Data Analyst Intern at Data Glacier

Week 13: Final Report

Project: Customer Segmentation

Name: Ece Yavuzyılmaz

University: Duzce University

Email: eceyavuzyilmaz@gmail.com

Country: Turkey

Specialization: Data Analyst

Internship Batch: LISUM21

Date: 30 July 2023

Table of Contents

1. Problem Description

XYZ bank wants to roll out Christmas offers to their customers. But Bank does not want to roll out same offer to all customers instead they want to roll out personalized offer to particular set of customers. If they manually start understanding the category of customer then this will be not efficient and also they will not be able to uncover the hidden pattern in the data (pattern which group certain kind of customer in one category). Bank approached ABC analytics company to solve their problem. Bank also shared information with ABC analytics that they don't want more than 5 group as this will be inefficient for their campaign.

2. Business Understanding

ABC analytics proposed customer segmentation approach to Bank. ABC analytics assigned this talk to their analytics team and instructed their team to come up with the approach and feature which group similar behavior customer in one category and others in different category.

3. Project Lifecycle

Weeks	Date	Plan
Week 7	18 June 2023	Business Understanding
Week 8	26 June 2023	Data Understanding
Week 9	2 July 2023	EDA
Week 10	9 July 2023	Feature Engineering, Model Building
Week 11	16 July 2023	Model Evaluation
Week 12	23 July 2023	Presentation
Week 13	30 July 2023	Document the challenges

4. Data understanding

```python
data.head()
```

	id	fecha_dato	cust_code	employee_index	custom_country_residence	sexo	age	date_regist	new_cust_index	cust_senior
0	0	2015-01-28	1375586	N	ES	H	35	2015-01-28	0.0	
1	1	2015-01-28	1050611	N	ES	V	23	2015-01-28	0.0	
2	2	2015-01-28	1050612	N	ES	V	23	2015-01-28	0.0	
3	3	2015-01-28	1050613	N	ES	H	22	2015-01-28	0.0	
4	4	2015-01-28	1050614	N	ES	V	23	2015-01-28	0.0	

5 rows × 45 columns

```python
[4] data.shape
```

```
(1000000, 48)
```

```python
[7] data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1000000 entries, 0 to 999999
Data columns (total 45 columns):
 #   Column                    Non-Null Count    Dtype
---  ------                    --------------    -----
 0   id                        1000000 non-null  object
 1   fecha_dato                1000000 non-null  object
 2   cust_code                 1000000 non-null  object
 3   employee_index            989218 non-null   object
 4   custom_country_residence  989218 non-null   object
 5   sexo                      989214 non-null   object
 6   age                       989218 non-null   object
 7   date_regist               989218 non-null   object
 8   new_cust_index            989218 non-null   object
 9   cust_seniority            989218 non-null   object
 10  primary_cust              989218 non-null   object
 11  cust_type                 989218 non-null   object
```

```
[7]
 12  cust_relation_type        989218 non-null   object
 13  residence_index           989218 non-null   object
 14  foreigner_index           989218 non-null   object
 15  channel                   989139 non-null   object
 16  deceased_index            989218 non-null   object
 17  addres_type               989218 non-null   object
 18  cod_prov                  982266 non-null   object
 19  province_name             982266 non-null   object
 20  activity_index            989218 non-null   object
 21  gross_income              824817 non-null   object
 22  save_account              1000000 non-null  object
 23  current_account           1000000 non-null  object
 24  derivada_account          1000000 non-null  object
 25  payroll_account           1000000 non-null  object
 26  junior_account            1000000 non-null  object
 27  mas_particu_account       1000000 non-null  object
 28  particu_account           1000000 non-null  object
 29  particu_plus_account      1000000 non-null  object
 30  short_term_deposit        1000000 non-null  object
 31  medium_term_deposit       1000000 non-null  object
 32  long_term_deposit         1000000 non-null  object
 33  e_account                 1000000 non-null  object
 34  funds                     1000000 non-null  object
 35  mortgage                  1000000 non-null  object
 36  pensions                  1000000 non-null  object
 37  loans                     1000000 non-null  object
 38  taxes                     1000000 non-null  object
 39  credit_card               1000000 non-null  object
```

```
 40  securities              1000000 non-null  object
 41  home_account            1000000 non-null  object
 42  payroll                 994598 non-null   object
 43  a_pension               994598 non-null   object
 44  direct_debit            1000000 non-null  object
dtypes: object(45)
memory usage: 343.3+ MB
```

