6	7	8	9	10
0	-0.451141	-0.186299	-0.769578	-0.098242	-0.062767	-0.702786	-1.104065	-0.328603	-0.362168	1.395454	-0.766919
1	-0.539457	-0.186299	-0.769578	-0.098242	-0.062767	-0.701088	-0.999926	-0.328603	-0.362168	1.379957	-0.766919
2	-0.320031	-0.186299	-0.769578	-0.098242	-0.062767	-0.539253	-0.791649	-0.328603	-0.362168	1.411920	-0.766919
3	-0.583799	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	-0.328603	-0.362168	1.372208	-0.766919
4	0.239678	-0.186299	1.409471	-0.098242	-0.062767	1.083121	1.499394	1.154932	0.692490	1.363975	1.303918
...	...	...	...	...	...	...	...	...	...	...	...
3994	-0.547079	-0.186299	-0.769578	-0.098242	-0.062767	-0.356960	-0.791649	-0.185750	-0.098503	-1.315120	1.303918
3995	-0.091465	-0.186299	-0.769578	-0.098242	-0.062767	-0.669367	-0.687511	-0.328603	-0.362168	-1.318994	1.303918
3996	-0.000043	-0.186299	0.683121	-0.098242	-0.062767	0.343804	-0.375096	-0.328603	-0.362168	-1.315604	1.303918
3997	-0.185607	-0.186299	-0.769578	-0.098242	-0.062767	-0.689286	-1.104065	0.028531	-0.098503	-1.316088	-0.766919
3998	-0.700508	-0.186299	-0.769578	-0.098242	-0.062767	-0.709992	-1.208203	-0.328603	-0.362168	-1.317541	-0.766919

3999 rows × 11 columns

6. K Means Clustering

```
X = airline_data.drop(['Award?'], axis=1)
kmeans = KMeans(n_clusters=3, random_state=42)
kmeans.fit(X)
```

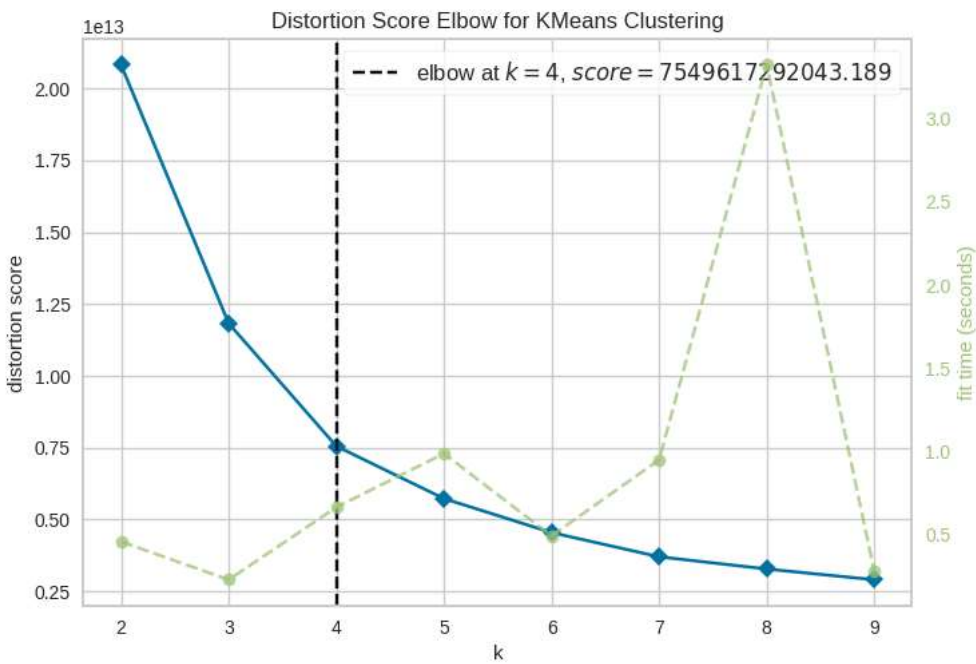
```
KMeans(n_clusters=3, random_state=42)
```

```
KMeans
KMeans(n_clusters=3, random_state=42)
```

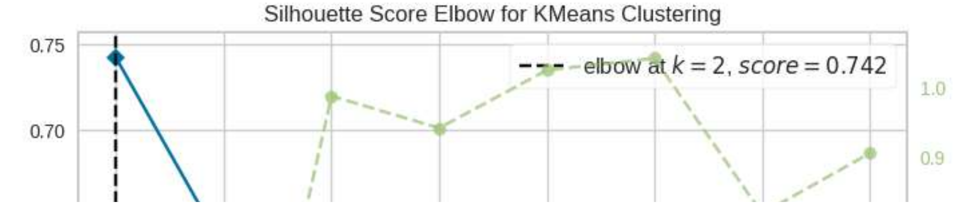
6.1 Elbow Method for Determining Cluster Amount

