Market Segmentation on JavaScript Job market

At *Feynn Labs*

April 2022

By *: Vraj pandya,*

Ashish Raghani,

Noel Prakash and

Sriram Koyalkar.

FeyNN Labs
EXPERIMENT WITH YOUR KNOWLEDGE

Index.

Abstract

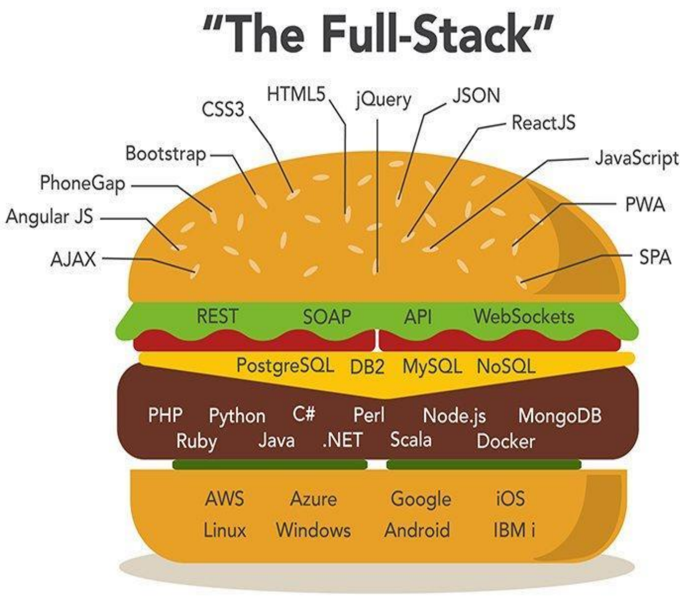
Segmentation is when you isolate potential target audiences to determine which ones will bring the highest return on your marketing efforts. Segmentation is based on specific criteria related to an individual’s age, income, subjects of interest and behaviours. When you segment various markets, you learn more about their core values and what will ultimately attract them to your brand. The difference between conventional market segmentation methods and the jobs-based segmentation approach is the primary unit of analysis for grouping customers. For conventional segmentation, the primary unit of analysis is the attributes of customers’ themselves. The primary unit of analysis for jobs-based segmentation is a job that customers are trying to get done. The conventional definition of a market is based on the product and service categories defined by solution providers. Jobs Theory, on the other hand, defines a market as an aggregation of all available solutions, both provider and non-provider, that customers regard as being able to satisfy their needs with respect to getting a job done. Job segmentation gives a company a significant advantage over competitors because they can anticipate the value that customers want—even before customers are aware of certain needs. A company can quickly and efficiently enhance their existing offerings and create new offerings that can satisfy customer needs better than competitive alternatives at the lowest possible cost.

Introduction

As we know, the job market in the world for full-stack (Full stack engineers are software engineers who control both backend and frontend for in this case, websites/web-applications) engineers has been developed and since then it's been growing steadily.

As of 2022, the need of a full-stack engineer in India has risen to almost 30% since the past year as India has made business of approximately 135 billion USD outsourcing for the companies. Here, Front-end can be used for designing the website as well as making it responsive and such. Backend can be used to connect the website to the server etc. Front end uses JS libraries such as react.js where as backend uses node.js to connect to servers such as mongo-Db or SQL server.

Thus, this Market Segmentation is based on finding companies for a Java-script developer who needs a job in Front-end, Back-end as well as Full-stack areas. The data would be based on the factors such as geographic factors, socio-demographic factors, psychographic and behavioural factors.



Data Collection

The data here was used from multiple sources. The ready and clean dataset regarding the jobs, companies and the job-titles were extracted from a dataset available in Kaggle. The rest of the columns like salary expectations were found by fermi estimation as it was based on a lot of factors already available in the dataset. The remaining of the columns such as “No. of vacancies” were based on the scraping from the websites such as Glassdoor and Indeed.

The dataset consisted of several columns which were reduced down to 7 columns and had 100 rows.

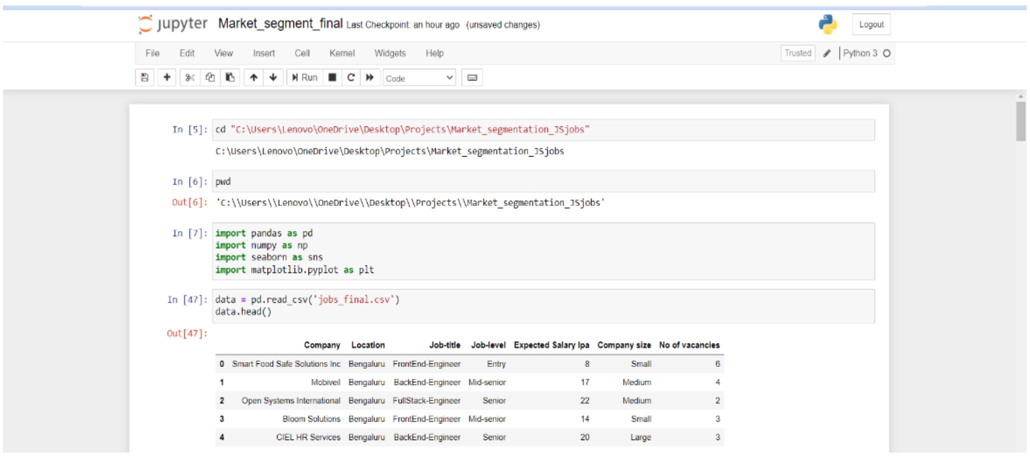
The factors were based as the following:

For geographic: Company’s location which consisted of five cities.

For socio-demographic: We had No of vacancies, Job-level which consisted of three values which were Entry, Mid-senior and Senior and the company size.

For psychographic: There is a column which the expected salaries which would be affected by almost all the factors listed.

For Behavioural: There is a column which is the Job-title which would specify the type of the job needed viz. Frontend engineer, backend engineer or a Full-stack engineer.

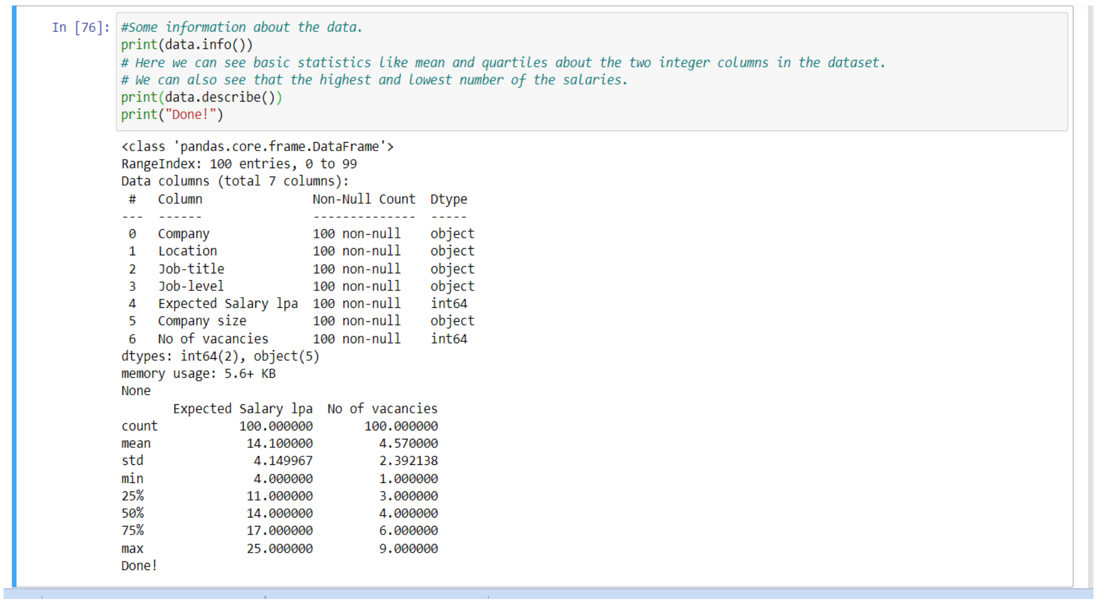

jupyter Market_segment_final Last Checkpoint: an hour ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3
Run
Code

```
In [5]: cd "C:\Users\Lenovo\OneDrive\Desktop\Projects\Market_segmentation_JSjobs"
        C:\Users\Lenovo\OneDrive\Desktop\Projects\Market_segmentation_JSjobs

In [6]: pwd
Out[6]: 'C:\\Users\\Lenovo\\OneDrive\\Desktop\\Projects\\Market_segmentation_JSjobs'

In [7]: import pandas as pd
        import numpy as np
        import seaborn as sns
        import matplotlib.pyplot as plt

In [47]: data = pd.read_csv('jobs_final.csv')
         data.head()

Out[47]:
```