4.1. Changing Column Name

First, some column names have been changed for better understanding and interpretation the data.

```
#changing some column name
data.rename({'ncodpers':'cust_code', 'ind_empleado':'employee_index', 'fecha_alta': 'date_regist'}, axis=1, inplace=Tr
data.rename({'Unnamed: 0':'id','pais_residencia':'custom_country_residence', 'ind_nuevo':'new_cust_index', 'antiguedad
data.rename({'indrel':'primary_cust', 'ult_fec_cli_1t':'last_data_primary_cust', 'indrel_1mes': 'cust_type'}, axis=1,
data.rename({'tiprel_1mes':'cust_relation_type', 'indresi':'residence_index', 'indext': 'foreigner_index'}, axis=1, in
data.rename({'conyuemp':'spouse_index', 'canal_entrada':'channel', 'indfall':'deceased_index','tipodom':'addres_type'}
data.rename({'nomprov':'province_name', 'ind_actividad_cliente':'activity_index', 'renta': 'gross_income','ind_ahor_fi
data.rename({'ind_cco_fin_ult1':'current_account', 'ind_cder_fin_ult1':'derivada_account', 'ind_cno_fin_ult1':'payroll
data.rename({'ind_ctma_fin_ult1':'mas_particu_account', 'ind_ctop_fin_ult1':'particu_account','ind_ctpp_fin_ult1':'par
data.rename({'ind_deco_fin_ult1':'short_term_deposit', 'ind_deme_fin_ult1':'medium_term_deposit', 'ind_dela_fin_ult1':
data.rename({'ind_fond_fin_ult1':'funds', 'ind_hip_fin_ult1':'mortgage', 'ind_plan_fin_ult1':'pensions','ind_pres_fin_
data.rename({'ind_reca_fin_ult1':'taxes', 'ind_tjcr_fin_ult1':'credit_card', 'ind_valo_fin_ult1':'securities','ind_viv
data.rename({'ind_nomina_ult1':'payroll', 'ind_nom_pens_ult1':'a_pension', 'ind_recibo_ult1':'direct_debit'}, axis=1,
```

4.2. Data Types

All data types are seen as 'objects'. These data types have been changed as datetime64[ns](2), float64(3), int64(29), object(11).

```
data['cust_code']=data['cust_code'].astype("int64")
data['cust_seniority']=data['cust_seniority'].astype("int64")
data['activity_index']=data['activity_index'].astype("int64")
data['save_account']=data['save_account'].astype("int64")
data['current_account']=data['current_account'].astype("int64")
data['derivada_account']=data['derivada_account'].astype("int64")
data['payroll_account']=data['payroll_account'].astype("int64")
data['junior_account']=data['junior_account'].astype("int64")
data['mas_particu_account']=data['mas_particu_account'].astype("int64")
data['particu_account']=data['particu_account'].astype("int64")
data['particu_plus_account']=data['particu_plus_account'].astype("int64")
data['short_term_deposit']=data['short_term_deposit'].astype("int64")
data['medium_term_deposit']=data['medium_term_deposit'].astype("int64")
data['long_term_deposit']=data['long_term_deposit'].astype("int64")
data['e_account']=data['e_account'].astype("int64")
data['e_account']=data['e_account'].astype("int64")
data['funds']=data['funds'].astype("int64")
data['mortgage']=data['mortgage'].astype("int64")
data['pensions']=data['pensions'].astype("int64")
data['loans']=data['loans'].astype("int64")
data['taxes']=data['taxes'].astype("int64")
data['credit_card']=data['credit_card'].astype("int64")
data['securities']=data['securities'].astype("int64")
data['home_account']=data['home_account'].astype("int64")
data['direct_debit']=data['direct_debit'].astype("int64")
```

5. Data Cleaning

5.1. Deleting Unnecessary Column

```
#deleting unnecessary column
data.drop('ult_fec_cli_1t',axis=1, inplace=True)
data.drop('conyuemp',axis=1, inplace=True)
data.drop('ind_aval_fin_ult1',axis=1, inplace=True)
```

5.2. Missing Values

Two different methods were used to fill in the missing value. These methods are; fillna() and interpolate().

fillna()

