Department of Computer Science and Engineering (Data Science)

Subject: Machine Learning – I (DJ19DSC402)

AY: 2022-23

Experiment 9

(K-Means)

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 K-means.
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.

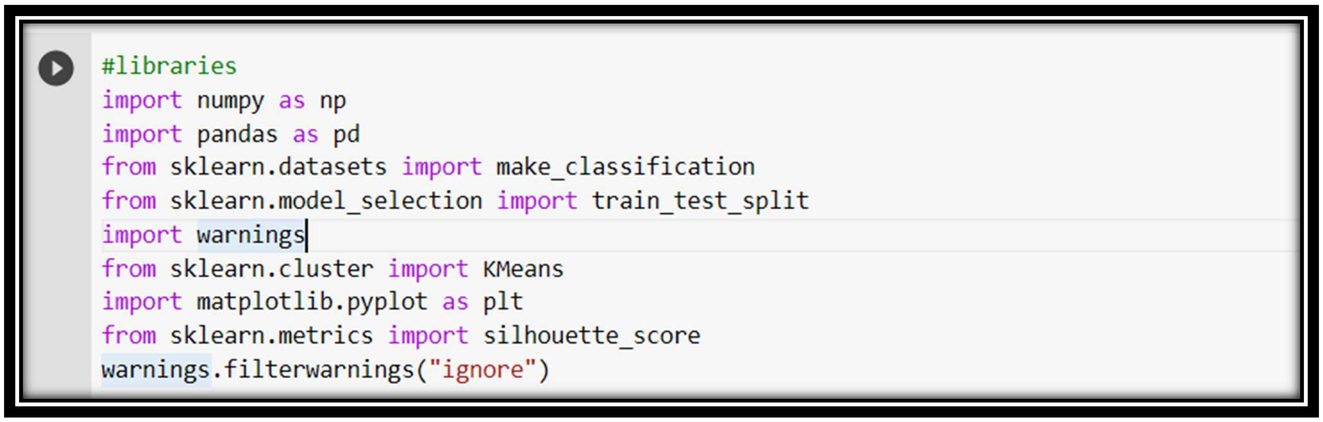
When using the K-means algorithm, we must keep the following points in mind:
It is suggested to normalize the data while dealing with clustering algorithms such as K-Means since such algorithms employ distance-based measurement to identify the similarity between data points.
Because of the iterative 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:

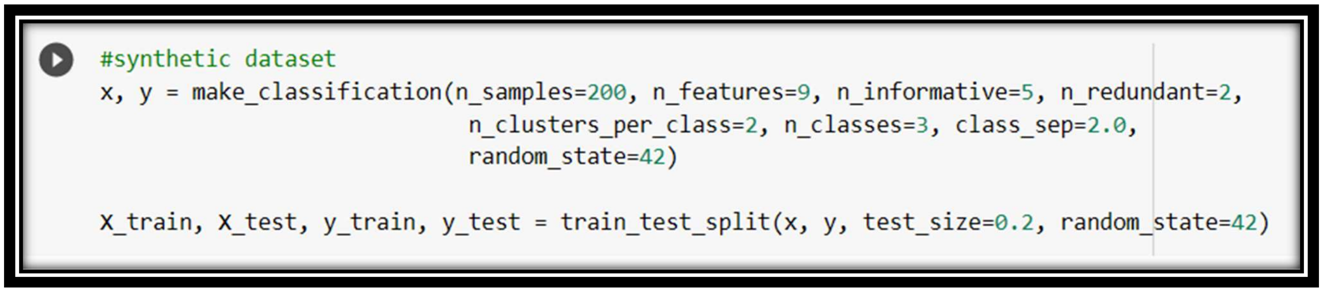
Use the given dataset and perform the following tasks:
Dataset 1: Synthetic Data (200 samples, 3 clusters and cluster_std = 2.7)
Dataset 2: Titanic dataset (http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/train.csv

And http://s3.amazonaws.com/assets.datacamp.com/course/Kaggle/test.csv)

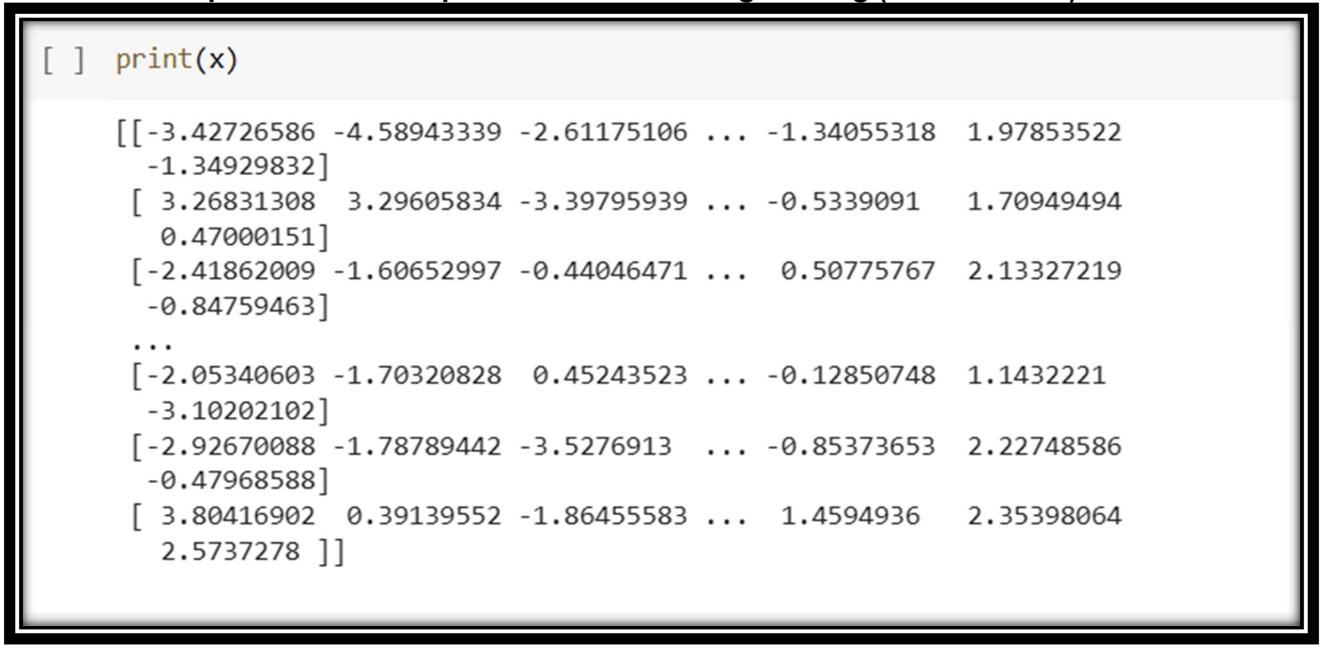
Task 1: Perform Kmeans clustering on Dataset 1 with random initialisation, 10 variations of initial means, 300 iteration. Find Lowest SSE value, final location of centroids and number of iterations to converge. Show the predicted labels for first 10 points.
Task 2: Perform elbow method and silhouette method to find appropriate clustering value on Dataset 1.
Task 3: Preform data cleaning and pre-processing on dataset 2. Form three clustering using Kmeans++ initialisation.