	Company	Location	Job-title	Job-level	Expected Salary lpa	Company size	No of vacancies
0	Smart Food Safe Solutions Inc	Bengaluru	FrontEnd-Engineer	Entry	8	Small	6
1	Mobiveil	Bengaluru	BackEnd-Engineer	Mid-senior	17	Medium	4
2	Open Systems International	Bengaluru	FullStack-Engineer	Senior	22	Medium	2
3	Bloom Solutions	Bengaluru	FrontEnd-Engineer	Mid-senior	14	Small	3
4	CIEL HR Services	Bengaluru	BackEnd-Engineer	Senior	20	Large	3

The above figure states the head of the dataset.

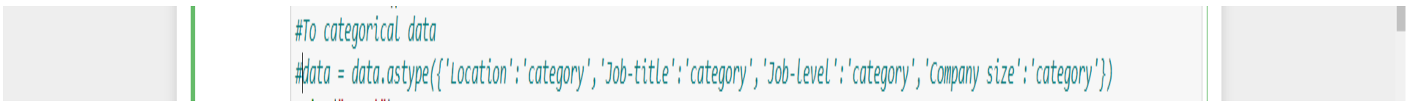
This was some information about the dataset that was extracted using Pandas (a Python Library).



```
In [76]: #Some information about the data.
         print(data.info())
         # Here we can see basic statistics like mean and quartiles about the two integer columns in the dataset.
         # We can also see that the highest and lowest number of the salaries.
         print(data.describe())
         print("Done!")

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 100 entries, 0 to 99
         Data columns (total 7 columns):
          #   Column              Non-Null Count  Dtype
         ---  ------              --------------  -----
          0   Company             100 non-null    object
          1   Location            100 non-null    object
          2   Job-title           100 non-null    object
          3   Job-level           100 non-null    object
          4   Expected Salary lpa  100 non-null    int64
          5   Company size        100 non-null    object
          6   No of vacancies     100 non-null    int64
         dtypes: int64(2), object(5)
         memory usage: 5.6+ KB
         None
                Expected Salary lpa  No of vacancies
         count           100.000000       100.000000
         mean             14.100000         4.570000
         std               4.149967         2.392138
         min               4.000000         1.000000
         25%              11.000000         3.000000
         50%              14.000000         4.000000
         75%              17.000000         6.000000
         max              25.000000         9.000000
         Done!
```

Here, data.info() tells us about the data, what values it is missing, memory used by the data and the type of the data used.

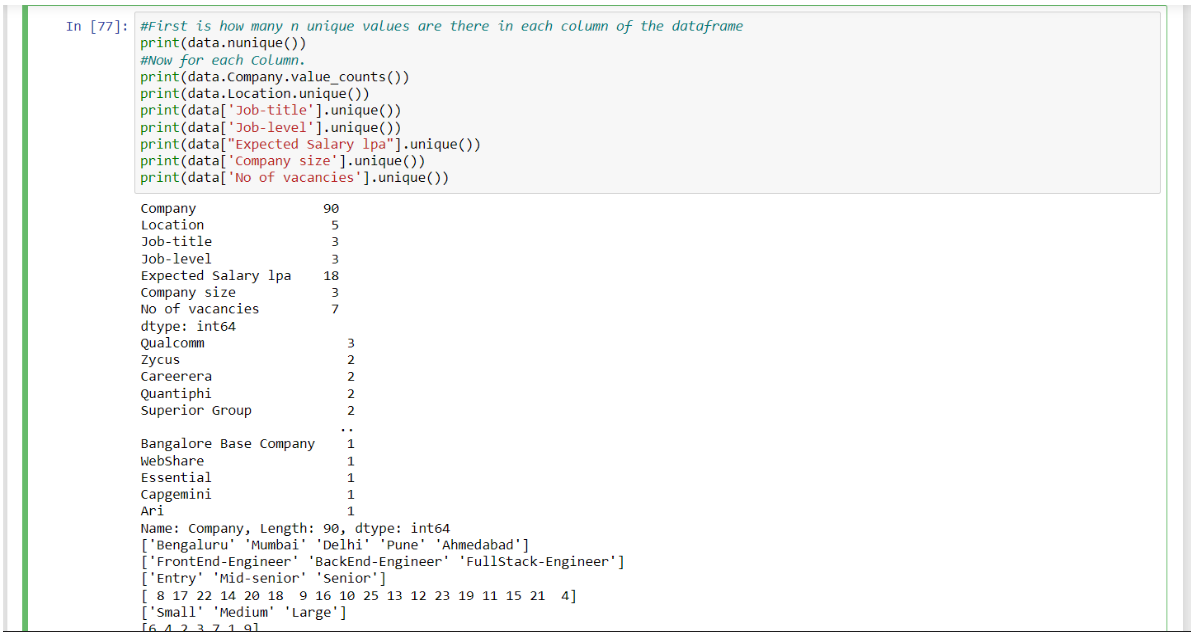
Note: Here, if we convert some columns to categorical dtype using this code the memory would decrease but would cause some problems for the visualizations part. Thus, it would be in constant upgradation.



```
#To categorical data
#data = data.astype({'Location':'category','Job-title':'category','Job-level':'category','Company size':'category'})
```

After, data.describe() gives us the basic statistics summary about the numeric columns of the data which in this case would be Expected Salary and No. of vacancies. The information we get after using data.describe is max values, min values, mean, 25th(50th and the 75th) quartile values and count of the values as well.

After that the final step in data collection process is looking at how many unique values are there in the dataset.



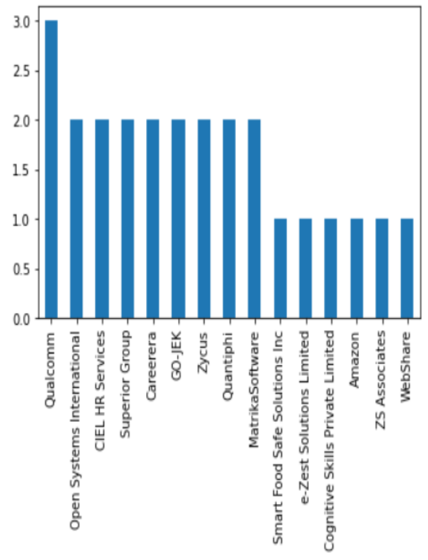
```
In [77]: #First is how many n unique values are there in each column of the dataframe
         print(data.nunique())
         #Now for each Column.
         print(data.Company.value_counts())
         print(data.Location.unique())
         print(data['Job-title'].unique())
         print(data['Job-level'].unique())
         print(data["Expected Salary lpa"].unique())
         print(data['Company size'].unique())
         print(data['No of vacancies'].unique())
```

```
Company                90
Location                5
Job-title               3
Job-level               3
Expected Salary lpa    18
Company size            3
No of vacancies         7
dtype: int64
Qualcomm                  3
Zycus                     2
Careerera                 2
Quantiphi                 2
Superior Group            2
                         ..
Bangalore Base Company    1
WebShare                  1
Essential                 1
Capgemini                 1
Ari                       1
Name: Company, Length: 90, dtype: int64
['Bengaluru' 'Mumbai' 'Delhi' 'Pune' 'Ahmedabad']
['FrontEnd-Engineer' 'BackEnd-Engineer' 'FullStack-Engineer']
['Entry' 'Mid-senior' 'Senior']
[ 8 17 22 14 20 18  9 16 10 25 13 12 23 19 11 15 21  4]
['Small' 'Medium' 'Large']
[6 4 2 3 7 1 9]
```

Here, we can see that
The company column has 90 unique values,
Where as location has 5,
Job-title has 3,
Job-level has 3 as well and
Company size has 3 as well.