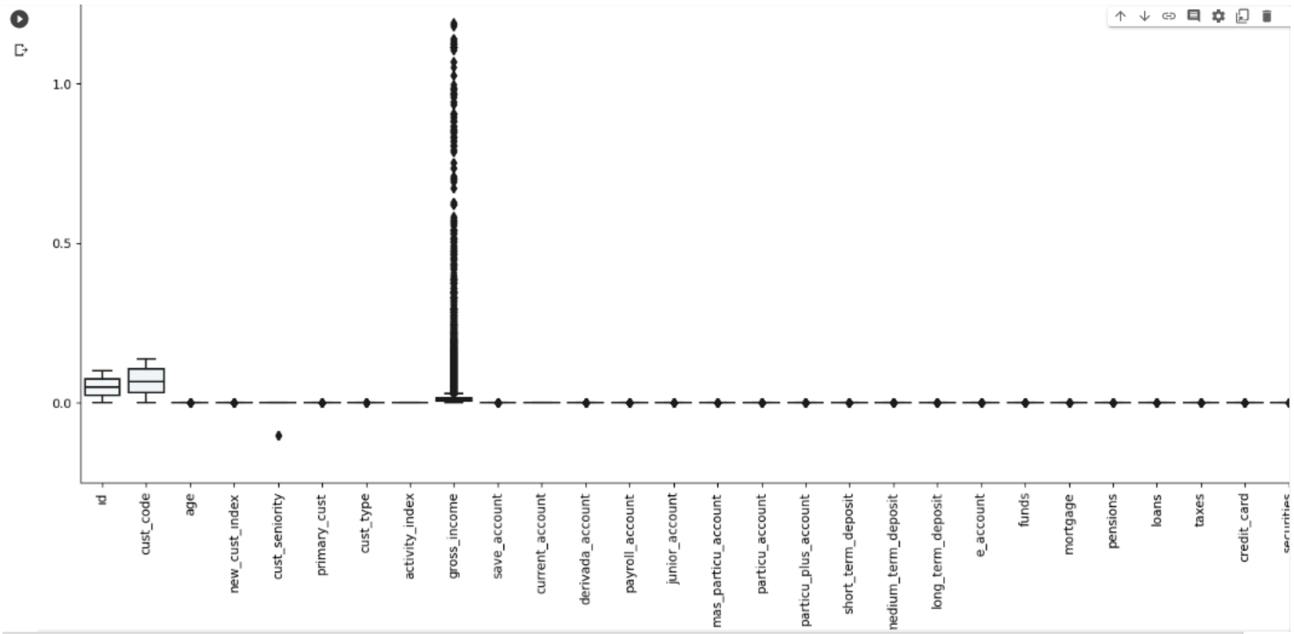
We can use ***fillna()*** function to fill NaN values. In this here, we can use different methods such 'backfill', 'bfill','pad','ffill' and mean/median/mode based approach to fill the missing values.

```
data['age']=data['age'].fillna(data['age'].mean()).astype("int64")
data['cust_type']=data['cust_type'].fillna(method ='ffill').astype("int64")
data['primary_cust']=data['primary_cust'].fillna(method ='ffill')
data['cust_relation_type']=data['cust_relation_type'].fillna(method ='ffill')
data['residence_index']=data['residence_index'].fillna(method='ffill')
data['foreigner_index']=data['foreigner_index'].fillna(method='ffill')
data['channel']=data['channel'].fillna(method='ffill')
data['deceased_index']=data['deceased_index'].fillna(method='ffill')
data['addres_type']=data['addres_type'].fillna(method='ffill')
data['province_name']=data['province_name'].fillna(method='ffill')
data['activity_index']=data['activity_index'].fillna(method='ffill')
data['payroll']=data['payroll'].fillna(method='ffill').astype("int64")
data['a_pension']=data['a_pension'].fillna(method='ffill').astype("int64")
data['employee_index']=data['employee_index'].fillna(method='ffill')
data['custom_country_residence']=data['custom_country_residence'].fillna(method='ffill')
data['sexo']=data['sexo'].fillna(method='ffill')
data['new_cust_index']=data['new_cust_index'].fillna(method='ffill')
data['cod_prov']=data['cod_prov'].fillna(method='ffill')
```

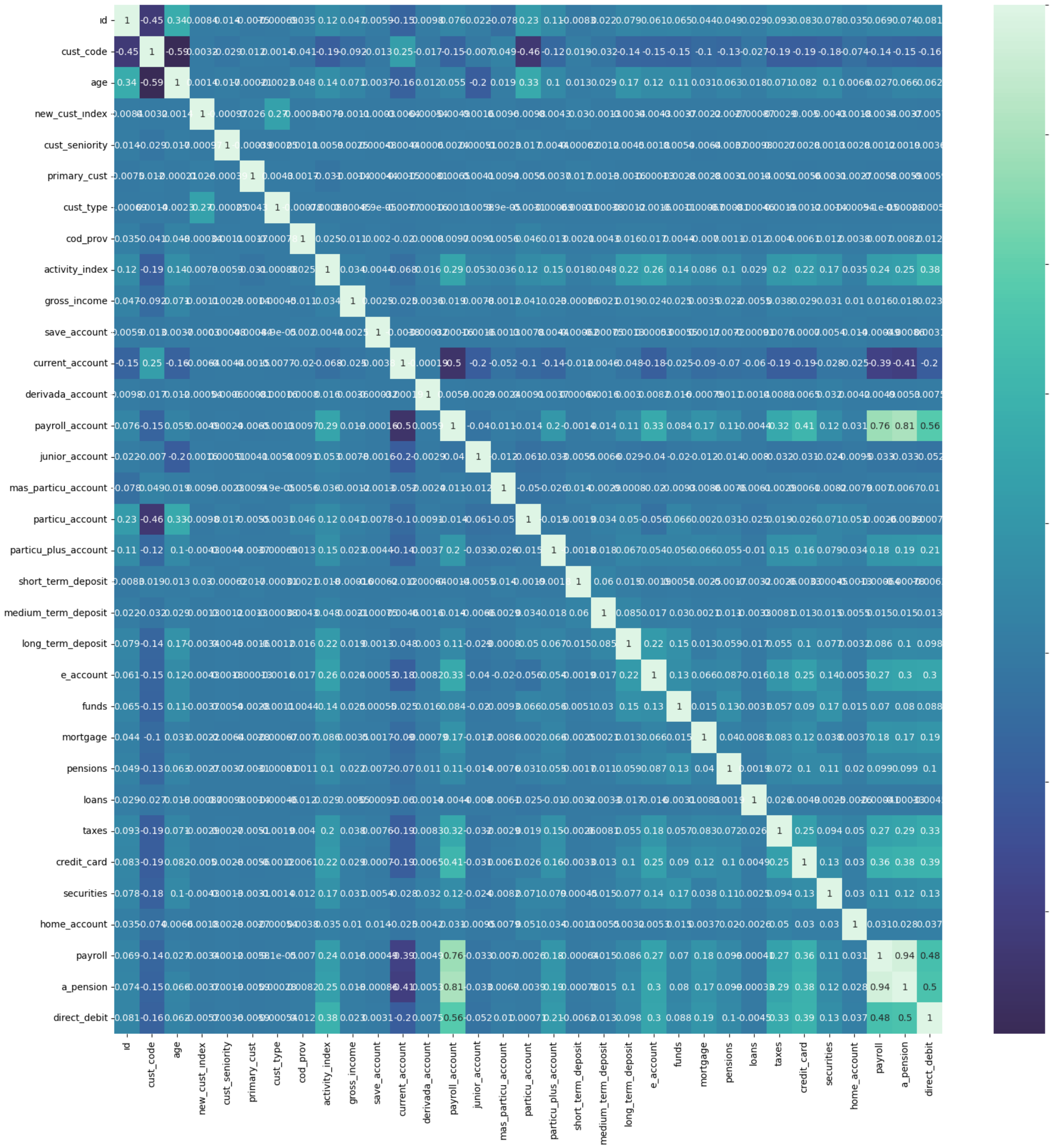
interpolate()