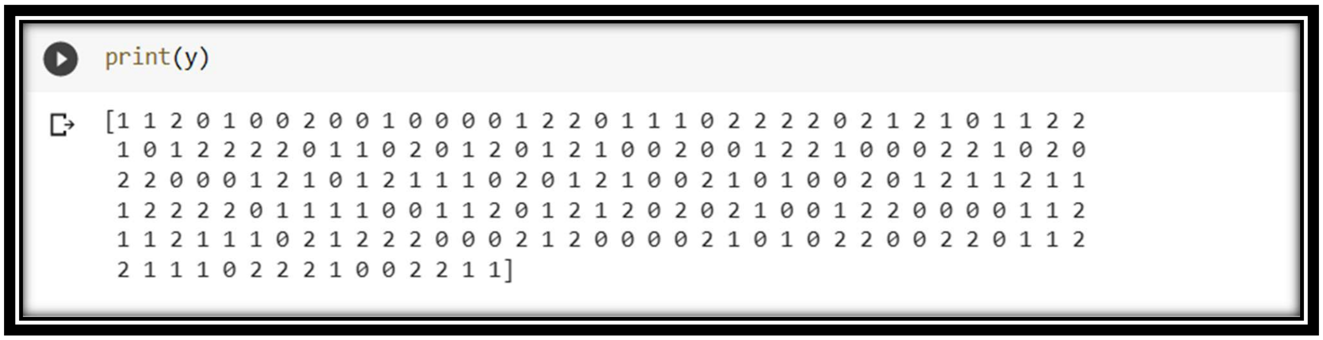
```
#libraries
import numpy as np
import pandas as pd
from sklearn.datasets import make_classification
from sklearn.model_selection import train_test_split
import warnings
from sklearn.cluster import KMeans
import matplotlib.pyplot as plt
from sklearn.metrics import silhouette_score
warnings.filterwarnings("ignore")
```



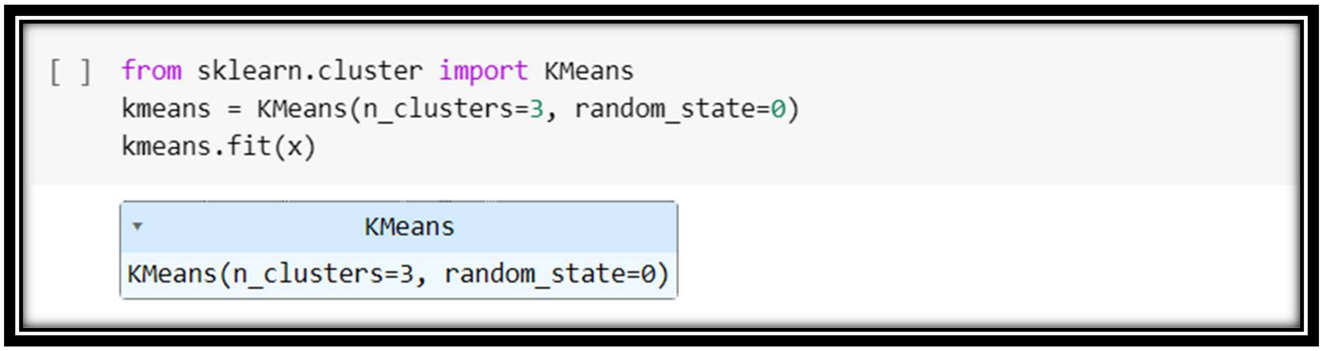
```
#synthetic dataset
x, y = make_classification(n_samples=200, n_features=9, n_informative=5, n_redundant=2,
                           n_clusters_per_class=2, n_classes=3, class_sep=2.0,
                           random_state=42)

X_train, X_test, y_train, y_test = train_test_split(x, y, test_size=0.2, random_state=42)
```



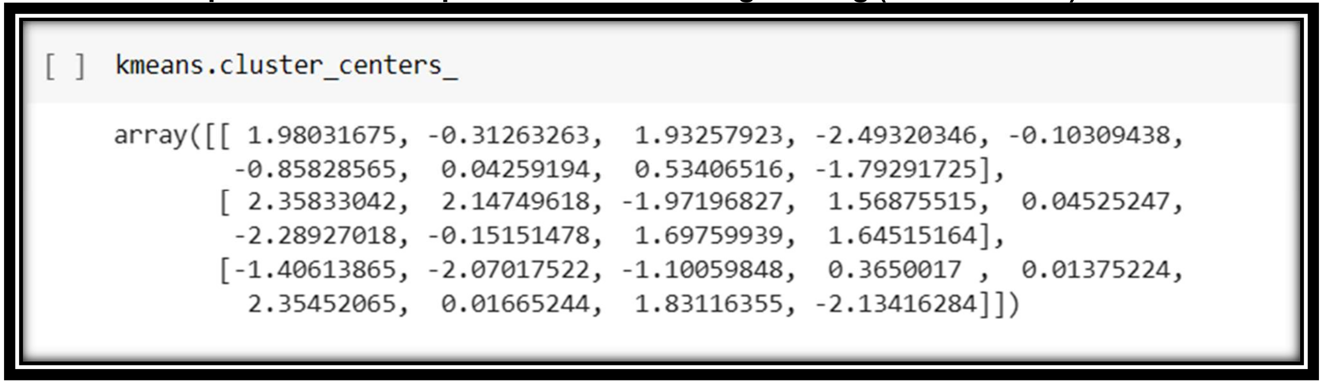
```
[ ]  print(x)

     [[-3.42726586 -4.58943339 -2.61175106 ... -1.34055318  1.97853522
       -1.34929832]
      [ 3.26831308  3.29605834 -3.39795939 ... -0.5339091   1.70949494
        0.47000151]
      [-2.41862009 -1.60652997 -0.44046471 ...  0.50775767  2.13327219
       -0.84759463]
      ...
      [-2.05340603 -1.70320828  0.45243523 ... -0.12850748  1.1432221
       -3.10202102]
      [-2.92670088 -1.78789442 -3.5276913  ... -0.85373653  2.22748586
       -0.47968588]
      [ 3.80416902  0.39139552 -1.86455583 ...  1.4594936   2.35398064
        2.5737278 ]]
```