Here, in this process we attempt to find the corelations between the columns and extract segments out of it.

```
In [11]: o = df['Company'].value_counts()
         o[:15].plot.bar()

Out[11]: <AxesSubplot:>
```

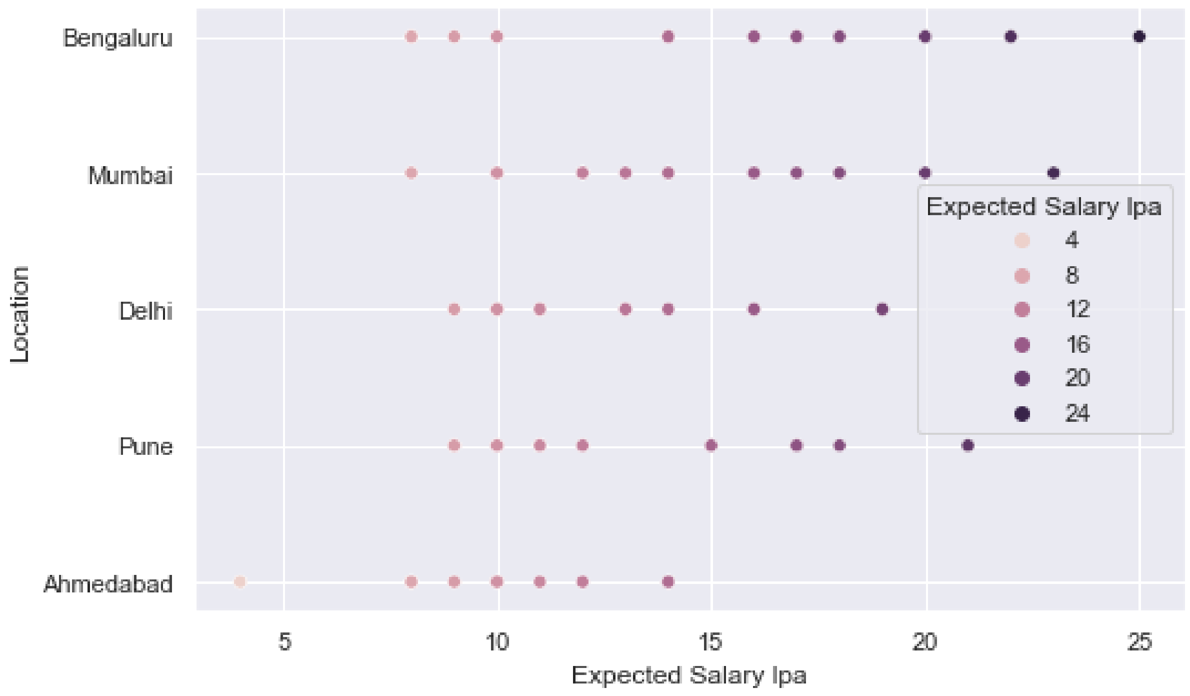


Here, we see that the company Qualcomm has the most number of the values in the dataset (3).

Now to see the histograms of:

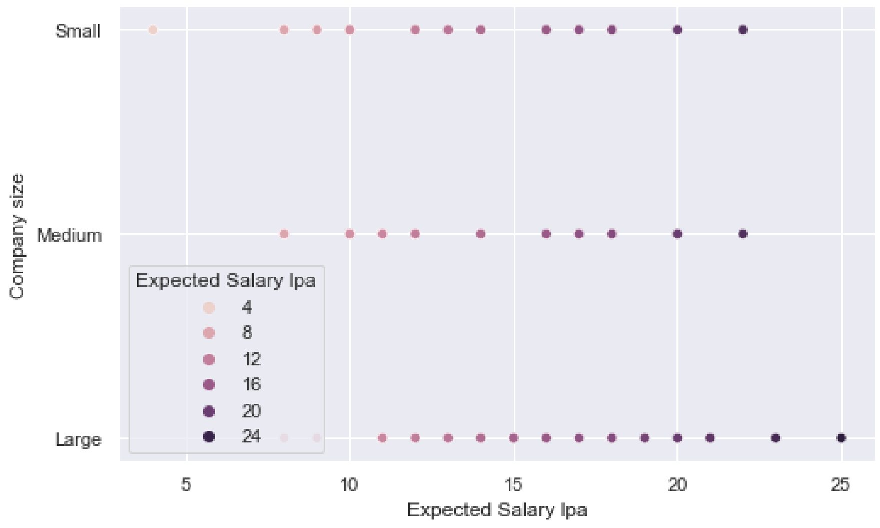
Expected salaries vs Various columns.

1) Expected salaries vs Location



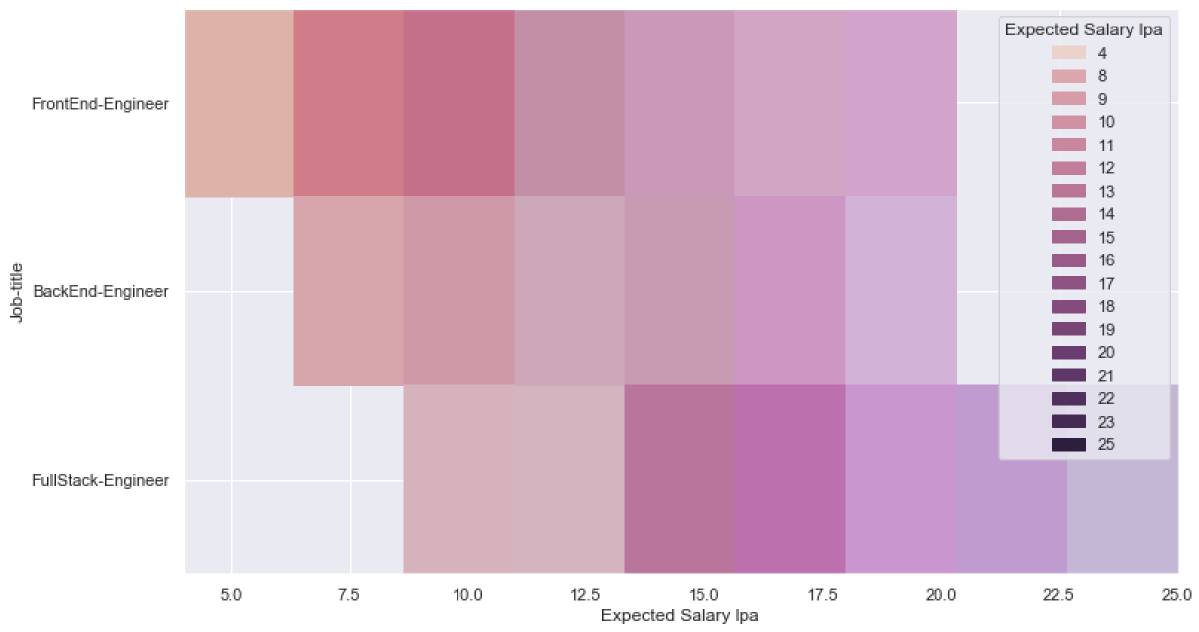
Here, we can clearly see that the Most expected salary is in Bangalore where as the lowest expected salary is in Ahmedabad.