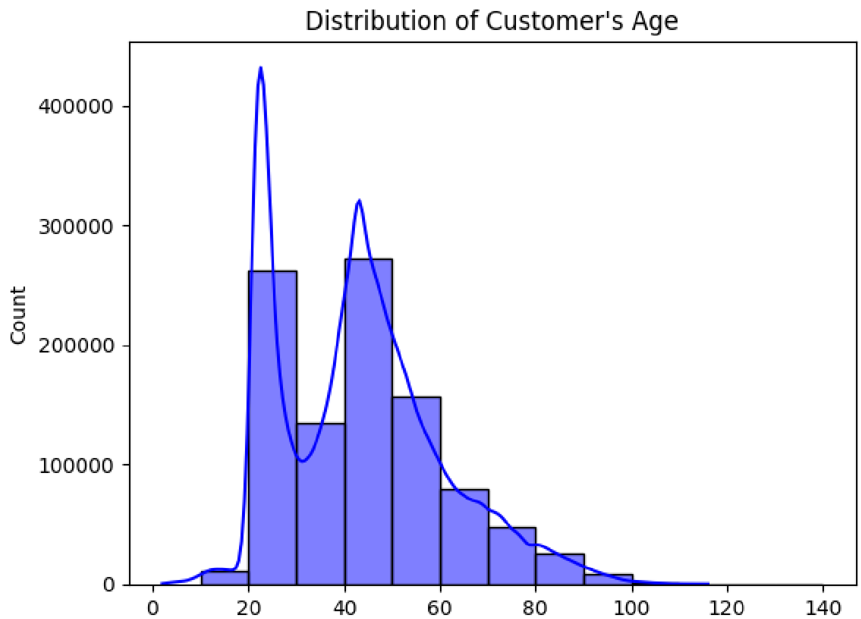
```
# Interpolate backwardly across the column:
data['gross_income'].interpolate(method ='linear', limit_direction ='backward', inplace=True)
#data['gross_income']=data['gross_income'].fillna(data['gross_income'].mean())
```

