

# EXPOSYS DATA LABS INTERNSHIP REPORT

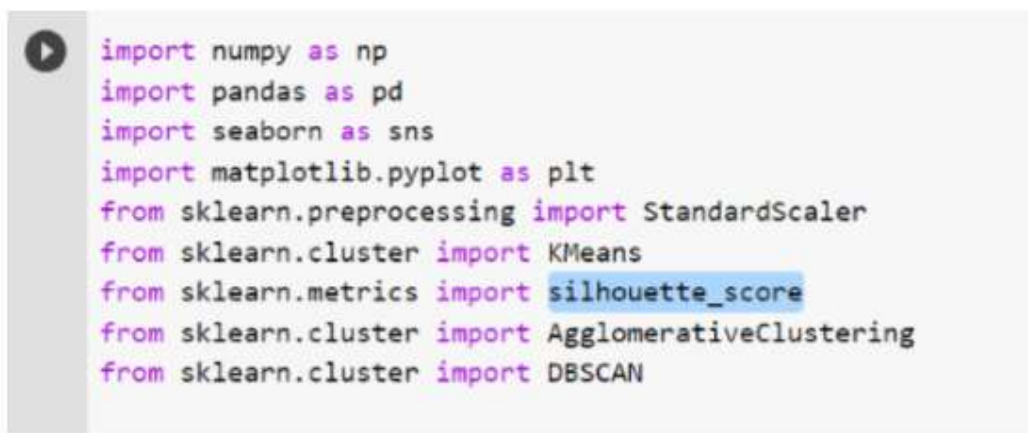
## INTRODUCTION

The following report is on segmentation of customers using given data of like Age, Gender, Average Salary, their average expenditure. As the data does not have results or we can say that given data does not contain a label column so we have to use unsupervised learning in order to assign customers different clusters.

The following report contains insight on data that, analysis of data, manipulation of data, and several applied algorithms for clustering to obtain the best result possible. We also check that what number of clusters to obtain the best results.

## LIBRARIES USED

Python libraries for data manipulation and analysis have been used like matplotlib, pandas, NumPy, seaborn. Also, several libraries of scikit learn have been used for making machine learning models like sklearn.metrics, sklearn.cluster.KMean, sklearn. AgglomerativeClustering, sklearn.DBSCAN, sklearn.silhouette\_score, sklearn.preprocessing.StandardScaler.



```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans
from sklearn.metrics import silhouette_score
from sklearn.cluster import AgglomerativeClustering
from sklearn.cluster import DBSCAN
```

## ABOUT DATA

The given data is on the history of customers coming to the mall, the data contains information about their age, gender, their avg. income, avg. Expenditure etc. Data contains 5 columns and 200 rows, all values are int type except leaving gender column where values are of string type but we convert them too to int type to use them in our model, the data does not have any missing entry.

```
df.info()

<class 'pandas.core.frame.DataFrame'>
Int64Index: 200 entries, 1 to 200
Data columns (total 4 columns):
#   Column                Non-Null Count  Dtype
---  -
0   Gender                 200 non-null   object
1   Age                    200 non-null   int64
2   Annual Income (k$)     200 non-null   int64
3   Spending Score (1-100) 200 non-null   int64
dtypes: int64(3), object(1)
memory usage: 7.8+ KB
```