2). Expected Salaries vs Company size.



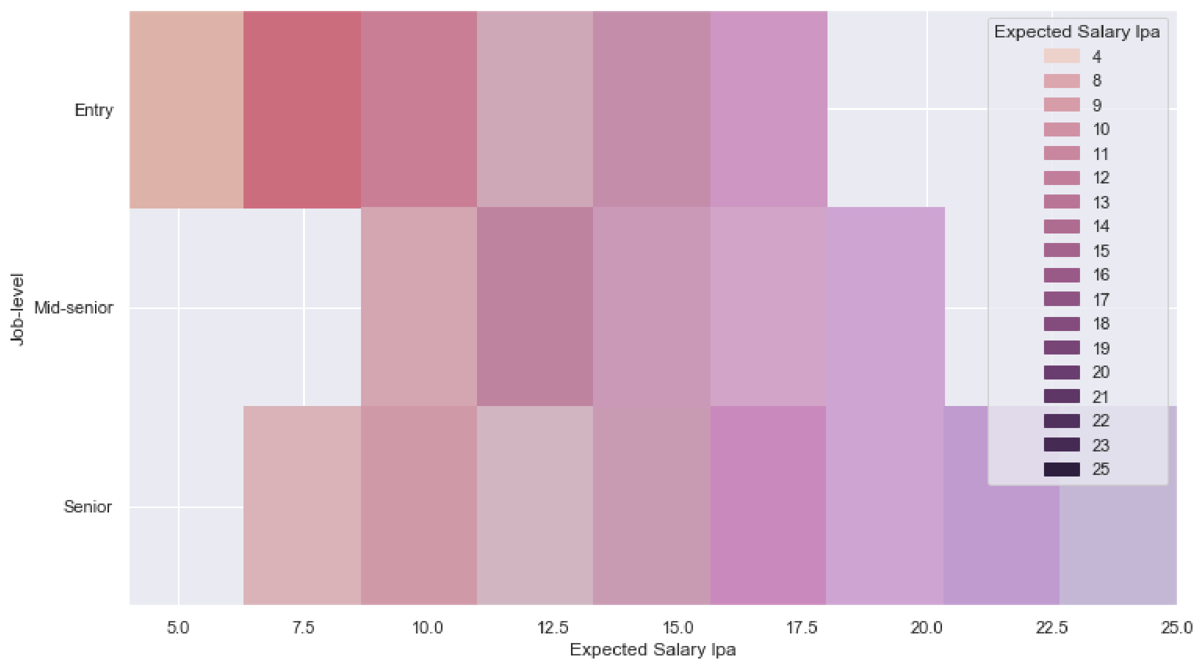
Here, we can see that larger the company size, more the expected salary.

3). Expected salary vs Job title.



Here, we can see that the more expected salary goes towards the Fullstack engineer whereas the lower salary goes towards the Frontend engineer.

4). Expected salary vs Job Level.

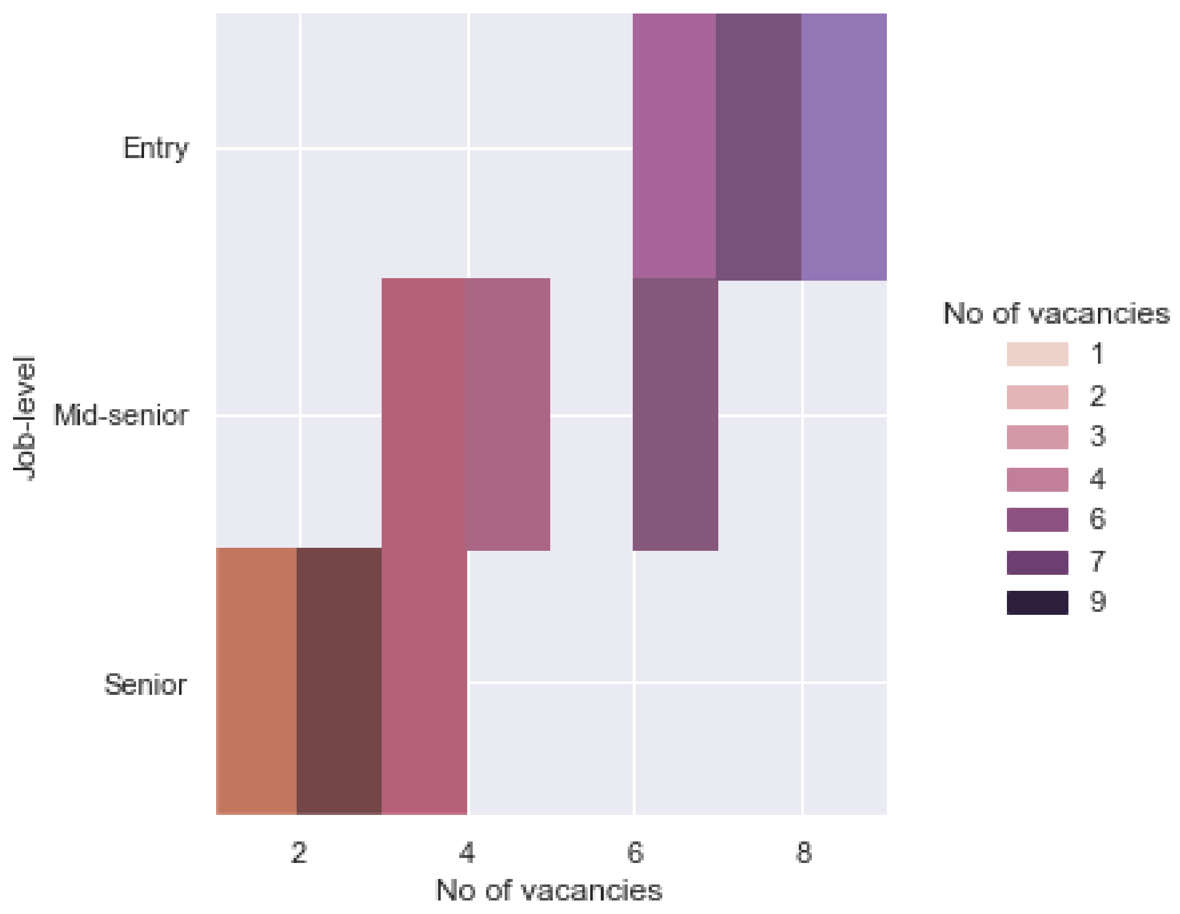


Here, we can see that the more salary is earned by the senior developers than that of the entry level developers.