5.3. Outlier Detection

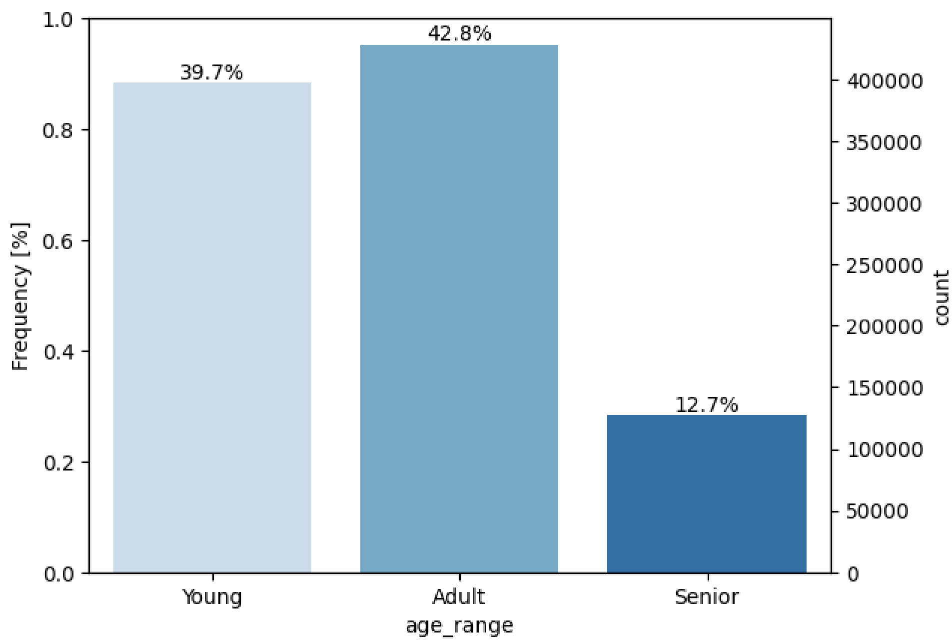


6. Exploratory Data Analysis (EDA)

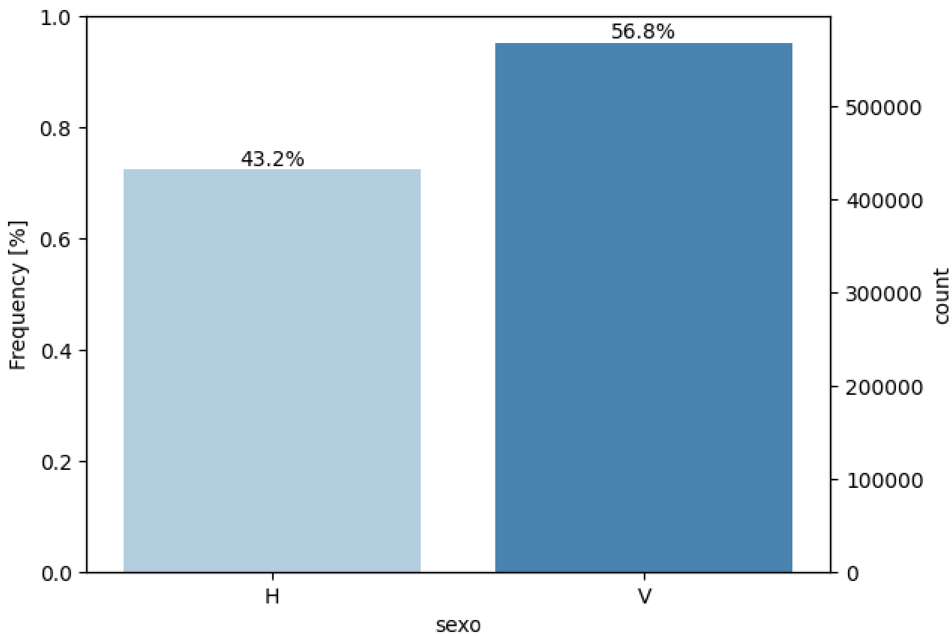




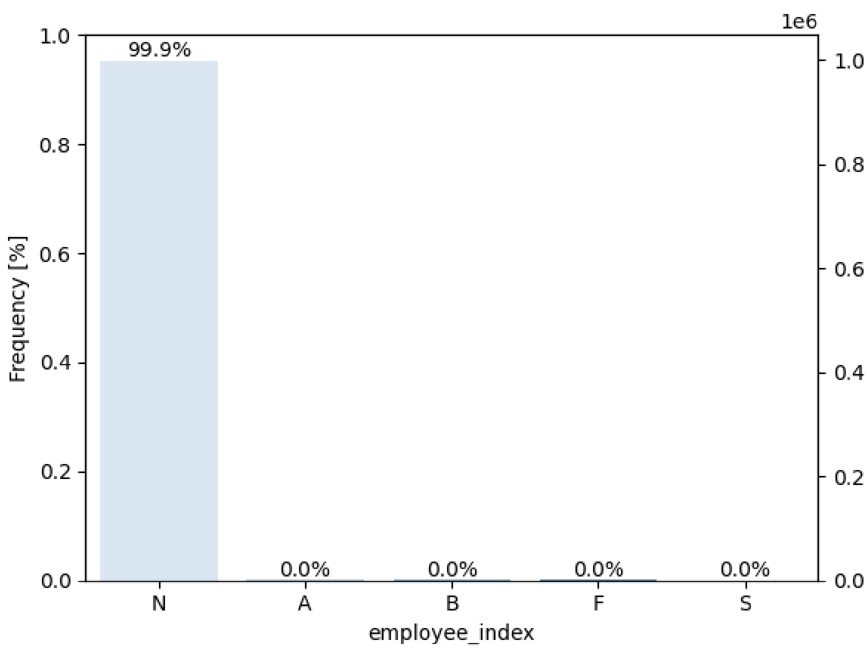
We can see from the above summary that most of the customers belong in the age range of 20-60. The number of adults among customers is 42.8% higher.



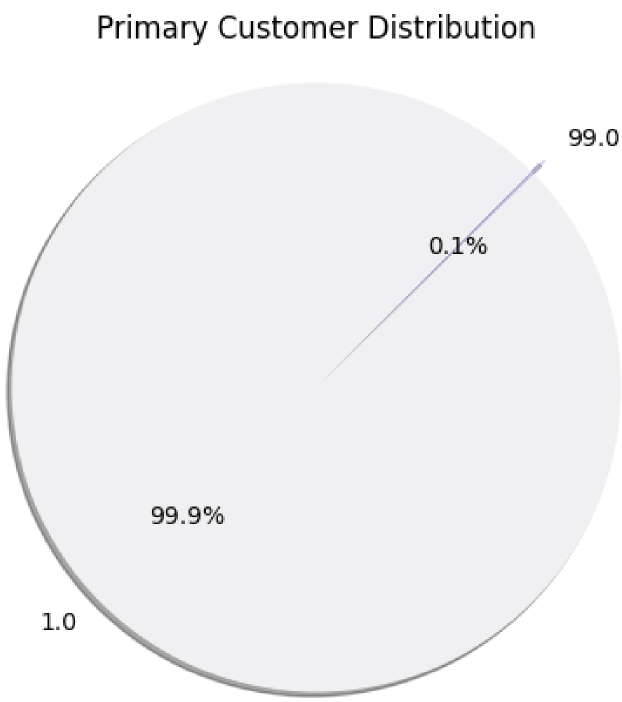
43.2% of customers are women and 56.8% are men.



99% of customers are not bank employees. (employee index: A active, B ex employed, F filial, N not employee, P passive)



It is seen that the primary customer ratio is very high compared to the second customer.



7. Final Recommendation

A model can be built using the K-means method for customer segmentation and the Elbow method for determining the number of clusters.