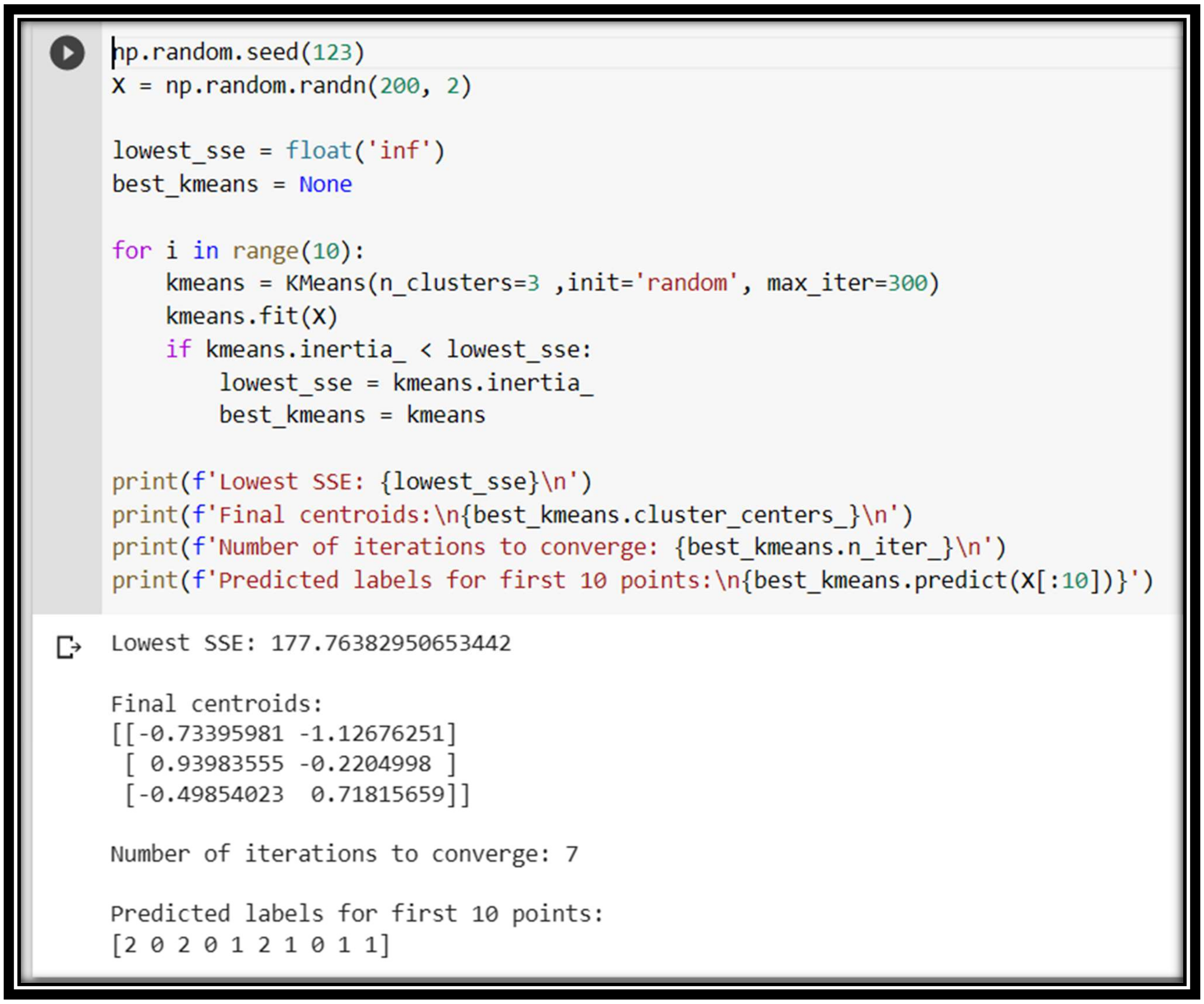
```
print(y)

[1 1 2 0 1 0 0 2 0 0 1 0 0 0 0 1 2 2 0 1 1 1 0 2 2 2 2 0 2 1 2 1 0 1 1 2 2
 1 0 1 2 2 2 2 0 1 1 0 2 0 1 2 0 1 2 1 0 0 2 0 0 1 2 2 1 0 0 0 2 2 1 0 2 0
 2 2 0 0 0 1 2 1 0 1 2 1 1 1 0 2 0 1 2 1 0 0 2 1 0 1 0 0 2 0 1 2 1 1 2 1 1
 1 2 2 2 2 0 1 1 1 1 0 0 1 1 2 0 1 2 1 2 0 2 0 2 1 0 0 1 2 2 0 0 0 0 1 1 2
 1 1 2 1 1 1 0 2 1 2 2 2 0 0 0 2 1 2 0 0 0 0 2 1 0 1 0 2 2 0 0 2 2 0 1 1 2
 2 1 1 1 0 2 2 2 1 0 0 2 2 1 1]
```



```
[ ]  from sklearn.cluster import KMeans
     kmeans = KMeans(n_clusters=3, random_state=0)
     kmeans.fit(x)

     KMeans
     KMeans(n_clusters=3, random_state=0)
```

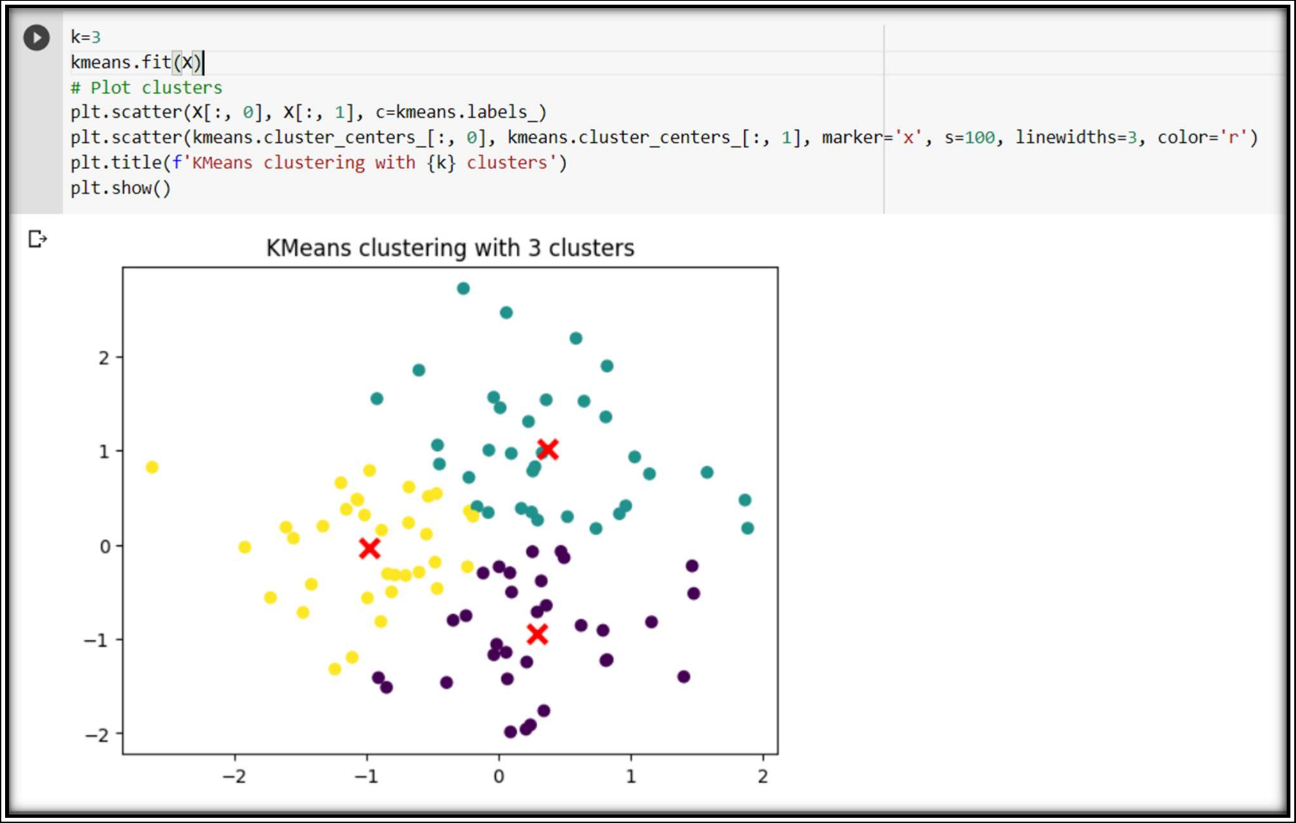


```
[ ]  kmeans.cluster_centers_

     array([[ 1.98031675, -0.31263263,  1.93257923, -2.49320346, -0.10309438,
             -0.85828565,  0.04259194,  0.53406516, -1.79291725],
            [ 2.35833042,  2.14749618, -1.97196827,  1.56875515,  0.04525247,
             -2.28927018, -0.15151478,  1.69759939,  1.64515164],
            [-1.40613865, -2.07017522, -1.10059848,  0.3650017 ,  0.01375224,
              2.35452065,  0.01665244,  1.83116355, -2.13416284]])
```