Now, we'll see

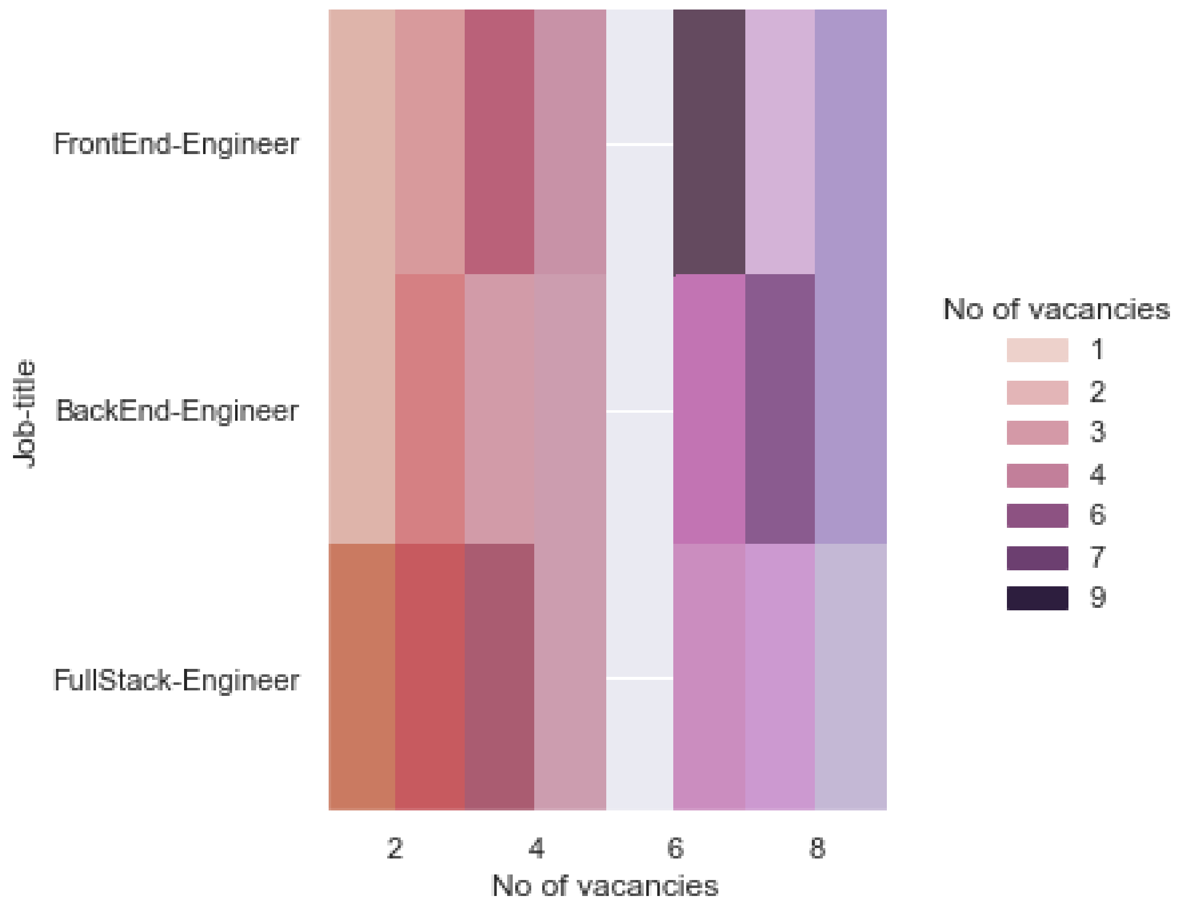
Job vacancies vs Various columns.

1) Locations vs No. of Job Vacancies.



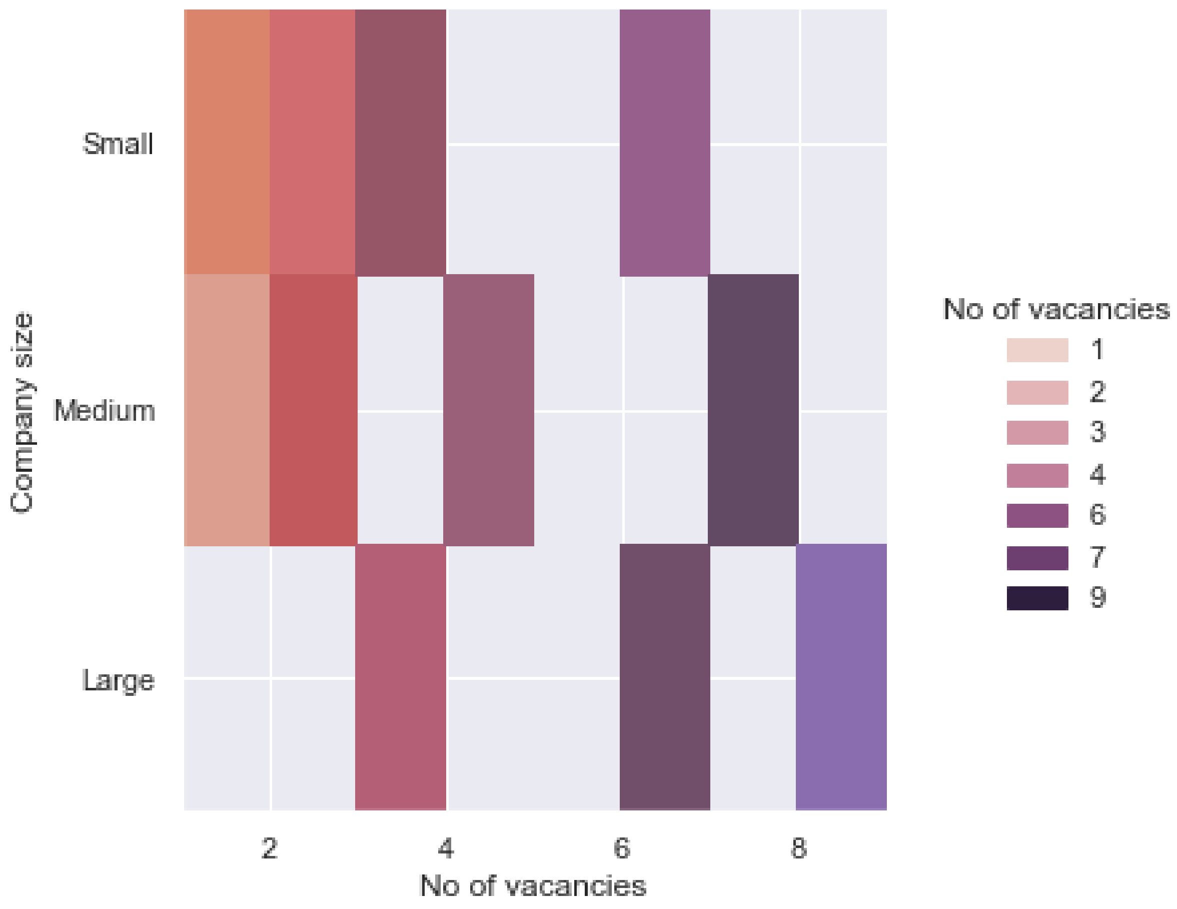
Here, we can see that the no. of vacancies are higher in Entry level and lesser in the senior level positions.