8. Model Selection and Model Building/Dashboard

We will use the K-means method for customer segmentation. It is also quite effective model for customer segmentation. The K-Means model is an unsupervised machine learning model that works by simply splitting N observations into K numbers of clusters. The observations are grouped into these clusters based on how close they are to the mean of that cluster, which is commonly referred to as centroids.

8.1. Standard Scalar

Before applying the elbow method, we will standardize the data. One of the best method for this is the StandardScalar() method. It prevents features with large variances from exerting an overly large influence during model training. Standard Scalar is already available in sklearn. Now we will standardize the feature set using Standard Scalar.

```python
# preprocessing
from sklearn.preprocessing import StandardScaler
```

```python
scaler = StandardScaler()
data_stand = pd.DataFrame(scaler.fit_transform(data), columns = data.columns)
```

```python
data_stand.head()
```

	id	cust_code	employee_index	custom_country_residence	sexo	age	new_cust_index	cust_seniority	primary_cust	cust_type	...	pensions	l
0	-1.732049	1.693807	0.028998	-0.03149	-1.147114	-0.484408	-0.022277	-1.391279	-0.033395	-0.006671	...	-0.121523	-0.06
1	-1.732046	0.890225	0.028998	-0.03149	0.871753	-1.187576	-0.022277	-0.948357	-0.033395	-0.006671	...	-0.121523	-0.06
2	-1.732042	0.890228	0.028998	-0.03149	0.871753	-1.187576	-0.022277	-0.948357	-0.033395	-0.006671	...	-0.121523	-0.06
3	-1.732039	0.890230	0.028998	-0.03149	-1.147114	-1.246173	-0.022277	-0.948357	-0.033395	-0.006671	...	-0.121523	-0.06
4	-1.732035	0.890232	0.028998	-0.03149	0.871753	-1.187576	-0.022277	-0.948357	-0.033395	-0.006671	...	-0.121523	-0.06

5 rows × 43 columns

8.2. Principal Component Analysis (PCA)

Now, We will apply PCA on the scaled dataset. Principal component analysis, or PCA, is a dimensionality reduction method that is often used to reduce the dimensionality of large data sets, by transforming a large set of variables into a smaller one that still contains most of the information in the large set.

The n_components argument defines the number of components that we want to reduce the features to.

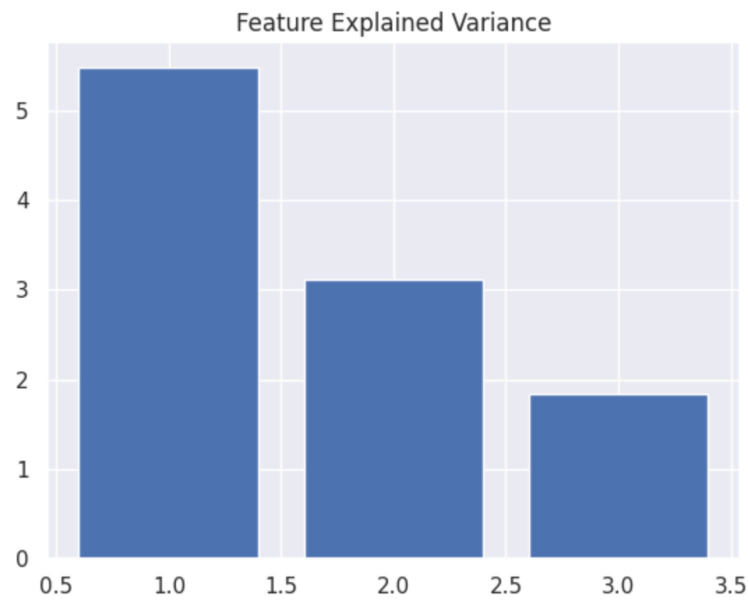
```python
from sklearn.decomposition import PCA
```

```python
pca = PCA(n_components = 3)
pca.fit(data_stand)
data_pca = pca.transform(data_stand)
data_pca = pd.DataFrame(data_pca,columns=['PC1','PC2','PC3'])
data_pca.head()
```

	PC1	PC2	PC3
0	-2.274396	-1.872841	-0.629351
1	-2.985063	-1.281629	1.434704
2	-2.905985	-1.281055	1.289202
3	-2.412987	-1.583722	-2.836378
4	-1.829137	-1.388748	-2.785494

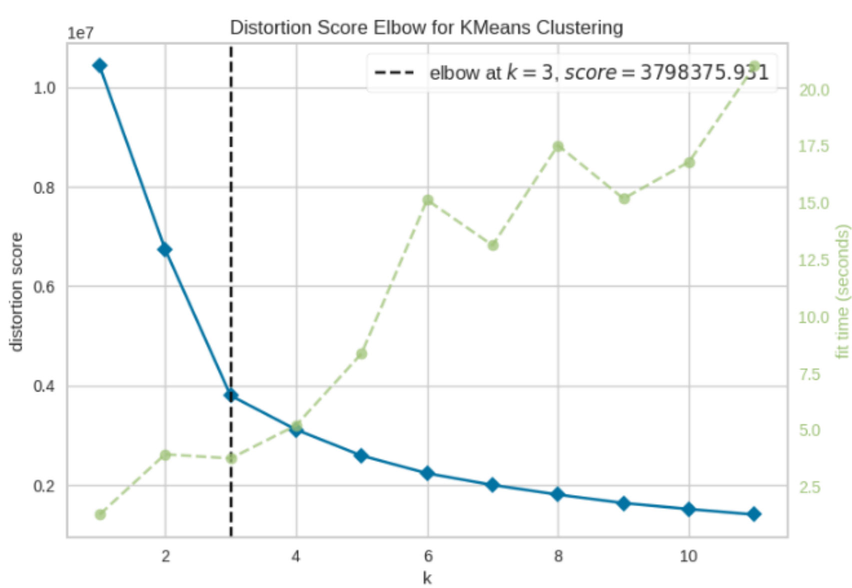
```python
pca.explained_variance_
```

```
array([5.48743958, 3.10746732, 1.83614692])
```