## DATA ANALYSIS

Most of the costumers are aged less than 60 as we find percent of costumers aged more than 60 than we find out it to be 8.5%.

```
df[df['Age']>60].shape[0]/df.shape[0]*100

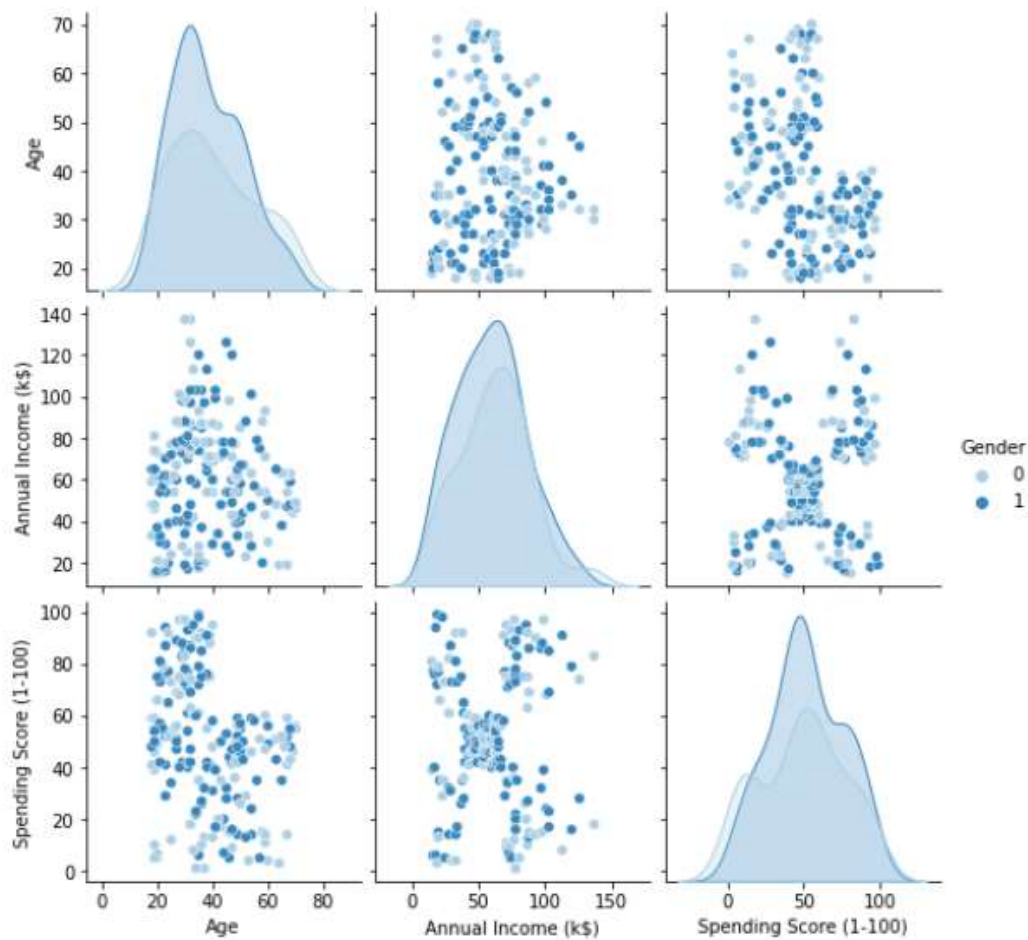
8.5
```

When we try to find average spending score of costumers than it comes out to be near 50 and we does not derive any main conclusion.

```
[58]
df['Spending Score (1-100)'].mean()

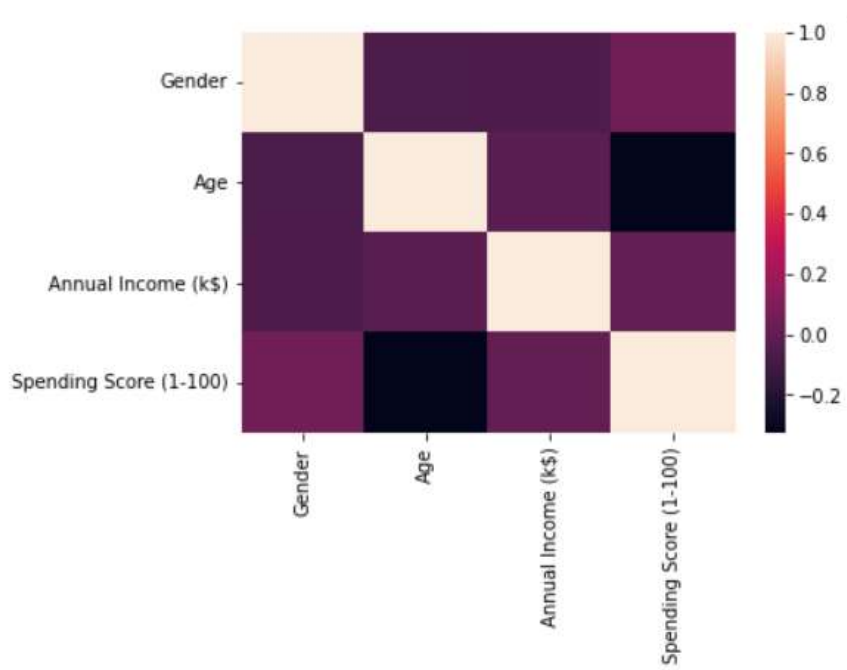
50.2
```

Next, we plot graphs between all the columns to find if any of them have any strong correlation keeping hue as gender which will help us to analyse between two genders.