2) Job title vs No. of Job vacancies



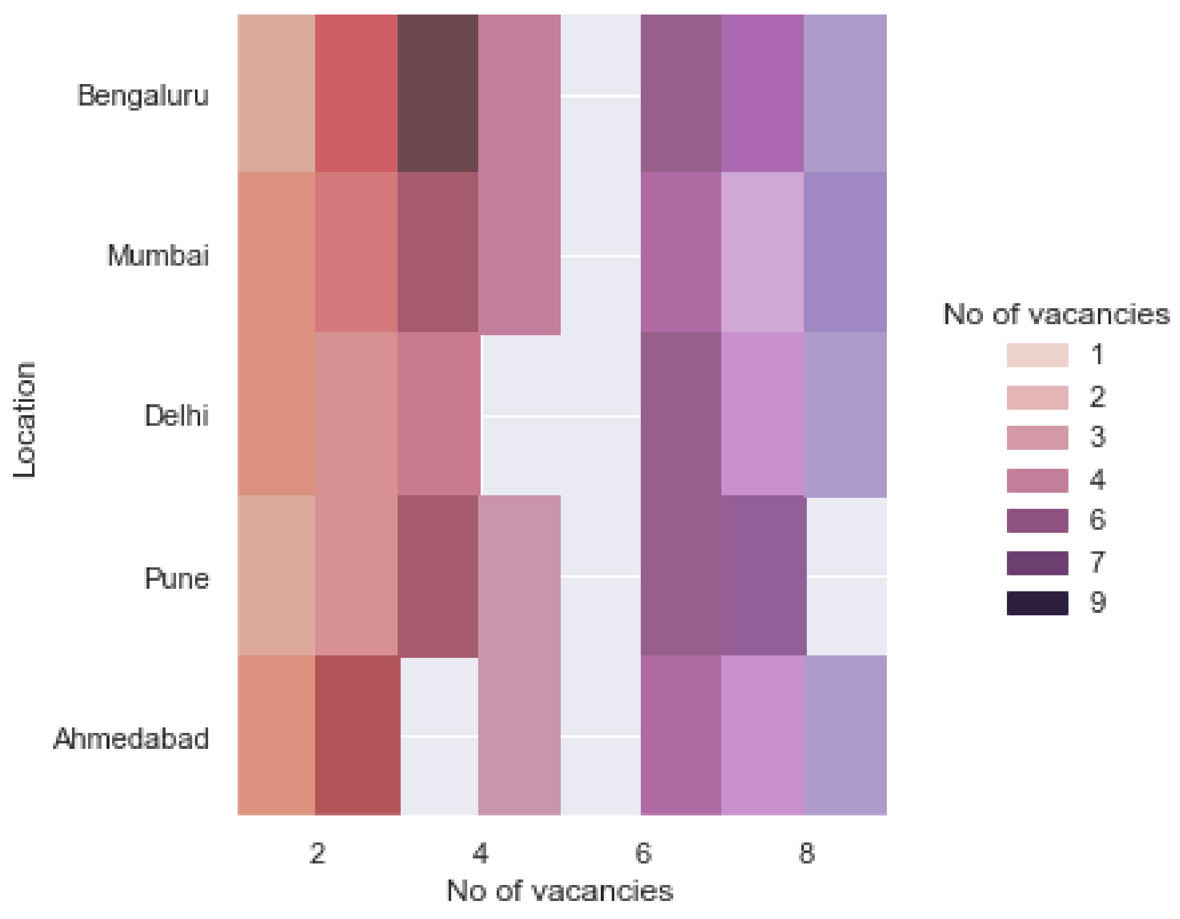
Here, we see that more vacancies are in Front-end position than in the Full-stack position.

3). Company Size vs No of vacancies



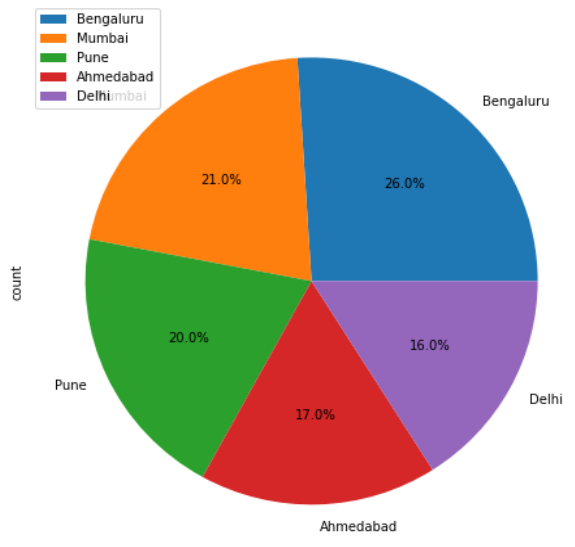
Here, we see that the medium levelled companies cover more segmentation market than the rest.

4) Location vs No of vacancies



Here, we see that most number of vacancies are available in Bangalore, closely followed by Mumbai and least in delhi.

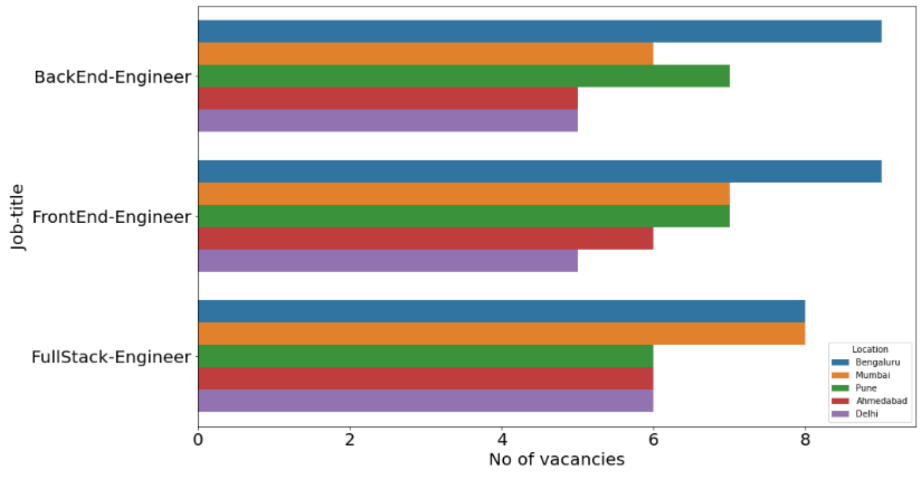
As we can see here too,



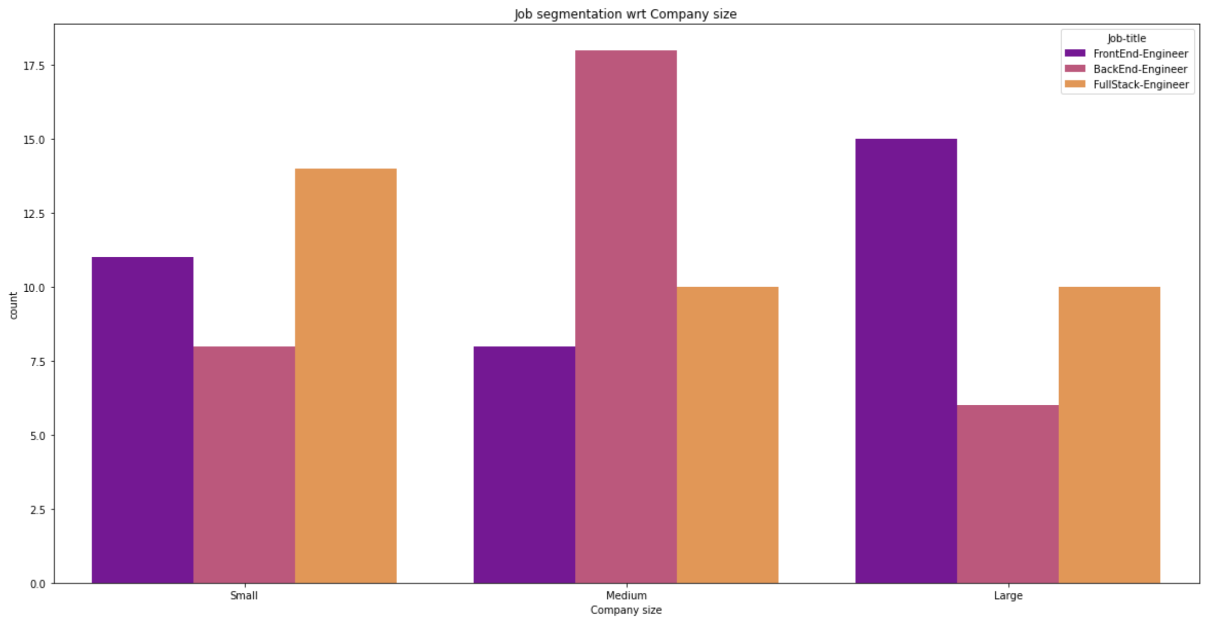
Now, to see each number of positions that are vacant according to the job title and location of where the job-title is based.