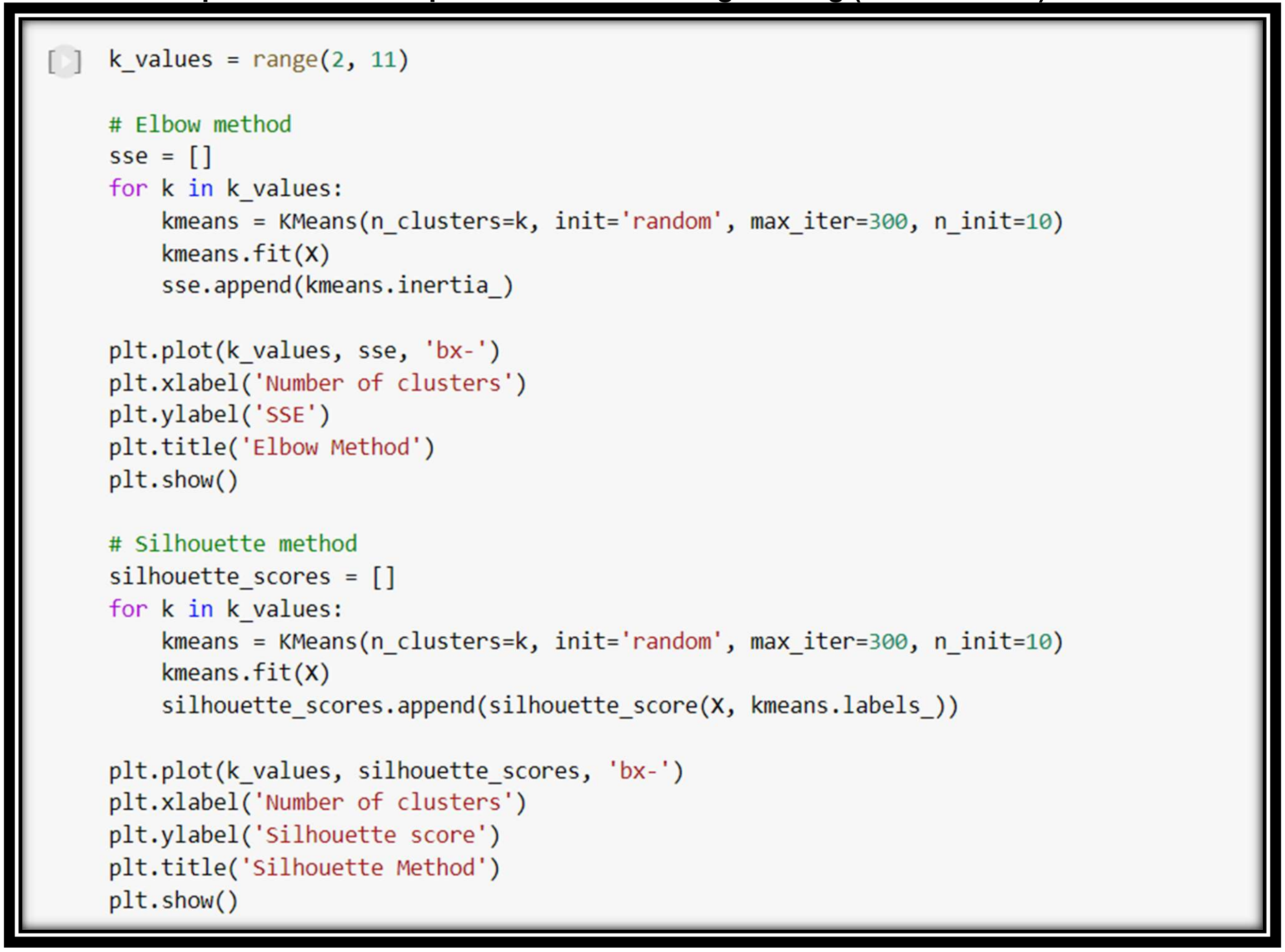


```
np.random.seed(123)
X = np.random.randn(200, 2)

lowest_sse = float('inf')
best_kmeans = None

for i in range(10):
    kmeans = KMeans(n_clusters=3 ,init='random', max_iter=300)
    kmeans.fit(X)
    if kmeans.inertia_ < lowest_sse:
        lowest_sse = kmeans.inertia_
        best_kmeans = kmeans

print(f'Lowest SSE: {lowest_sse}\n')
print(f'Final centroids:\n{best_kmeans.cluster_centers_}\n')
print(f'Number of iterations to converge: {best_kmeans.n_iter_}\n')
print(f'Predicted labels for first 10 points:\n{best_kmeans.predict(X[:10])}')
```