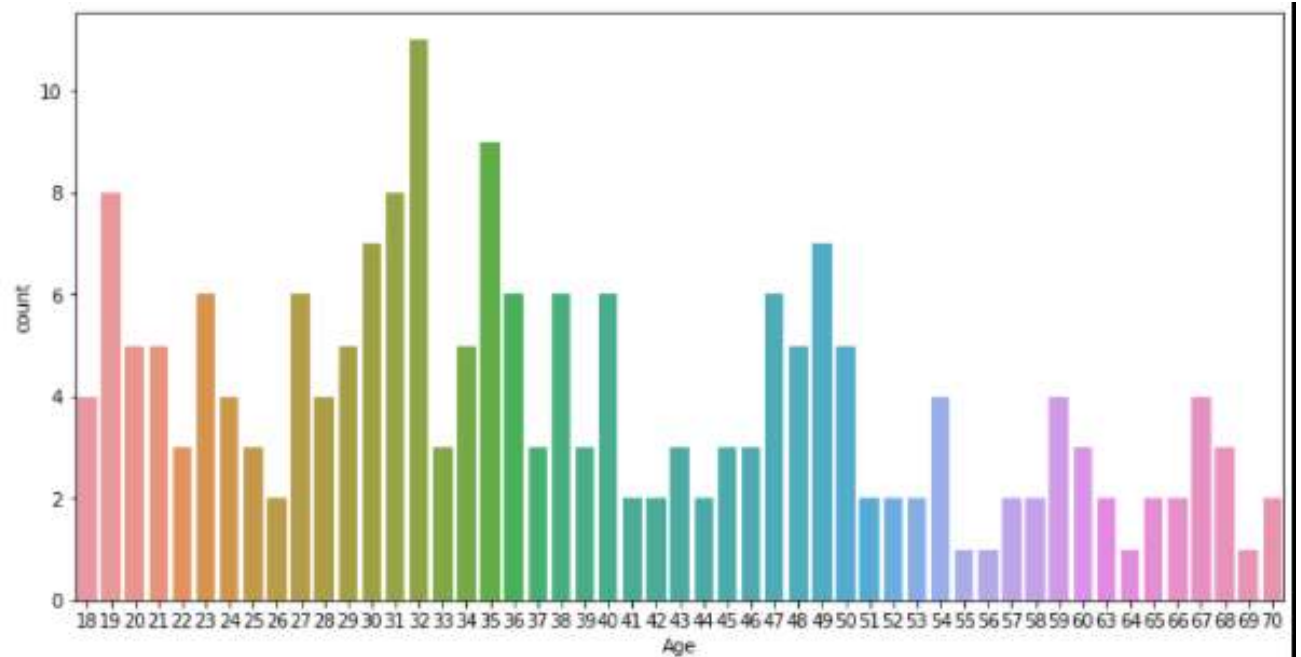


We do not find any strong correlation between any two columns.

Now, let us actually see the values that how much the values of different columns are related by plotting a heat map with a scale.