Standard Scaler Applied on Data

```
from yellowbrick.cluster import KElbowVisualizer
model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10))
visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```




```
model = KMeans(random_state=1)
visualizer = KElbowVisualizer(model, k=(2,10), metric='silhouette')
visualizer.fit(X_numerics)
visualizer.show()
plt.show()
```




```
y= airline_data['Award?']
```

```
# 2 CLUSTER

kmeans = KMeans(n_clusters=2, random_state=42)
kmeans.fit(X)

# check how many of the samples were correctly labeled

labels = kmeans.labels_
correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." %(correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 2577 out of 3999 samples were correctly labeled.
Accuracy score: 0.64
```

```
# 4 CLUSTER

kmeans = KMeans(n_clusters=4, random_state=42)
kmeans.fit(X)

# check how many of the samples were correctly labeled

labels = kmeans.labels_
correct_labels = sum(y == labels)
print("Result: %d out of %d samples were correctly labeled." %(correct_labels, y.size))
print('Accuracy score: {0:0.2f}'. format(correct_labels/float(y.size)))
```

```
Result: 2329 out of 3999 samples were correctly labeled.
Accuracy score: 0.58
```

```
KM_2_clusters = KMeans(n_clusters=2, init='k-means++').fit(X)# initialise and fit   K-Means model
KM2_clustered = X.copy()
KM2_clustered.loc[:,'Cluster'] = KM_2_clusters.labels_ # append labelsto points
```

```
fig1, (axes) = plt.subplots(1,2,figsize=(12,5))
scat_1 = sns.scatterplot(x = "Bonus_miles", y = "Balance", data=KM2_clustered,hue='Cluster', ax=axes[0], palette='Set1', legend='full')
sns.scatterplot(x = 'Qual_miles', y = 'Balance', data=KM2_clustered, hue='Cluster', palette='Set1', ax=axes[1], legend='full')
axes[0].scatter(KM_2_clusters.cluster_centers_[:,1],KM_2_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
axes[1].scatter(KM_2_clusters.cluster_centers_[:,0],KM_2_clusters.cluster_centers_[:,2], marker='s', s=40, c="blue")
plt.show()
```

7. DBSCAN - (Density Based Spatial Clustering of Applications with Noise)

```
from itertools import product

eps_values = np.arange(0.25,3,0.25) # eps values to be investigated
min_samples = np.arange(3,23) # min_samples values to be investigated
DBSCAN_params = list(product(eps_values, min_samples))
```


```
no_of_clusters = []
sil_score = []

for p in DBSCAN_params:
    DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_scaled)
    no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
    sil_score.append(silhouette_score(X_scaled, DBS_clustering.labels_))

tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_samples'])
tmp['No_of_clusters'] = no_of_clusters

pivot_1 = pd.pivot_table(tmp, values='No_of_clusters', index='Min_samples', columns='Eps')
fig, ax = plt.subplots(figsize=(12, 6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 16}, cmap="YlGnBu", ax=ax)
ax.set_title('Number of clusters')
plt.show()

tmp = pd.DataFrame.from_records(DBSCAN_params, columns=['Eps', 'Min_samples'])
tmp['Sil_score'] = sil_score

pivot_1 = pd.pivot_table(tmp, values='Sil_score', index='Min_samples', columns='Eps')
fig, ax = plt.subplots(figsize=(18, 6))
sns.heatmap(pivot_1, annot=True, annot_kws={"size": 10}, cmap="YlGnBu", ax=ax)
plt.show()
```

```
---------------------------------------------------------------------------
KeyboardInterrupt                         Traceback (most recent call last)
<ipython-input-37-7482d6890dc1> in <cell line: 4>()
      3
      4 for p in DBSCAN_params:
----> 5     DBS_clustering = DBSCAN(eps=p[0], min_samples=p[1]).fit(X_scaled)
      6     no_of_clusters.append(len(np.unique(DBS_clustering.labels_)))
      7     sil_score.append(silhouette_score(X_scaled, DBS_clustering.labels_))

                        6 frames
/usr/local/lib/python3.10/dist-packages/sklearn/neighbors/_base.py in
_tree_query_radius_parallel_helper(tree, *args, **kwargs)
   1013     cloudpickle under PyPy.
   1014     """
-> 1015     return tree.query_radius(*args, **kwargs)
   1016
   1017

KeyboardInterrupt:
```

SEARCH STACK OVERFLOW

```
DBS_clustering = DBSCAN(eps=12.5, min_samples=4).fit(X_numerics)
DBSCAN_clustered = X_numerics.copy()
DBSCAN_clustered.loc[:,'Cluster'] = DBS_clustering.labels_
# append labels to points


DBSCAN_clust_sizes =
DBSCAN_clustered.groupby('Cluster').size().to_frame()
DBSCAN_clust_sizes.columns = ["DBSCAN_size"]
DBSCAN_clust_sizes
```