Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2021-22

Experiment 9 (K-Means)

Name: Dev Patel **SAP ID: 60009200016**

Batch: K/K1 **Date: 30/05/2022**

Aim: Explore K means clustering with variations on different datasets.

Theory:

The K-means clustering algorithm computes centroids and repeats until the optimal centroid is found. It is presumptively known how many clusters there are. It is also known as the flat clustering algorithm. The number of clusters found from data by the method is denoted by the letter 'K' in Kmeans.

In this method, data points are assigned to clusters in such a way that the sum of the squared distances between the data points and the centroid is as small as possible. It is essential to note that reduced diversity within clusters leads to more identical data points within the same cluster.

The following stages will help us understand how the K-Means clustering technique works-

Step 1: First, we need to provide the number of clusters, K, that need to be generated by this algorithm.

Step 2: Next, choose K data points at random and assign each to a cluster. Briefly, categorize the data based on the number of data points.

Step 3: The cluster centroids will now be computed.

Step 4: Iterate the steps below until we find the ideal centroid, which is the assigning of data points to clusters that do not vary.

4.1 The sum of squared distances between data points and centroids would be calculated first.

4.2 At this point, we need to allocate each data point to the cluster that is closest to the others (centroid).

4.3 Finally, compute the centroids for the clusters by averaging all of the cluster's data points.

When using the K-means algorithm, we must keep the following points in mind:

It is suggested to normalize the data while dealing with clustering algorithms such as K-Means since such algorithms employ distance-based measurement to identify the similarity between data points.

Because of the iterative nature of K-Means and the random initialization of centroids, K-Means may become stuck in a local optimum and fail to converge to the global optimum. As a result, it is advised to employ distinct centroids' initializations.

Lab Assignments to complete in this session:

Use the given dataset and perform the following tasks:
Dataset 1: Synthetic Data (200 samples, 3 clusters and cluster_std = 2.7)
Dataset 2: **TCGA-PANCAN-HiSeq-801x20531.tar.gz** gene expression cancer RNA-Seq Data Set. his collection of data is part of the RNA-Seq (HiSeq) PANCAN data set, it is a random extraction of gene expressions of patients having different types of tumor: BRCA, KIRC, COAD, LUAD and PRAD.
Dataset 3: Titanic dataset
(http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv
And http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv)

Task 1: Perform Kmeans clustering on Dataset 1 with random initialisation, 10 variations of initial means, 300 iterations. Find Lowest SSE value, final location of centroids and number of iterations to converge.
Show the predicted labels for first 10 points.
Task 2: Perform elbow method and silhouette method to find appropriate clustering value on Dataset 1.
Task 3. Use dataset 2 and create a clustering pipeline with pre-processing using PCA (2 components) and clustering using Kmeans on Dataset 2. Predict the label, calculate the silhouette score and plot a scatterplot for 2 PCA components.
Task 4: Perform data cleaning and pre-processing on dataset 3. Form three clustering using Kmeans++ initialisation.

Code and Output:

Task 1: Perform Kmeans clustering on Dataset 1 with random initialisation, 10 variations of initial means, 300 iterations. Find Lowest SSE value, final location of centroids and number of iterations to converge.

In [1]:

```
from sklearn.datasets import make_blobs
features, true_labels = make_blobs(n_samples=200,centers=3,cluster_std=2.75,random_state
=42)
```

In [2]:

```
features[:5]
```

Out[2]:

```
array([[  9.77075874,   3.27621022],
       [ -9.71349666,  11.27451802],
       [ -6.91330582,  -9.34755911],
       [-10.86185913, -10.75063497],
       [ -8.50038027,  -4.54370383]])
```

In [3]:

```
true_labels[:5]
```

Out[3]:

```
array([1, 0, 2, 2, 2])
```

In [4]:

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler()
scaled_features = scaler.fit_transform(features)
```

In [5]:

```
from sklearn.cluster import KMeans
kmeans = KMeans(init="random",n_clusters=3,n_init=10,max_iter=300,random_state=42)
kmeans.fit(scaled_features)
```

Out[5]:

```
KMeans(init='random', n_clusters=3, random_state=42)
```

In [6]:

```
kmeans.n_iter_
```

Out[6]:

```
2
```

In [7]:

```
kmeans.cluster_centers_
```

Out[7]:

```
array([[-0.25813925,  1.05589975],
       [-0.91941183, -1.18551732],
       [ 1.19539276,  0.13158148]])
```

In [8]:

```
kmeans.inertia_
```

```
kmeans.inertia_
```

Out[8]:

```
74.57960106819854
```

In [9]:

```python
kmeans_kwargs = {"init": "random","n_init": 10,"max_iter": 300,"random_state": 42,}
```

Task 2: Perform elbow method and silhouette method to find appropriate clustering value on Dataset 1.

In [10]:

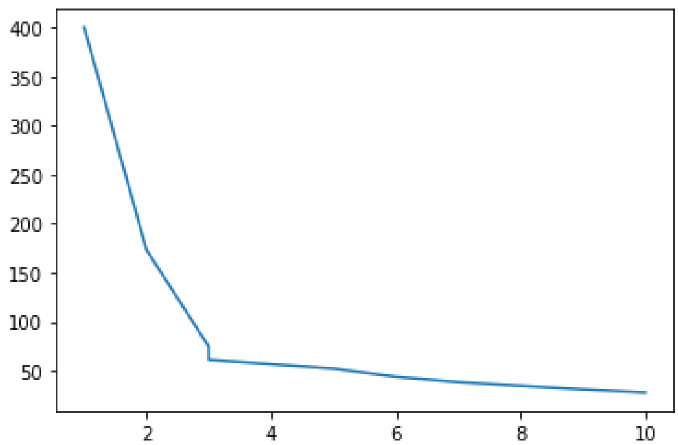
```python
# A list holds the SSE values for each k
sse = []
for k in range(1, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scaled_features)
    sse.append(kmeans.inertia_)
```

In [11]:

```python
import matplotlib.pyplot as plt
x = [1,2,3,3,5,6,7,8,9,10]
plt.plot(x,sse)
```

Out[11]:

```
[<matplotlib.lines.Line2D at 0x7f44d582e090>]
```



In [12]:

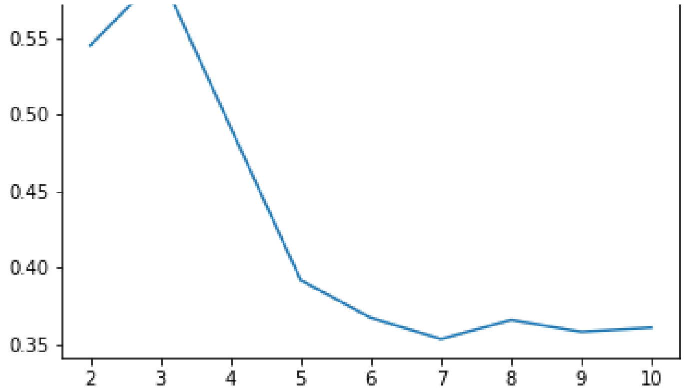
```python
from sklearn.metrics import silhouette_score
# A list holds the silhouette coefficients for each k
silhouette_coefficients = []
# Notice you start at 2 clusters for silhouette coefficient
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(scaled_features)
    score = silhouette_score(scaled_features, kmeans.labels_)
    silhouette_coefficients.append(score)
```

In [13]:

```python
import matplotlib.pyplot as plt
x = [2,3,4,5,6,7,8,9,10]
plt.plot(x,silhouette_coefficients)
```

Out[13]:

```
[<matplotlib.lines.Line2D at 0x7f44d5312e90>]
```



Task 3. Use dataset 2 and create a clustering pipeline with pre-processing using PCA (2 components) and clustering using Kmeans on Dataset 2. Predict the label, calculate the silhouette score and plot a scatterplot for 2 PCA components.

In [14]:

```
import tarfile
import urllib
import numpy as np
```

In [15]:

```
uci_tcga_url = "https://archive.ics.uci.edu/ml/machine-learning-databases/00401/"
archive_name = "TCGA-PANCAN-HiSeq-801x20531.tar.gz"

# Build the url
full_download_url = urllib.parse.urljoin(uci_tcga_url, archive_name)
# Download the file
r = urllib.request.urlretrieve (full_download_url, archive_name)