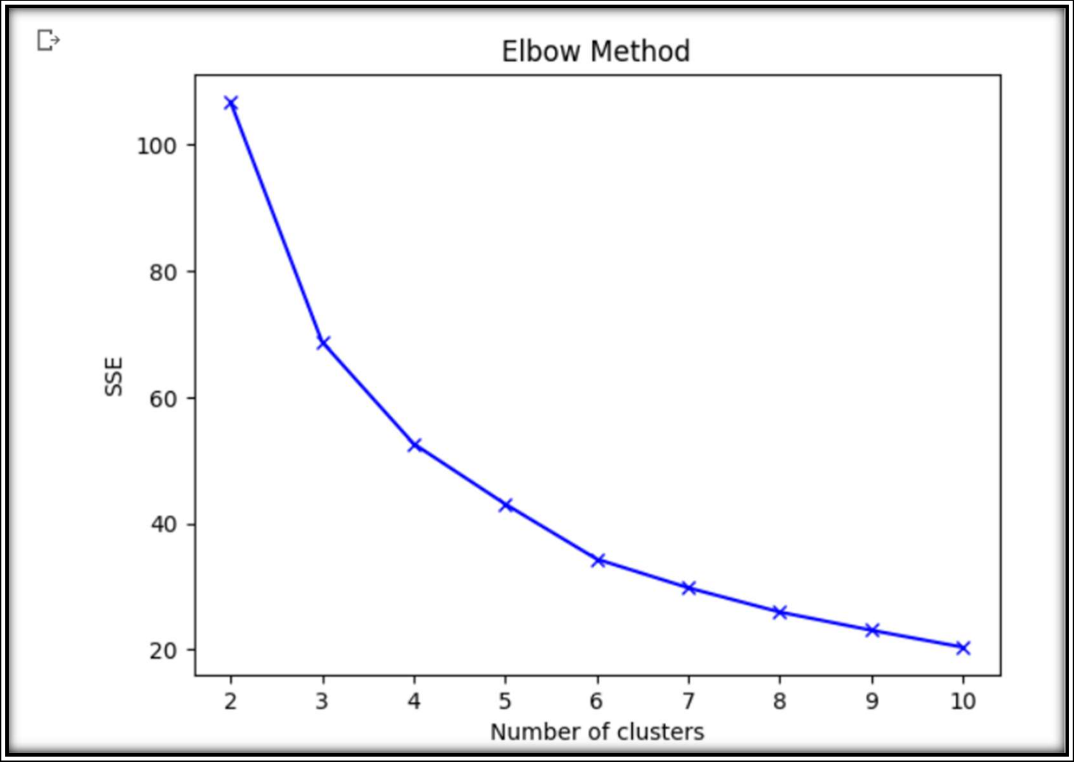
```
Lowest SSE: 177.76382950653442

Final centroids:
[[-0.73395981 -1.12676251]
 [ 0.93983555 -0.2204998 ]
 [-0.49854023  0.71815659]]

Number of iterations to converge: 7

Predicted labels for first 10 points:
[2 0 2 0 1 2 1 0 1 1]
```