Out[23]:

	Job-title	Location	No of vacancies
1	BackEnd-Engineer	Bengaluru	9
6	FrontEnd-Engineer	Bengaluru	9
11	FullStack-Engineer	Bengaluru	8
13	FullStack-Engineer	Mumbai	8
4	BackEnd-Engineer	Pune	7
8	FrontEnd-Engineer	Mumbai	7
9	FrontEnd-Engineer	Pune	7
3	BackEnd-Engineer	Mumbai	6
5	FrontEnd-Engineer	Ahmedabad	6
10	FullStack-Engineer	Ahmedabad	6
12	FullStack-Engineer	Delhi	6
14	FullStack-Engineer	Pune	6
0	BackEnd-Engineer	Ahmedabad	5
2	BackEnd-Engineer	Delhi	5
7	FrontEnd-Engineer	Delhi	5



Now finally, we see the job-titles based on company size.



Here, there are more positions of a full stack engineer in small sized company, backend in medium sized company and front end in the large sized company.

So, this was the data exploration part where we learned about the data as well as well as looked at the possible segmentation which included geographic, socio-graphic, psychographic and behavioural segments in the above visualizations.

K-means Clustering

What is K-means cluster?

A centroid is the imaginary or real location representing the centre of the cluster. In other words, the K-means algorithm identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible.

How the K-means algorithm works

To process the learning data, the K-means algorithm in data mining starts with a first group of randomly selected centroids, which are used as the beginning points for every cluster, and then performs iterative (repetitive) calculations to optimize the positions of the centroids

*It halts creating and optimizing clusters when either:*1. The centroids have stabilized — there is no change in their values because the clustering has been successful.

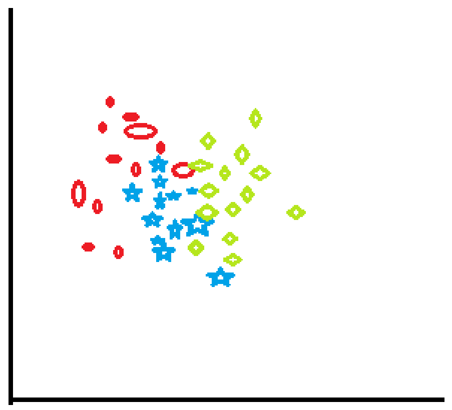


fig 1: before applying k-means clustering

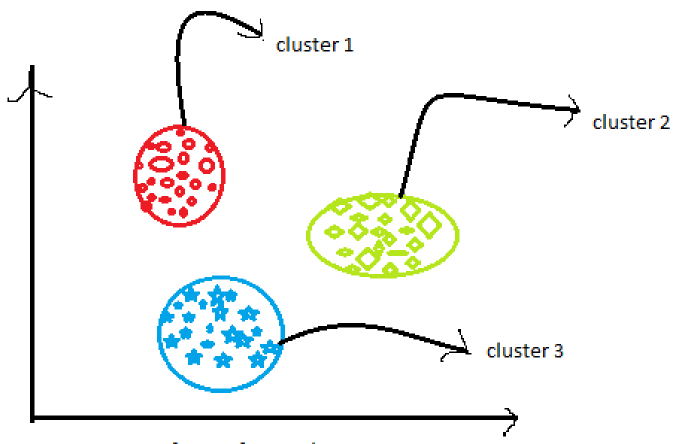
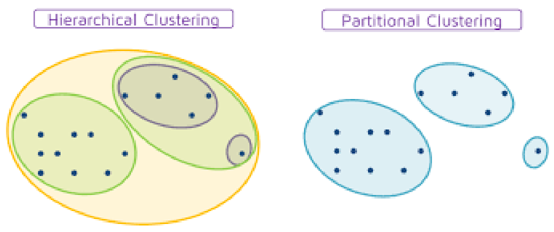
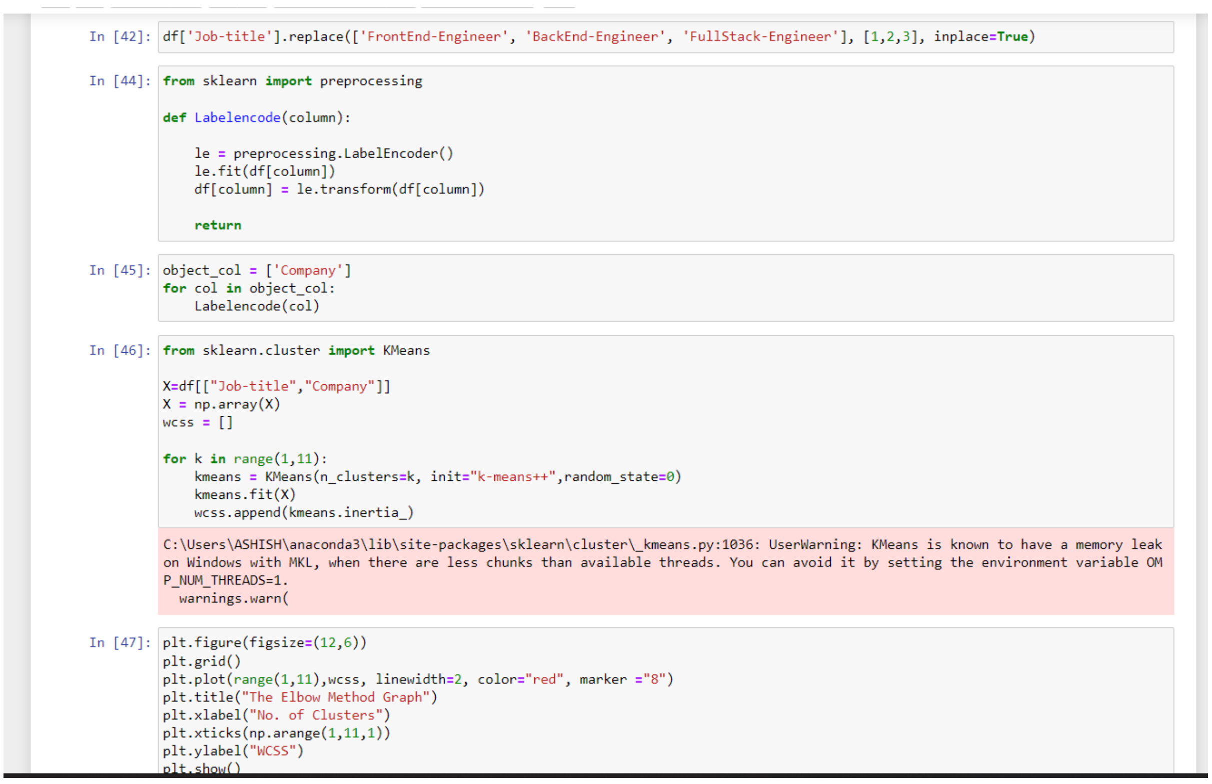