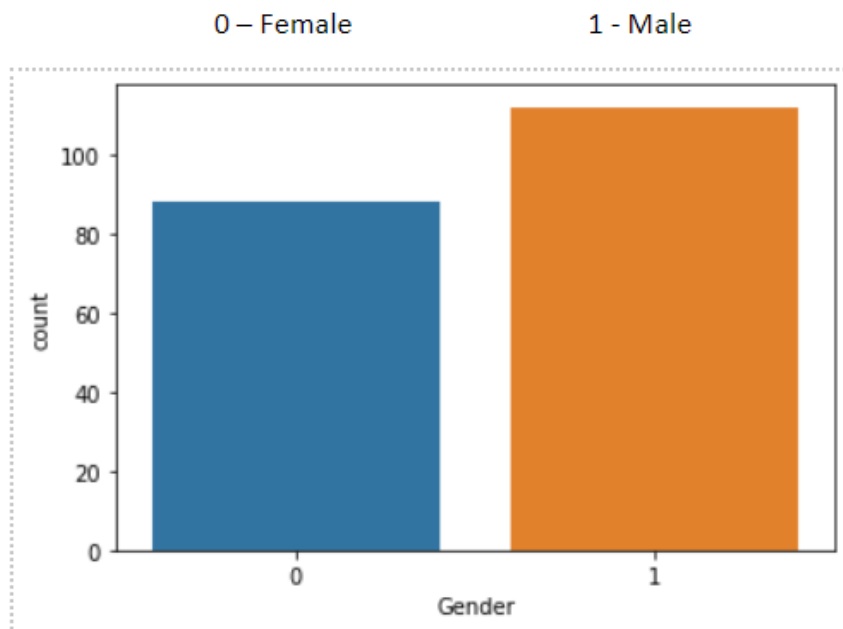


Let us, see the distribution of people of different age groups by plotting a count plot for age column

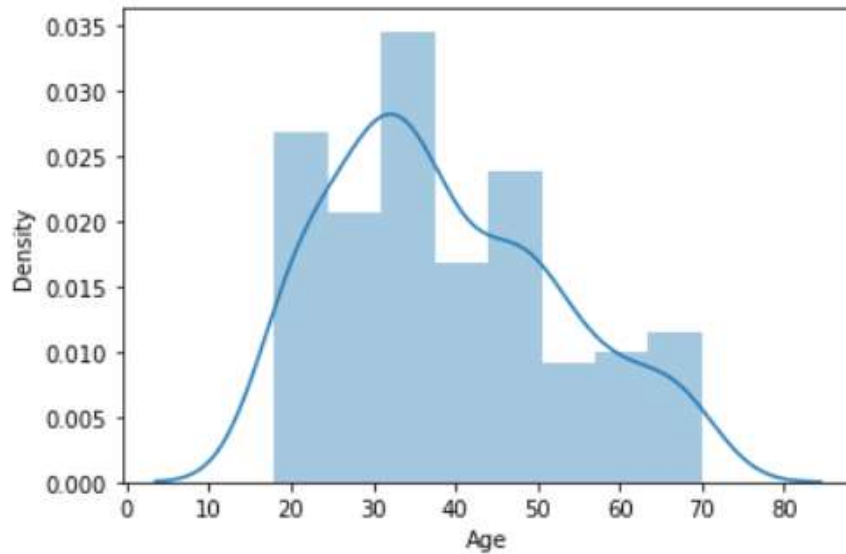


We see a peak at the age group of 32 describing that age group of 32 have max people.

Now let us see how data is distributed between different genders by plotting a count plot on genders column.

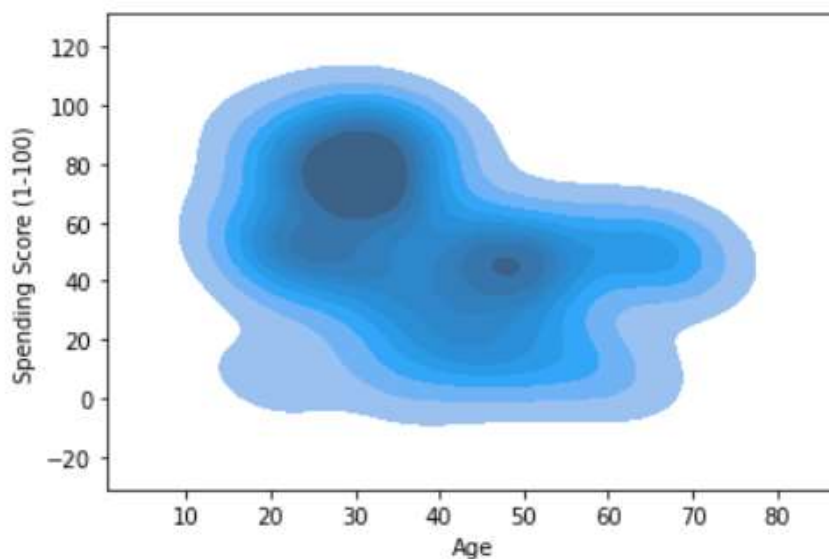


Also, lets plot a distplot to look more closely distribution of customers in different age groups.