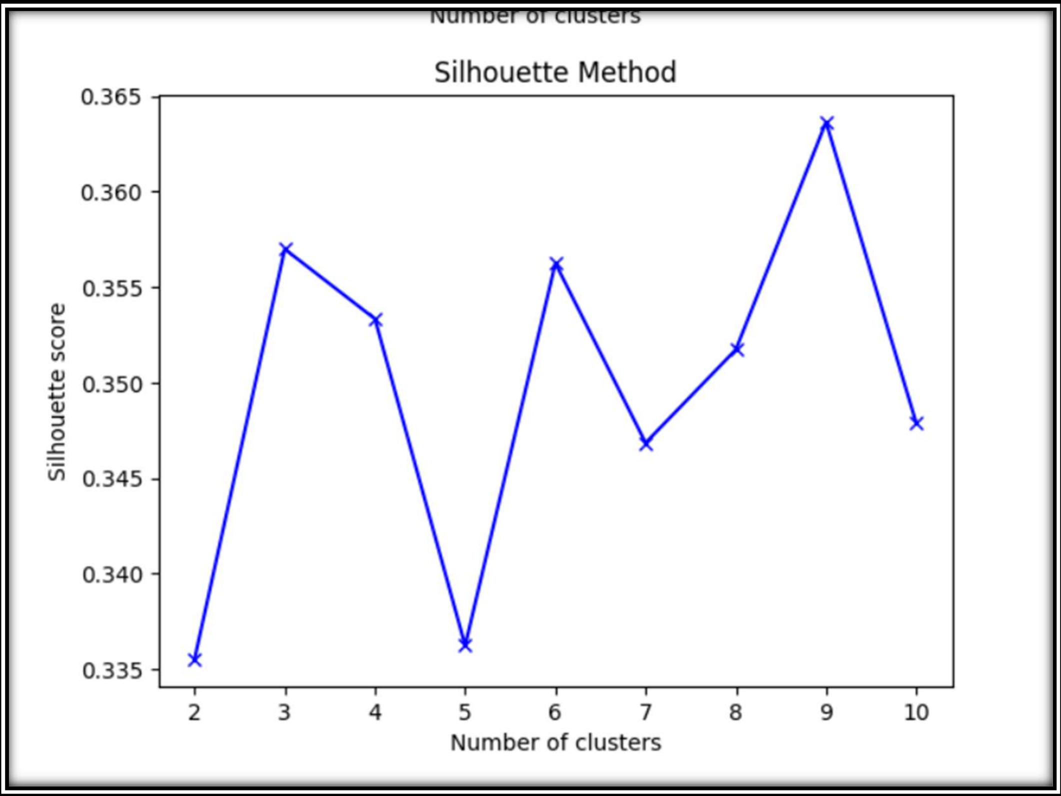


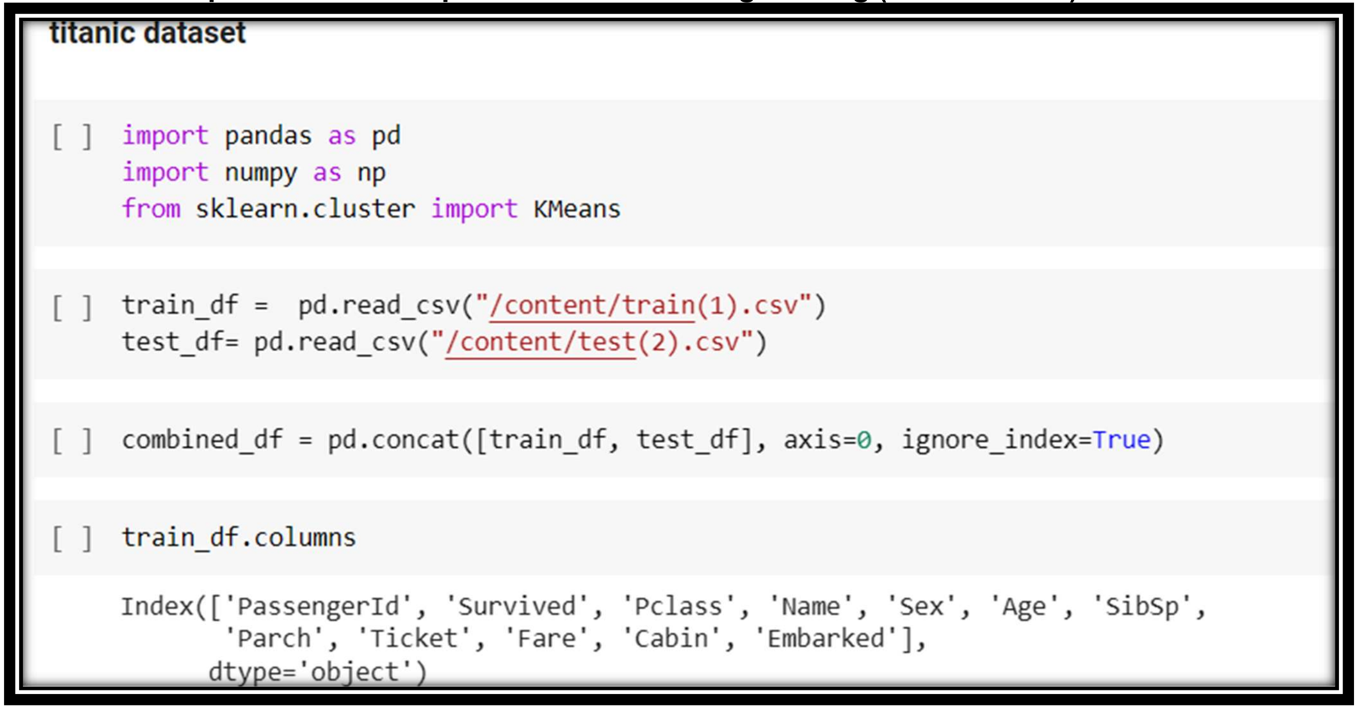
```
k=3
kmeans.fit(X)
# Plot clusters
plt.scatter(X[:, 0], X[:, 1], c=kmeans.labels_)
plt.scatter(kmeans.cluster_centers_[:, 0], kmeans.cluster_centers_[:, 1], marker='x', s=100, linewidths=3, color='r')
plt.title(f'KMeans clustering with {k} clusters')
plt.show()
```



```
k_values = range(2, 11)

# Elbow method
sse = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, init='random', max_iter=300, n_init=10)
    kmeans.fit(X)
    sse.append(kmeans.inertia_)

plt.plot(k_values, sse, 'bx-')
plt.xlabel('Number of clusters')
plt.ylabel('SSE')
plt.title('Elbow Method')
plt.show()

# Silhouette method
silhouette_scores = []
for k in k_values:
    kmeans = KMeans(n_clusters=k, init='random', max_iter=300, n_init=10)
    kmeans.fit(X)
    silhouette_scores.append(silhouette_score(X, kmeans.labels_))

plt.plot(k_values, silhouette_scores, 'bx-')
plt.xlabel('Number of clusters')
plt.ylabel('Silhouette score')
plt.title('Silhouette Method')
plt.show()
```


Elbow Method
SSE
20
40
60
80
100
2
3
4
5
6
7
8
9
10
Number of clusters


Number of clusters
Silhouette Method
Silhouette score
0.335
0.340
0.345
0.350
0.355
0.360
0.365
2
3
4
5
6
7
8
9
10
Number of clusters



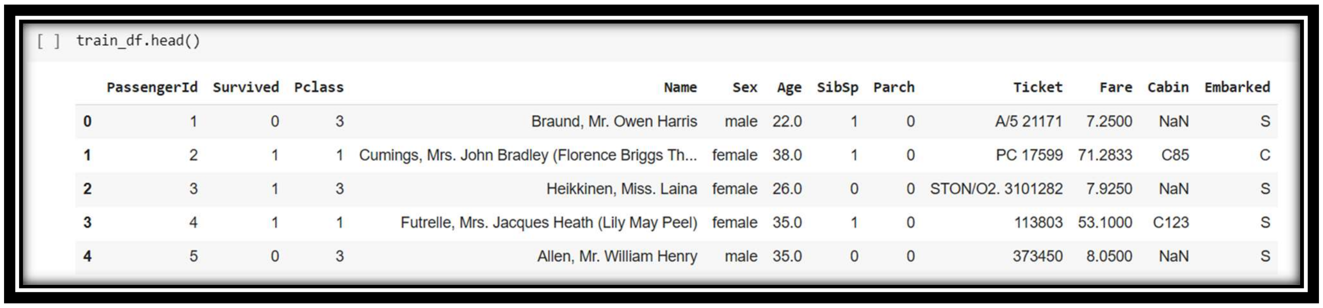
titanic dataset

```
[ ] import pandas as pd
    import numpy as np
    from sklearn.cluster import KMeans
```

```
[ ] train_df =  pd.read_csv("/content/train(1).csv")
    test_df= pd.read_csv("/content/test(2).csv")
```

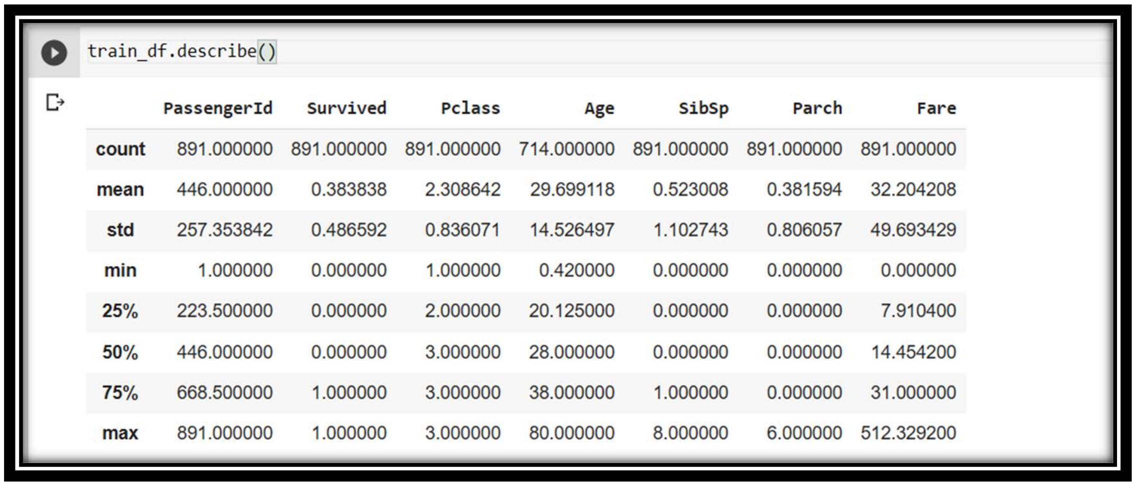
```
[ ] combined_df = pd.concat([train_df, test_df], axis=0, ignore_index=True)
```

```
[ ] train_df.columns

    Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
           'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
          dtype='object')
```