# Extract the data from the archive
tar = tarfile.open(archive_name, "r:gz")
tar.extractall()
tar.close()
```

In [16]:

```
datafile = "TCGA-PANCAN-HiSeq-801x20531/data.csv"
labels_file = "TCGA-PANCAN-HiSeq-801x20531/labels.csv"
data = np.genfromtxt(datafile, delimiter=",",usecols=range(1, 20532),skip_header=1)
true_label_names = np.genfromtxt(labels_file,delimiter=",",usecols=(1,),skip_header=1,dt
ype="str")
```

In [17]:

```
data[:5, :3]
```

Out[17]:

```
array([[0.        , 2.01720929, 3.26552691],
       [0.        , 0.59273209, 1.58842082],
       [0.        , 3.51175898, 4.32719872],
       [0.        , 3.66361787, 4.50764878],
       [0.        , 2.65574107, 2.82154696]])
```

In [18]:

```
true_label_names[:5]
```

Out[18]:

```
array(['PRAD', 'LUAD', 'PRAD', 'PRAD', 'BRCA'], dtype='<U4')
```

In [19]:

```python
from sklearn.pipeline import Pipeline
from sklearn.cluster import KMeans
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score, adjusted_rand_score
from sklearn.pipeline import Pipeline
from sklearn.preprocessing import LabelEncoder, MinMaxScaler
```

In [20]:

```python
label_encoder = LabelEncoder()
true_labels = label_encoder.fit_transform(true_label_names)
true_labels[:5]
```

Out[20]:

```
array([4, 3, 4, 4, 0])
```

In [21]:

```python
label_encoder.classes_
```

Out[21]:

```
array(['BRCA', 'COAD', 'KIRC', 'LUAD', 'PRAD'], dtype='<U4')
```

In [22]:

```python
n_clusters = len(label_encoder.classes_)
```

In [23]:

```python
preprocessor = Pipeline([("scaler", MinMaxScaler()),("pca", PCA(n_components=2, random_st
ate=42)),])
```

In [24]:

```python
clusterer = Pipeline([("kmeans",KMeans(n_clusters=n_clusters,init="k-means++",n_init=50,m
ax_iter=500,random_state=42,),),])
```

In [25]:

```python
pipe = Pipeline([("preprocessor", preprocessor),("clusterer", clusterer)])
pipe.fit(data)
```

Out[25]:

```
Pipeline(steps=[('preprocessor',
                 Pipeline(steps=[('scaler', MinMaxScaler()),
                                 ('pca',
                                  PCA(n_components=2, random_state=42))])),
                ('clusterer',
                 Pipeline(steps=[('kmeans',
                                  KMeans(max_iter=500, n_clusters=5, n_init=50,
                                         random_state=42))]))])
```

In [26]:

```python
preprocessed_data = pipe["preprocessor"].transform(data)
```

In [27]:

```python
predicted_labels = pipe["clusterer"]["kmeans"].labels_
```

In [28]:

```python
silhouette_score(preprocessed_data, predicted_labels)
```

Out[28]:

```
0.5118775528450308
```

In [29]:

```
adjusted_rand_score(true_labels, predicted_labels)
```

Out[29]:

```
0.722276752060253
```

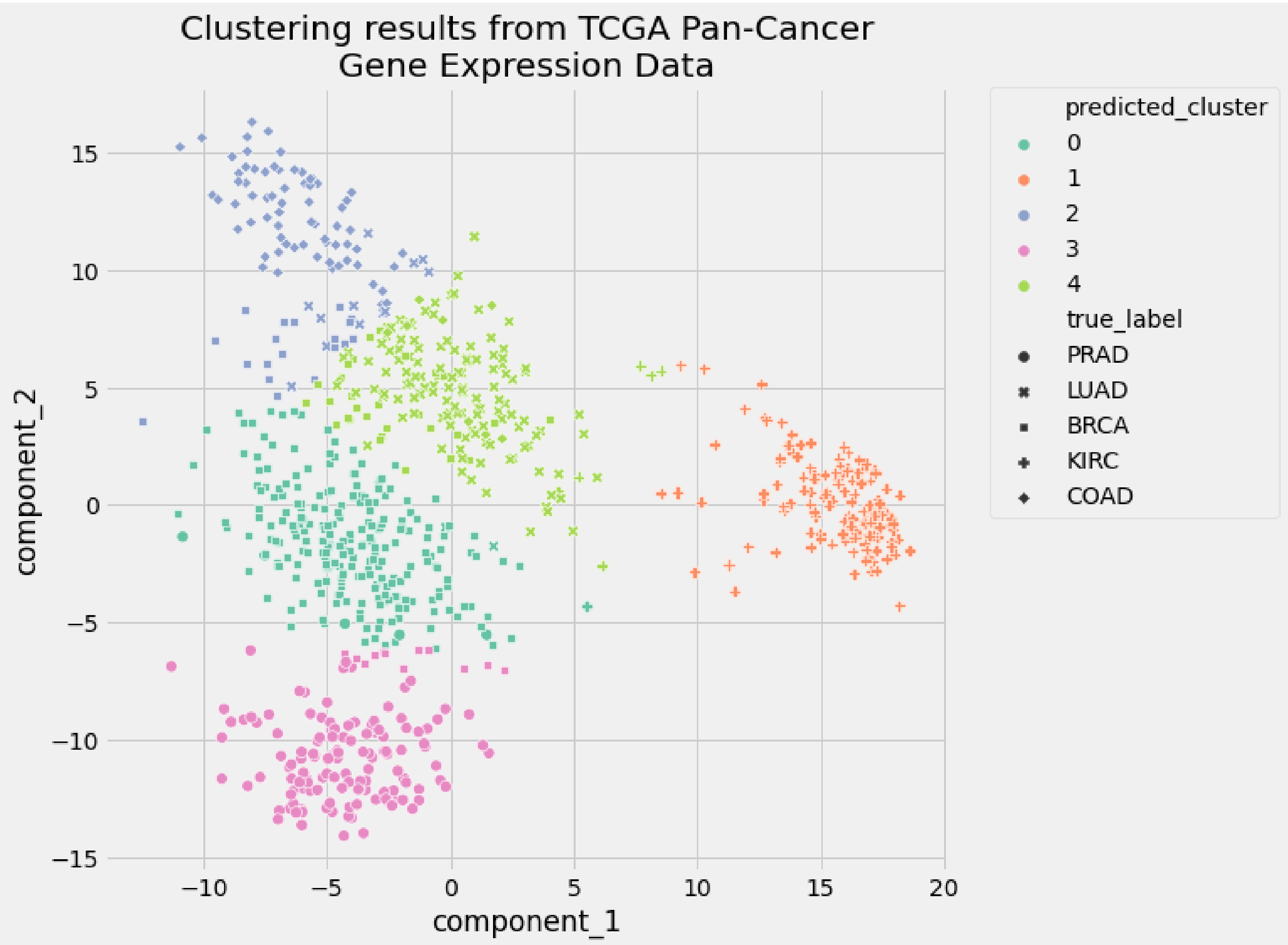
In [30]:

```
import pandas as pd
import seaborn as sns
pcadf = pd.DataFrame(pipe["preprocessor"].transform(data),columns=["component_1", "compo
nent_2"],)
pcadf["predicted_cluster"] = pipe["clusterer"]["kmeans"].labels_
pcadf["true_label"] = label_encoder.inverse_transform(true_labels)
```

In [31]:

```
plt.style.use("fivethirtyeight")
plt.figure(figsize=(8, 8))
scat = sns.scatterplot("component_1","component_2",s=50,data=pcadf,hue="predicted_cluster
",style="true_label",palette="Set2",)
scat.set_title("Clustering results from TCGA Pan-Cancer\nGene Expression Data")
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.0)
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the
following variables as keyword args: x, y. From version 0.12, the only valid positional a
rgument will be `data`, and passing other arguments without an explicit keyword will resu
lt in an error or misinterpretation.
  FutureWarning
```



In [32]:

```
# A list holds the silhouette coefficients for each k
silhouette_coefficients = []
# Notice you start at 2 clusters for silhouette coefficient
```

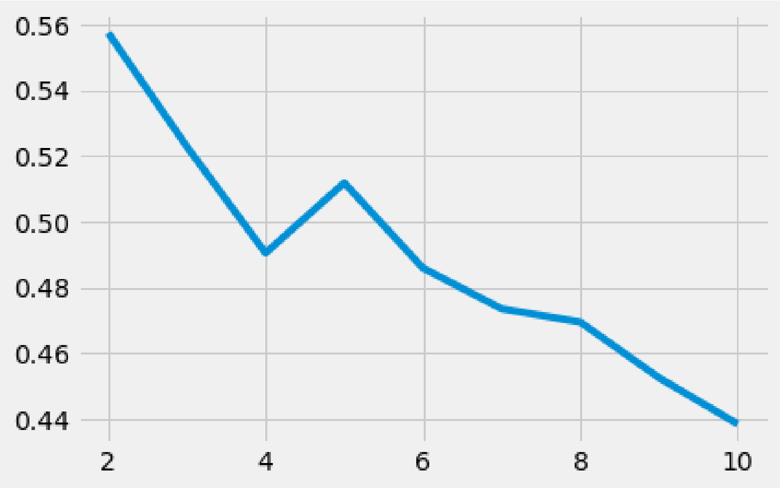
```
for k in range(2, 11):
    kmeans = KMeans(n_clusters=k, **kmeans_kwargs)
    kmeans.fit(preprocessed_data)
    score = silhouette_score(preprocessed_data, kmeans.labels_)
    silhouette_coefficients.append(score)
```

In [33]:

```
import matplotlib.pyplot as plt
x = [2,3,4,5,6,7,8,9,10]
plt.plot(x,silhouette_coefficients)
```

Out[33]:

```
[<matplotlib.lines.Line2D at 0x7f44cb19c390>]
```



Task 4: Perform data cleaning and pre-processing on dataset 3. Form three clustering using Kmeans++ initialisation.

In [34]:

```
import pandas as pd
import fsspec
train = pd.read_csv('http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv'
)
test = pd.read_csv('http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv')
```

In [35]:

```
train.head()
```

Out[35]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250	NaN	S
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000	C123	S
4	5	0	3	Allen, Mr. William Henry	male	35.0	0	0	373450	8.0500	NaN	S

In [36]:

```
test.head()
```

Out[36]:

	PassengerId	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	892	3	Kelly, Mr. James	male	34.5	0	0	330911	7.8292	NaN	Q
1	893	3	Wilkes, Mrs. James (Ellen Needs)	female	47.0	1	0	363272	7.0000	NaN	S
2	894	2	Myles, Mr. Thomas Francis	male	62.0	0	0	240276	9.6875	NaN	Q
3	895	3	Wirz, Mr. Albert	male	27.0	0	0	315154	8.6625	NaN	S
4	896	3	Hirvonen, Mrs. Alexander (Helga E Lindqvist)	female	22.0	1	1	3101298	12.2875	NaN	S

In [37]:

```
train.shape
```

Out[37]:

```
(891, 12)
```

In [38]:

```
test.shape
```

Out[38]:

```
(418, 11)
```

In [39]:

```
train.isnull().sum()
```

Out[39]:

```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

In [40]:

```
test.isnull().sum()
```

Out[40]:

```
PassengerId      0
Pclass           0
Name             0
Sex              0
Age             86
SibSp            0
Parch            0
Ticket           0
Fare             1
Cabin          327
Embarked         0
dtype: int64
```

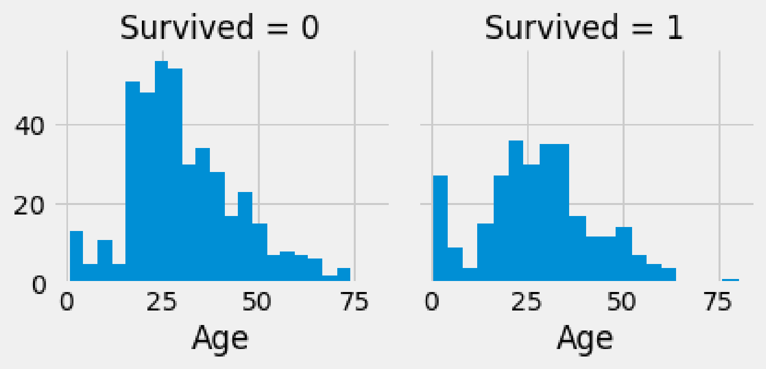
In [41]:

```
train[["SibSp", "Survived"]].groupby(['SibSp'], as_index=False).mean().sort_values(by='Survived', ascending=False)
```

```
g = sns.FacetGrid(train, col='Survived')
g.map(plt.hist, 'Age', bins=20)
```