8.3. Elbow Method

The KElbowVisualizer implements the “elbow” method to help data scientists select the optimal number of clusters by fitting the model with a range of values for. If the line chart resembles an arm, then the “elbow” (the point of inflection on the curve) is a good indication that the underlying model fits best at that point. In the visualizer “elbow” will be annotated with a dashed line.



The graph above shows that the optimal number of clusters (k) for our model is three.

8.4. K_Means Clustering

KMeans Clustering analysis results are as follows.

```
from sklearn.cluster import KMeans
kmeans = KMeans(n_clusters=3, init='k-means++', random_state=0).fit(data_pca)
```

```
kmeans.labels_
```

```
array([1, 1, 1, ..., 1, 1, 1], dtype=int32)
```

```
kmeans.inertia_
```

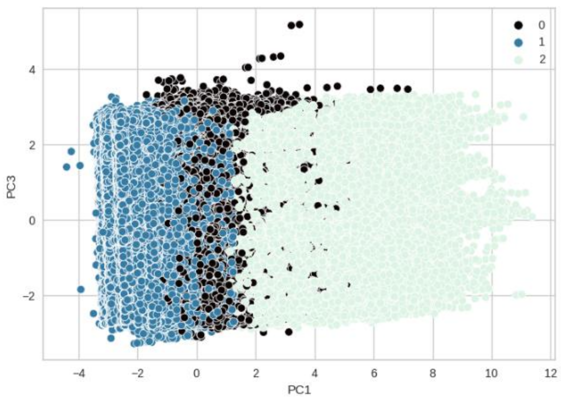
```
3798378.5084681325
```

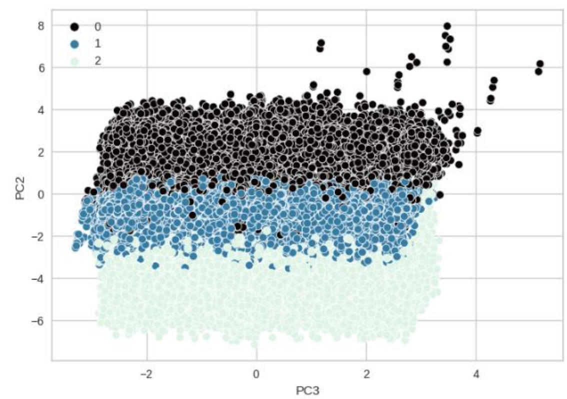
```
kmeans.cluster_centers_
```

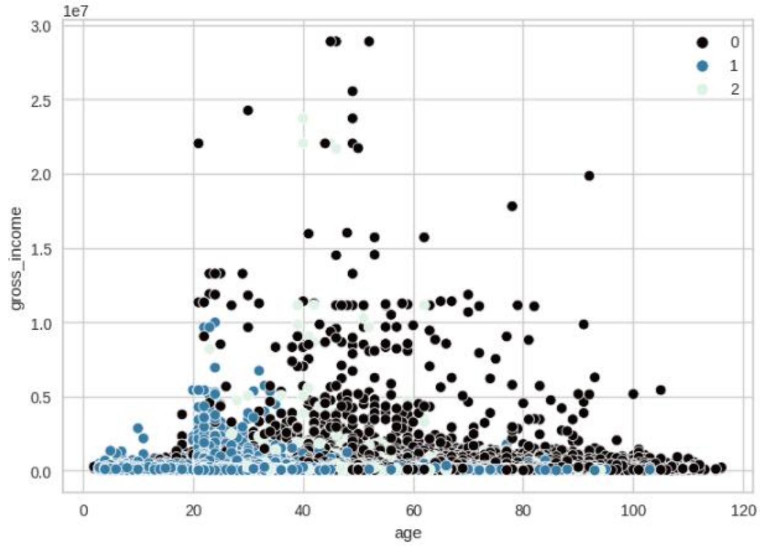
```
array([[ 0.61158024,  1.38439195,  0.02500715],
       [-2.07863506, -1.1083183 , -0.04310727],
       [ 5.03639032, -2.94844787,  0.03883934]])
```

```
from collections import Counter
Counter(kmeans.labels_)
```

```
Counter({1: 387087, 2: 96975, 0: 515938})
```