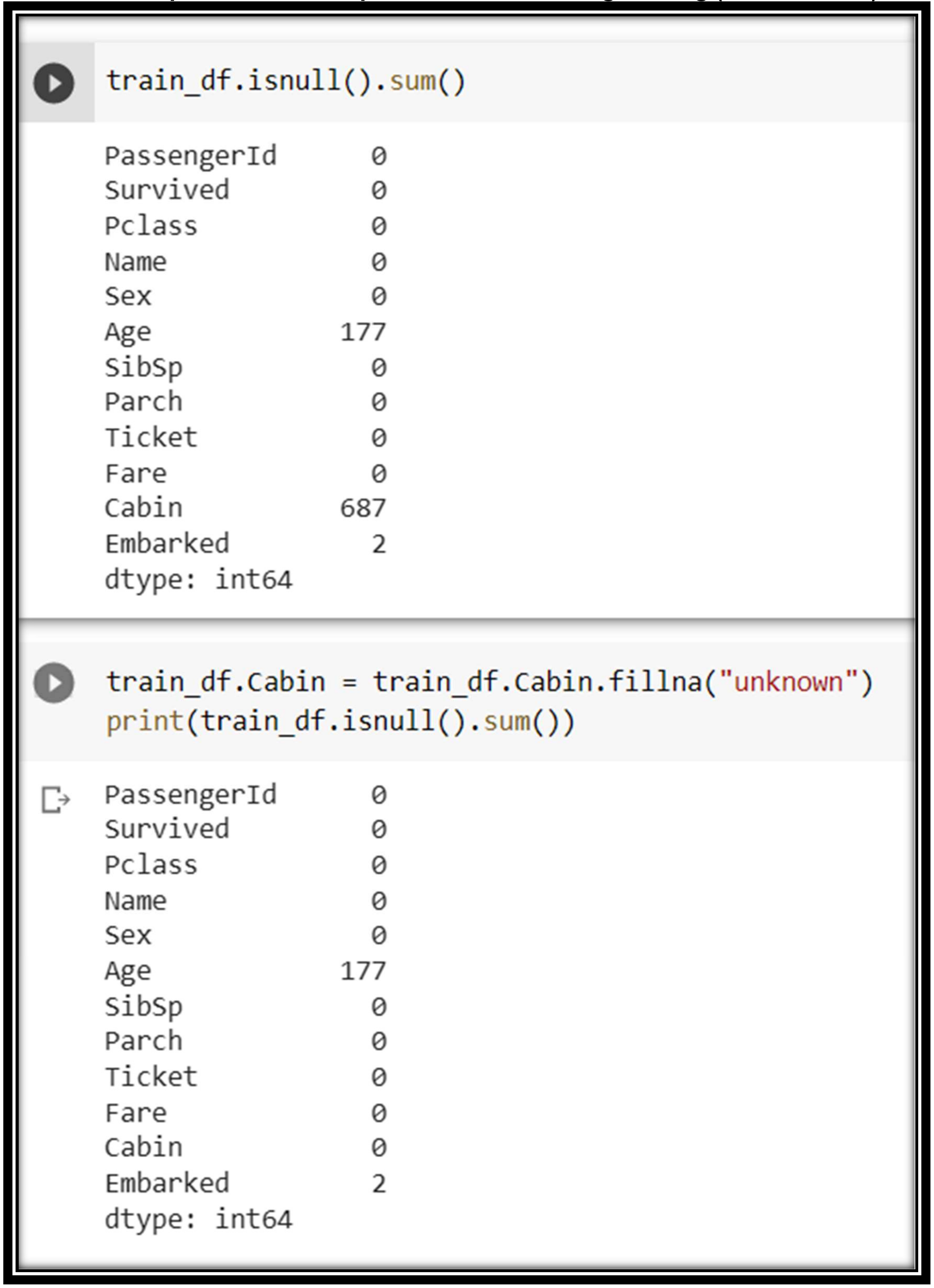
```
[ ] train_df.head()
```

	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



```
train_df.describe()
```