fig 2: After applying K-means clustering

2. The defined number of iterations has been achieved.



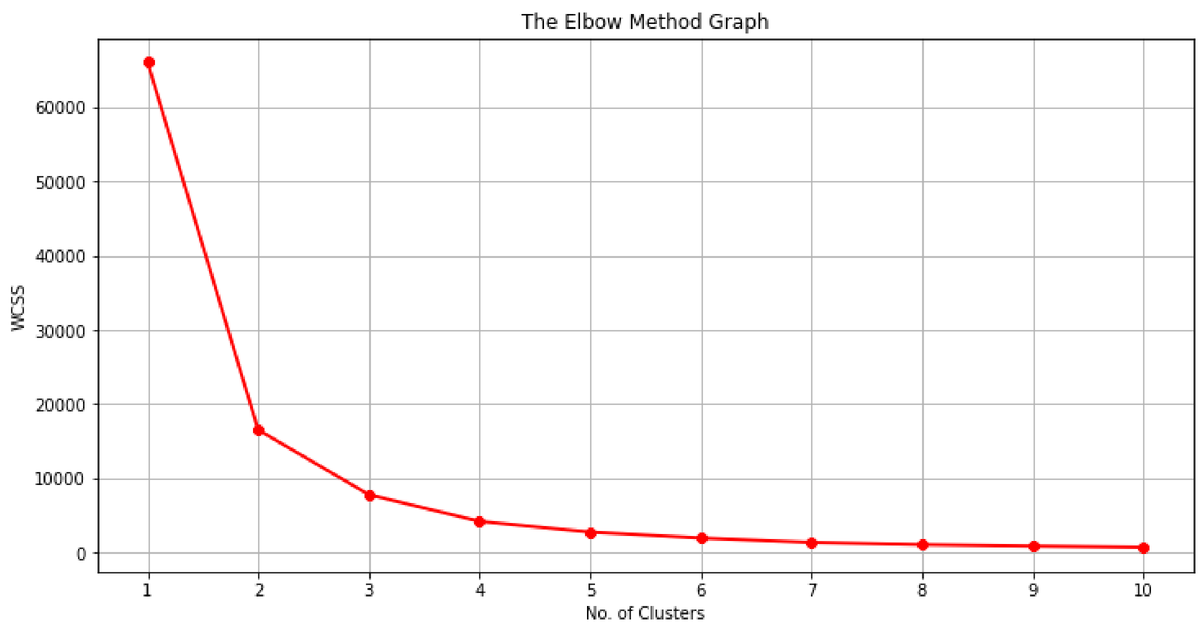
Here, we do the clustering based on Job-title and the company.

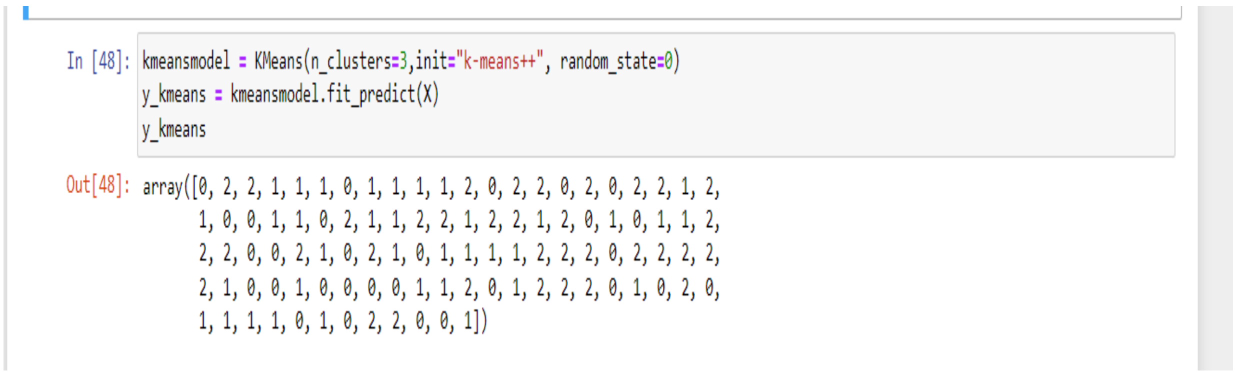


```
In [42]: df['Job-title'].replace(['FrontEnd-Engineer', 'BackEnd-Engineer', 'FullStack-Engineer'], [1,2,3], inplace=True)

In [44]: from sklearn import preprocessing

         def Labelencode(column):

             le = preprocessing.LabelEncoder()
             le.fit(df[column])
             df[column] = le.transform(df[column])

             return

In [45]: object_col = ['Company']
         for col in object_col:
             Labelencode(col)

In [46]: from sklearn.cluster import KMeans

         X=df[["Job-title","Company"]]
         X = np.array(X)
         wcss = []

         for k in range(1,11):
             kmeans = KMeans(n_clusters=k, init="k-means++",random_state=0)
             kmeans.fit(X)
             wcss.append(kmeans.inertia_)

         C:\Users\ASHISH\anaconda3\lib\site-packages\sklearn\cluster\_kmeans.py:1036: UserWarning: KMeans is known to have a memory leak
         on Windows with MKL, when there are less chunks than available threads. You can avoid it by setting the environment variable OM
         P_NUM_THREADS=1.
           warnings.warn(

In [47]: plt.figure(figsize=(12,6))
         plt.grid()
         plt.plot(range(1,11),wcss, linewidth=2, color="red", marker ="8")
         plt.title("The Elbow Method Graph")
         plt.xlabel("No. of Clusters")
         plt.xticks(np.arange(1,11,1))
         plt.ylabel("WCSS")
         plt.show()
```

And here the cluster is done using partitional clustering.

Here, is the elbow graph to show the number of clusters possibly formed using the above variables.





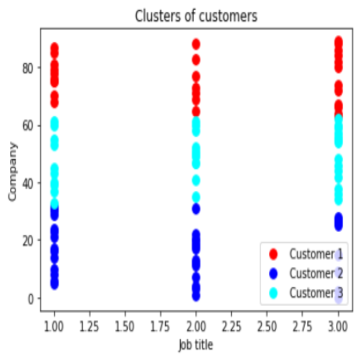
```
In [48]: kmeansmodel = KMeans(n_clusters=3,init="k-means++", random_state=0)
         y_kmeans = kmeansmodel.fit_predict(X)
         y_kmeans

Out[48]: array([0, 2, 2, 1, 1, 1, 0, 1, 1, 1, 1, 2, 0, 2, 2, 0, 2, 0, 2, 2, 1, 2,
                1, 0, 0, 1, 1, 0, 2, 1, 1, 2, 2, 1, 2, 2, 1, 2, 0, 1, 0, 1, 1, 2,
                2, 2, 0, 0, 2, 1, 0, 2, 1, 0, 1, 1, 1, 1, 2, 2, 2, 0, 2, 2, 2, 2,
                2, 1, 0, 0, 1, 0, 0, 0, 0, 1, 1, 2, 0, 1, 2, 2, 2, 0, 1, 0, 2, 0,
                1, 1, 1, 1, 0, 1, 0, 2, 2, 0, 0, 1])
```

Here, the kmeans model predicts the job-titles based on the companies.

(Graph below).

```
In [49]: from mpl_toolkits.mplot3d import Axes3D
```

```
In [51]: plt.scatter(X[y_kmeans == 0,0], X[y_kmeans == 0,1],  c='red', s=60 , label='Customer 1')
         plt.scatter(X[y_kmeans == 1,0], X[y_kmeans == 1,1],  c='blue', s=60 , label='Customer 2')
         plt.scatter(X[y_kmeans == 2,0], X[y_kmeans == 2,1],  c='cyan', s=60 , label='Customer 3')
         #plt.scatter(kmeans.cluster_centers_[:, 0],kmeans.cluster_centers_[:, 1],s=100,c='magenta',label='Centroids')
         plt.title('Clusters of customers')
         plt.xlabel("Job title")
         plt.ylabel("Company")
         plt.legend()
         plt.show()
```



```
In [ ]:
```

Conclusion

We were able to analyse the data collected from Kaggle as well as scrapped from Glassdoor and Indeed and also used fermi estimation technique to fill in the right rows in the right columns. We were able to find insights from them as well as make visualizations to clearly understand the data.

The, customer segments are here based on different job titles like front-end, back-end and full-stack. To, each customer there is a different segment possible like only experience which can be based on the location and the company size (to say the least!) or to gain experience working in a company based on its size which would still have a benefit of having to filter out the salary and work in the company with best salary in a particular city.

Here, we saw that for every JS developer, the best city to work in right now is Bangalore which is followed closely by Mumbai.

However, if we see the best fit by a candidate’s job title and the company size, small sized companies are the best for Full stack engineers in terms of salary, medium sized companies are best for the Backend engineers and finally, large sized companies are best for Front end engineers.

All in all, it’s an excellent opportunity to become a JS developer in a growing country (in terms of IT sector) like India.

Thank you.

Github link

Here is the Github link for well documented codes and the dataset.

https://github.com/Vraj103/Job-segmentation