Most of the costumers are from the age group of 32-38 which is almost 35% of the total costumers.

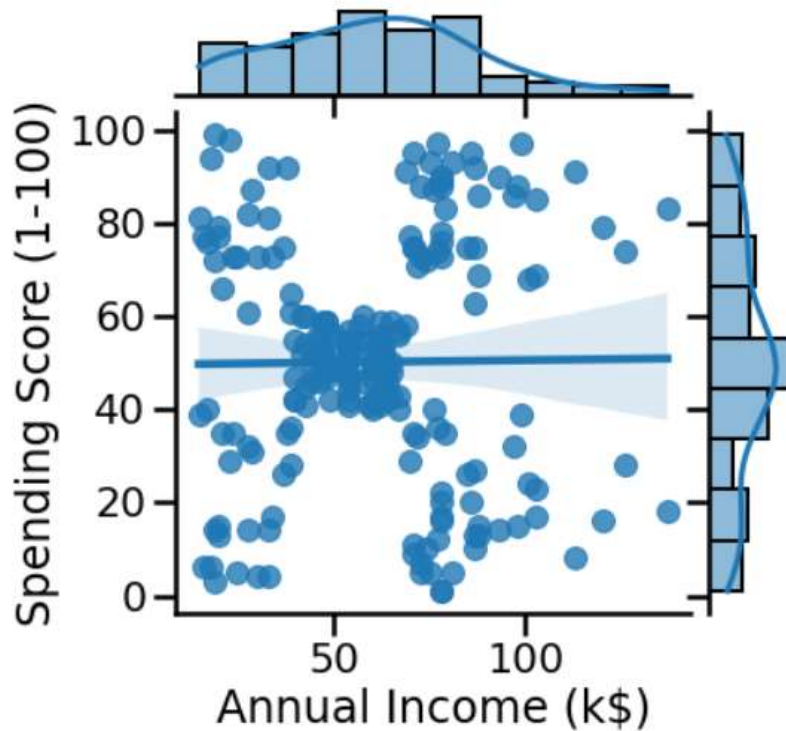
Now, let's try to see if there's relation between age and their average expenditure by plotting a kde plot between them.



So, we see observe that when a person reaches an age of 30–35 he is most settled and spends most than other age groups.

From here we get a relation between age and spending score, there's two more columns in which we can see relation if there's any which are their income and spending score.

We observe these two columns by plotting a joint plot among these.



We don't see any relation among them so it is not very likely that if a person earns more he would spend more.

## PREDICTION

We have used here three different types of models to make our predictions which are KMean, AgglomerativeClustering, and DBSCAN.

1. Our first model here would be KMeans

-> We import it by `sklearn.cluster.KMeans`

-> Next, our task is to choose number of clusters so we run a loop which chooses the different value for number of clusters.

-> For, each model we build find our prediction and then calculate silhouette score for our prediction.

-> The Silhouette Coefficient is calculated using the mean intra-cluster distance ( $a$ ) and the mean nearest-cluster distance ( $b$ ) for each sample. The Silhouette Coefficient for a sample is  $(b - a) / \max(a, b)$ .