	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

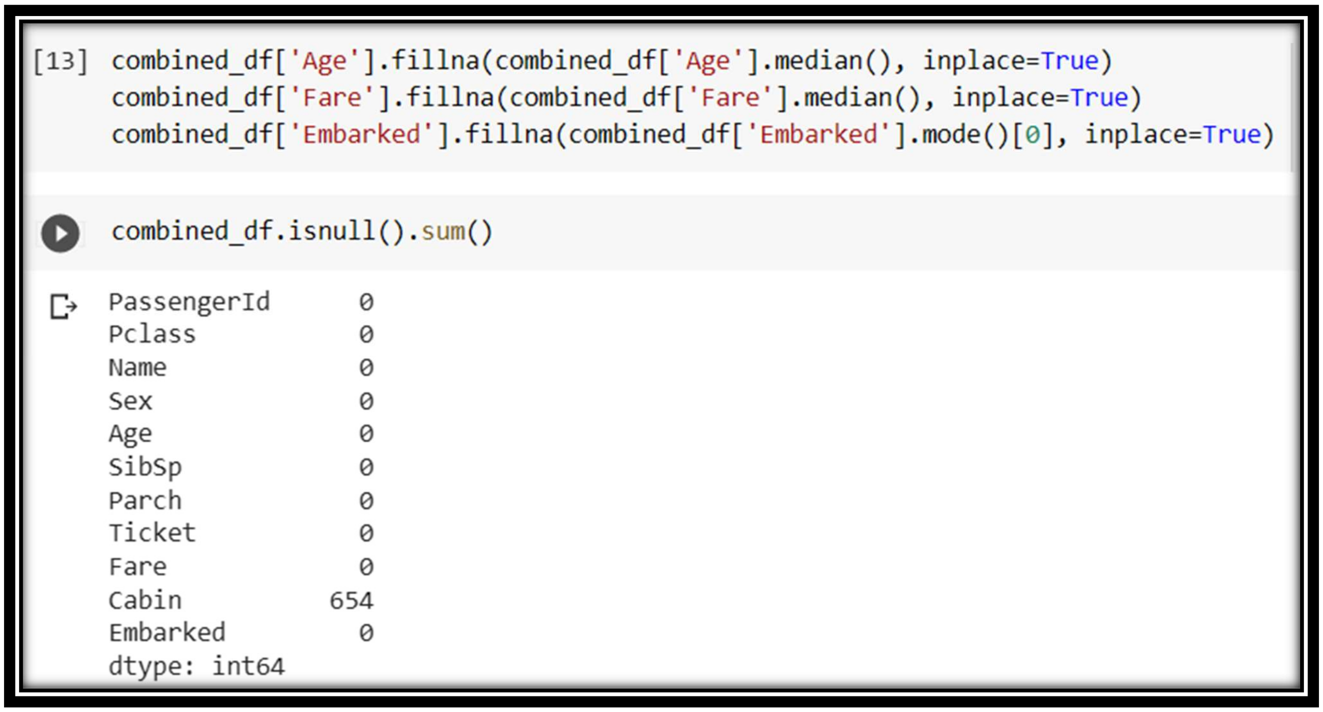


```
train_df.isnull().sum()
```

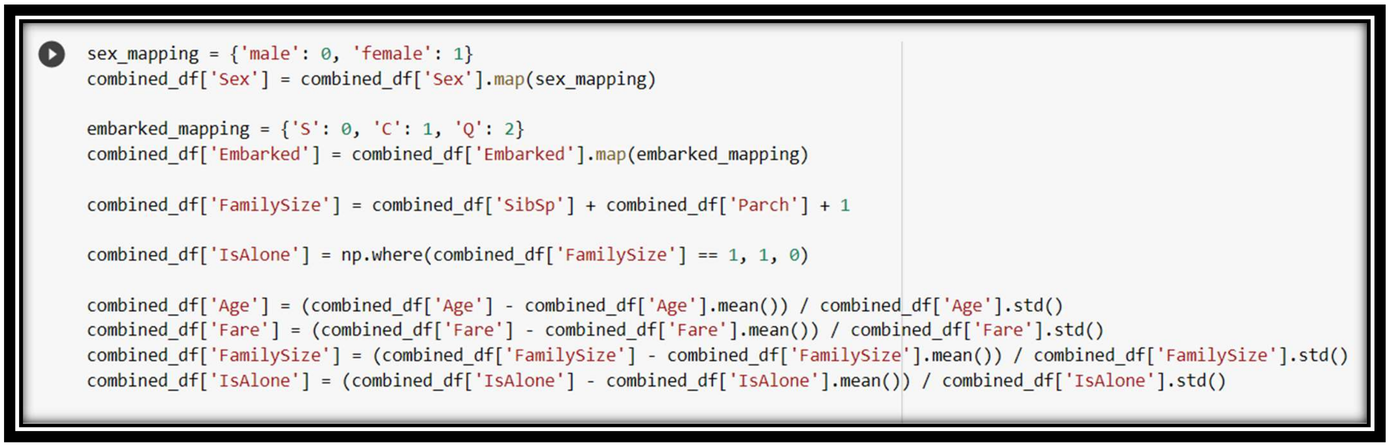
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin          687
Embarked         2
dtype: int64
```

```
train_df.Cabin = train_df.Cabin.fillna("unknown")
print(train_df.isnull().sum())
```

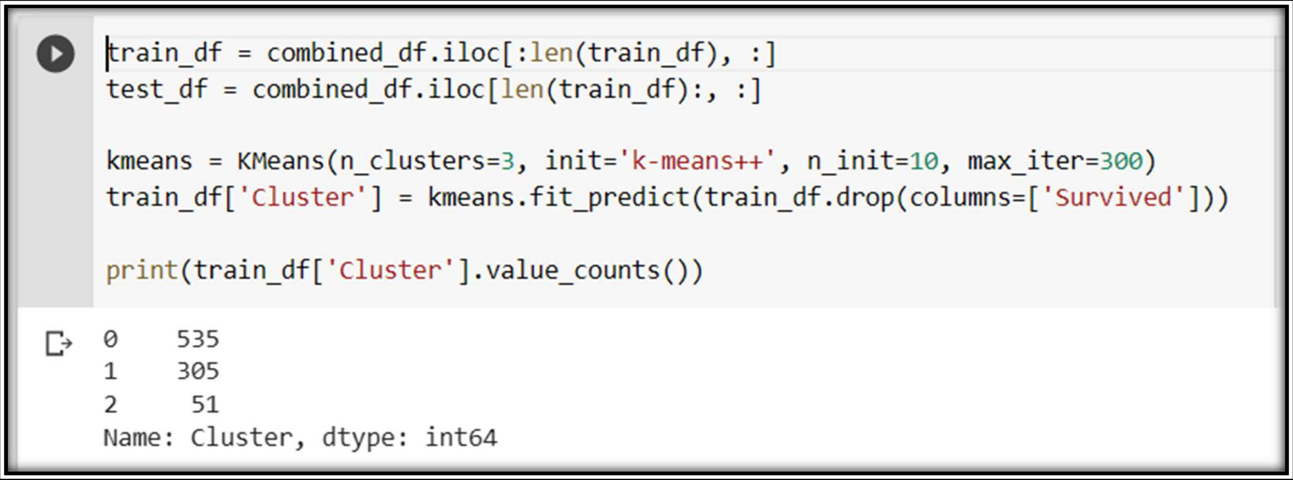
```
PassengerId      0
Survived         0
Pclass           0
Name             0
Sex              0
Age            177
SibSp            0
Parch            0
Ticket           0
Fare             0
Cabin            0
Embarked         2
dtype: int64
```



```
[13] combined_df['Age'].fillna(combined_df['Age'].median(), inplace=True)
     combined_df['Fare'].fillna(combined_df['Fare'].median(), inplace=True)
     combined_df['Embarked'].fillna(combined_df['Embarked'].mode()[0], inplace=True)

     combined_df.isnull().sum()

     PassengerId      0
     Pclass           0
     Name             0
     Sex              0
     Age              0
     SibSp            0
     Parch            0
     Ticket           0
     Fare             0
     Cabin          654
     Embarked         0
     dtype: int64
```



```
sex_mapping = {'male': 0, 'female': 1}
combined_df['Sex'] = combined_df['Sex'].map(sex_mapping)

embarked_mapping = {'S': 0, 'C': 1, 'Q': 2}
combined_df['Embarked'] = combined_df['Embarked'].map(embarked_mapping)

combined_df['FamilySize'] = combined_df['SibSp'] + combined_df['Parch'] + 1

combined_df['IsAlone'] = np.where(combined_df['FamilySize'] == 1, 1, 0)

combined_df['Age'] = (combined_df['Age'] - combined_df['Age'].mean()) / combined_df['Age'].std()
combined_df['Fare'] = (combined_df['Fare'] - combined_df['Fare'].mean()) / combined_df['Fare'].std()
combined_df['FamilySize'] = (combined_df['FamilySize'] - combined_df['FamilySize'].mean()) / combined_df['FamilySize'].std()
combined_df['IsAlone'] = (combined_df['IsAlone'] - combined_df['IsAlone'].mean()) / combined_df['IsAlone'].std()
```



```
train_df = combined_df.iloc[:len(train_df), :]
test_df = combined_df.iloc[len(train_df):, :]

kmeans = KMeans(n_clusters=3, init='k-means++', n_init=10, max_iter=300)
train_df['Cluster'] = kmeans.fit_predict(train_df.drop(columns=['Survived']))

print(train_df['Cluster'].value_counts())
```

```
0    535
1    305
2     51
Name: Cluster, dtype: int64
```