Out[41]:

<seaborn.axisgrid.FacetGrid at 0x7f44c97f7b10>



In [42]:

```
grid = sns.FacetGrid(train, col='Survived', row='Pclass', size=2.2, aspect=1.6)
grid.map(plt.hist, 'Age', alpha=.5, bins=20)
grid.add_legend();
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/axisgrid.py:337: UserWarning: The `size` p
arameter has been renamed to `height`; please update your code.
  warnings.warn(msg, UserWarning)
```

In [43]:

```
train = train.drop(["Cabin"],axis = 1)
test = test.drop(["Cabin"],axis = 1)
train = train.drop(["Name"],axis = 1)
test = test.drop(["Name"],axis = 1)
train = train.drop(["PassengerId"],axis = 1)
test = test.drop(["PassengerId"],axis = 1)
train = train.drop(["Ticket"],axis = 1)
test = test.drop(["Ticket"],axis = 1)
y = np.array(train['Survived'])
train = train.drop(["Survived"],axis = 1)
train['Age'] = train['Age'].fillna(train['Age'].median())
test['Age'] = test['Age'].fillna(test['Age'].median())
```

```
test['Fare'] = test['Fare'].fillna(test['Fare'].mean())
train['Embarked'] = train['Embarked'].fillna(train['Embarked'].mode()[0])
```

In [44]:

```
train.isnull().sum()
```

Out[44]:

```
Pclass      0
Sex         0
Age         0
SibSp       0
Parch       0
Fare        0
Embarked    0
dtype: int64
```

In [45]:

```
label_encoder = LabelEncoder()
train["Sex"] = label_encoder.fit_transform(train["Sex"].values)
test["Sex"] = label_encoder.fit_transform(test["Sex"].values)
train["Embarked"] = label_encoder.fit_transform(train["Embarked"].values)
test["Embarked"] = label_encoder.fit_transform(test["Embarked"].values)
train.head()
```

Out[45]:

	Pclass	Sex	Age	SibSp	Parch	Fare	Embarked
0	3	1	22.0	1	0	7.2500	2
1	1	0	38.0	1	0	71.2833	0
2	3	0	26.0	0	0	7.9250	2
3	1	0	35.0	1	0	53.1000	2
4	3	1	35.0	0	0	8.0500	2

In [46]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    891 non-null    int64  
 1   Sex       891 non-null    int64  
 2   Age       891 non-null    float64
 3   SibSp     891 non-null    int64  
 4   Parch     891 non-null    int64  
 5   Fare      891 non-null    float64
 6   Embarked  891 non-null    int64  
dtypes: float64(2), int64(5)
memory usage: 48.9 KB
```

In [47]:

```
test.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 418 entries, 0 to 417
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   Pclass    418 non-null    int64  
 1   Sex       418 non-null    int64  
 2   Age       418 non-null    float64
 3   SibSp     418 non-null    int64  
 4   Parch     418 non-null    int64
```

```
 4   Parch     418 non-null    int64
 5   Fare      418 non-null    float64
 6   Embarked  418 non-null    int64
dtypes: float64(2), int64(5)
memory usage: 23.0 KB
```

In [48]:

```
# Normalize parameters in training dataframe X
scaler = MinMaxScaler()
train = scaler.fit_transform(train)
```

In [49]:

```
kmeans = KMeans( init='k-means++', max_iter=600, n_clusters=2, n_init=10)
kmeans.fit(train)
```

Out[49]:

```
KMeans(max_iter=600, n_clusters=2)
```

In [50]:

```
correct = 0
for i in range(len(train)):
    predict_me = np.array(train[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    if prediction[0] == y[i]:
        correct += 1
print("Accuracy: ",end="")
print(str(correct/len(train)*100)+"%")
```

```
Accuracy: 78.67564534231201%
```