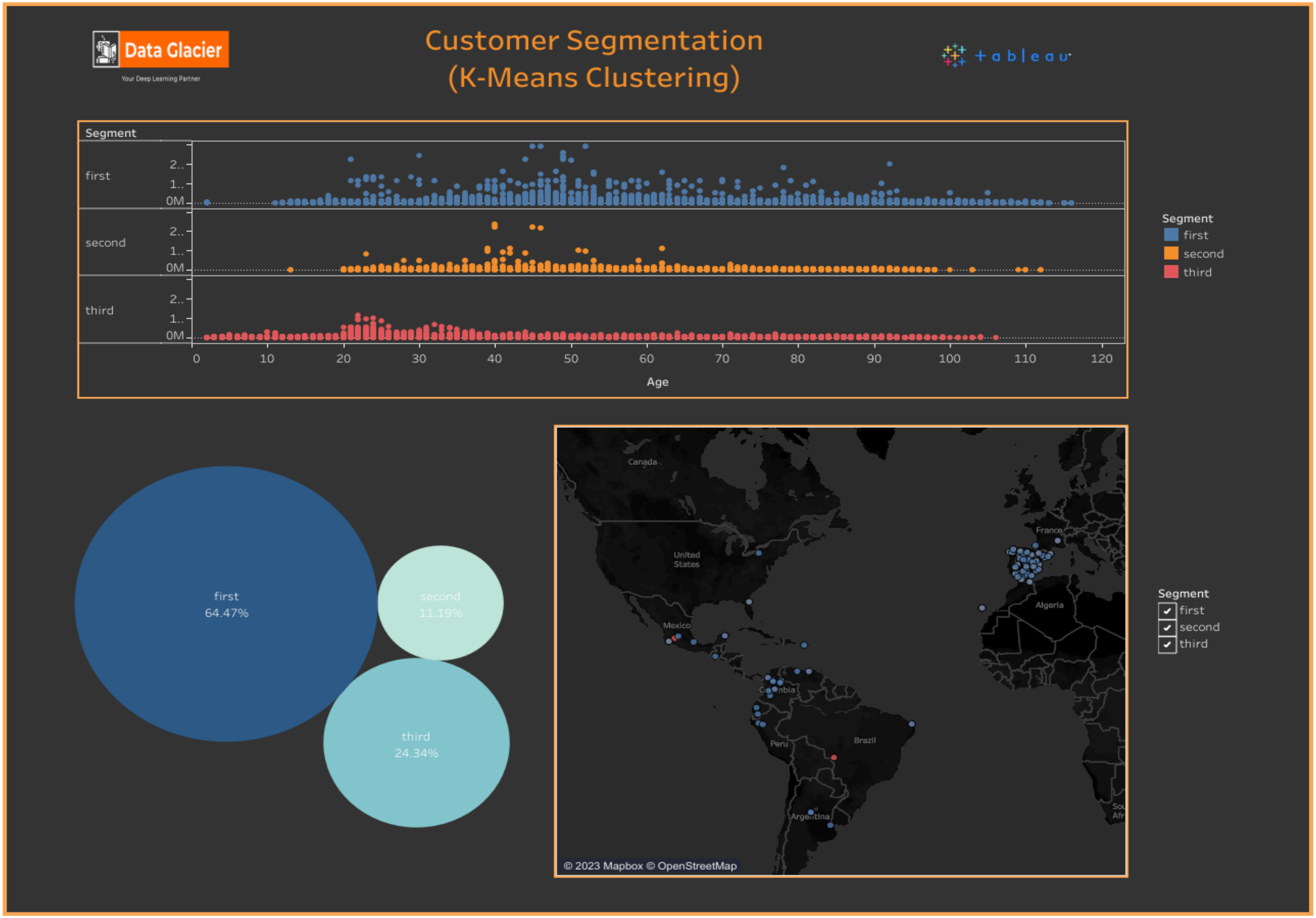
According to KMeans Clustering analysis, the 3 customer groups are as follows:

- *0 group: Customers who age rate 40 to 60 and income rate 130000 to1500000*
- *1 group: Customers who age rate 30 to 60 and income rate 120000 to 1500000*
- *2 group: Customers who age rate 20 to 40 and income rate 100000 to1000000*

The number of customer by clusters are as follows:

0 group is 52% of all customers, any improvements achieved in this customer group will dramatically benefit the campaign.

8.5. Dasboard



Data Glacier
Your Deep Learning Partner
Customer Segmentation
(K-Means Clustering)
+ a b | e a u
Segment
first
second
third
2..
1..
0M
0
10
20
30
40
50
60
70
80
90
100
110
120
Age
first
64.47%
second
11.19%
third
24.34%
Canada
United States
Mexico
Colombia
Peru
Brazil
Argentina
France
Algeria
© 2023 Mapbox © OpenStreetMap

9. Conclusion

For an unsupervised machine learning task, the dataset was well-suited. To tackle this project, we used an unsupervised machine learning approach with the K Means clustering method using the PCA method as the dataset contained many features. After applying the Elbow method, it was determined that three clusters would be optimal. Using K-means clustering, patterns in the data were identified and used to create groups, paving the way for strategies tailored to these groups. In the future, customer groups can be created using specific features from the dataset, allowing for personalized offers to be extended.

10. Reference

1. https://towardsdatascience.com/customer-segmentation-with-machine-learning-a0ac8c3d4d84
2. https://www.freecodecamp.org/news/customer-segmentation-python-machine-learning/
3. https://github.com/Pegah-Ardehkhani/Customer-Segmentation/blob/main/Customer%20Segmentation%20(Clustering).ipynb