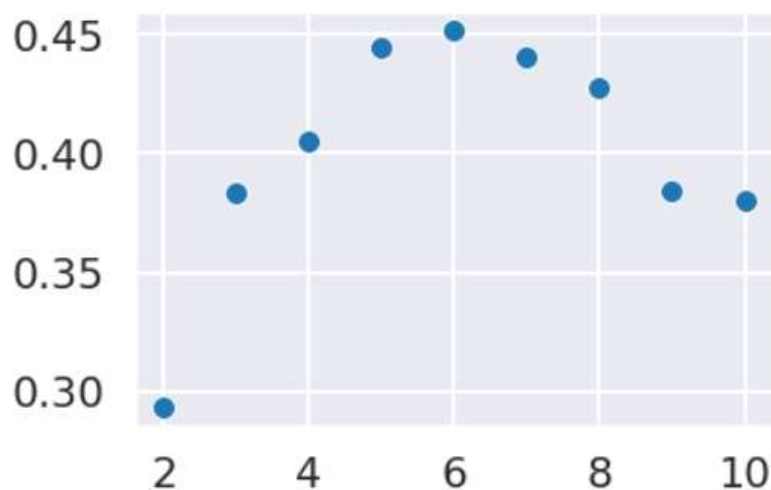
-> 1 is the best value for silhouette score and -1 is the worst.

-> We find this list of scores after running through the loop.

```
[15]
kmean_scores = []
for i in range(2,11):
    model = KMeans(n_clusters=i)
    model.fit_predict(df)
    kmean_scores.append(silhouette_score(df,model.labels_))
x_axis = [x for x in range(2,11)]
kmean_scores
```

```
[0.29307334005502633,
0.383798873822341,
0.4052954330641215,
0.44482259384548795,
0.45205475380756527,
0.4409411333609709,
0.427541566977401,
0.3846704810869784,
0.38002849865056265]
```

-> Now, let's plot a graph to see when do we reach max silhouette score.



-> Seeing, this graph we see at 6 we obtain max value which means if we take number of clusters to be 6 then we achieve max score which is .44.

So, we obtain our KMeans model and set it's hyperparameter `n_clusters` to be 6 to attain maximum accuracy.

2. Our, Next Model is AgglomerativeClustering.

-> First, we import it by `sklearn.cluster.AgglomerativeClustering`.

-> Next, here also we do same by traversing through the loop for giving different values to the number of clusters and obtain silhouette scores.

Again, we plot graph between different number of clusters and silhouette scores.

```
agglo_scores = []
for i in range(2,11):
    model = AgglomerativeClustering(n_clusters = i)
    y = model.fit_predict(df)
    agglo_scores.append(silhouette_score(df,y))
agglo_scores
```

```
[0.29916602844367,
 0.3812859576468096,
 0.4102091359195185,
 0.43997527212476695,
 0.4428008535928764,
 0.42593881050851595,
 0.4223297424559345,
 0.37773214836148283,
 0.3612629591916317]
```

Again, we plot graph between different number of clusters and silhouette scores.



-> Again, we get maximum score for 6 number of clusters and a score of .42 which is less than KMeans.

3. Moving on to our next model which is DBSCAN.

-> First, we import model by sklearn.cluster.DBSCAN.

-> Here, instead of choosing values for number of clusters our hyperparameter would be eps value and we traverse through the different value of eps and obtain these different scores.



```

dbscan_score = []
for i in range(11,20):
    model_dbscan = DBSCAN(eps=i,min_samples = 2)
    y_dbscan = model_dbscan.fit_predict(df)
    dbscan_score.append(silhouette_score(df,y_dbscan))
dbscan_score

```

```

[0.07747245359486282,
 0.08656096770197945,
 0.01806483728401882,
 0.21265289538202517,
 0.2548540886299427,
 0.30440237856418906,
 0.30440237856418906,
 0.39036425615520437,
 0.39036425615520437]

```

->Let us plot graph between eps values and scores.

->So, after observing we do not find any good accuracy which was found max to be .39 which is lesser than for kmeans and Agglomerative.

So, our final model is kmeans with 6 to be the number of clusters.

MODEL = KMeans(n\_clusters = 6)

```

final_model = KMeans(n_clusters = 6)
Y = final_model.fit_predict(df)
df['results']=Y
df

```

CustomerID	Gender	Age	Annual Income (k\$)	Spending Score (1-100)	results
1	0	19	15	39	4
2	0	21	15	81	5
3	1	20	16	6	4
4	1	23	16	77	5
5	1	31	17	40	4
...	...	...	...	...	...
196	1	35	120	79	2
197	1	45	126	28	3
198	0	32	126	74	2
199	0	32	137	18	3
200	0	30	137	83	2