BANK CUSTOMER CHURN PREDICTION

GROUP 15

Anjali Yamdagni
Aparna Raghavendra Rao
Arvind Kanhirampara Ravi
Charishma Jaladi
Nishanthi Ravichandran
Sai Arjun Madikonda

ABC Multistate Bank - About the Dataset

Customer Id	Customer's unique identifier
Credit Score	Credit score of the customer
Country	Country of residence
Gender	Gender defined as Male, Female
Age	Age of the member
Tenure	No. of years the customer had a bank acc. with ABC

Balance	Account Balance
Products Number	Number of Products from Bank
Credit Card	Binary variable - if customer has a credit card
active_member	Binary variable - if the customer is an active member
estimated_salary	Estimated salary of the customer
churn	Used as the target 1 if the client has left the bank during some period 0 if he/she has not left the bank

What is customer churn?

Introduction

- Customer Churn: the customers that stopped using the bank's service or products
- Customer Retention is an imperative metric for bank's repeated and continued business with bank

Objectives

- Predict bank's customer churn
- Analyze the bank's demographic information

Data Visualization - Churn with Demographics

The dataset is sliced into the following subset, to further analyze Churn with respect to Demographics

```
Bank_Demographics_Data = bank_data[['country','gender','age','age group','churn']]
print("The below subset of the original dataset is to be used from here on to analyze Churn with respect to Demographics")

Bank_Demographics_Data
```

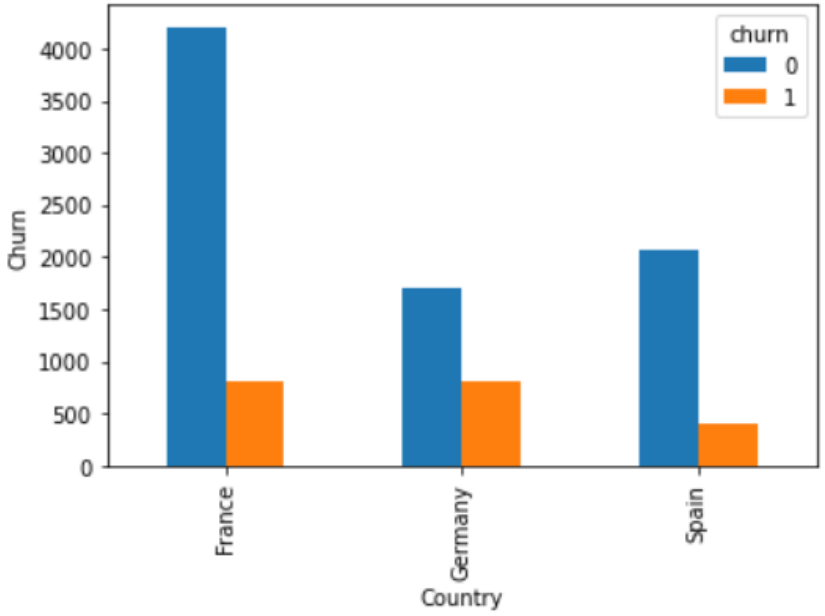
```
The below subset of the original dataset is to be used from here on to analyze Churn with respect to Demographics
```

customer_id	country	gender	age	age group	churn
15634602	France	Female	42	Middle-Age Adult	1
15647311	Spain	Female	41	Middle-Age Adult	0
15619304	France	Female	42	Middle-Age Adult	1
15701354	France	Female	39	Adult	0
15737888	Spain	Female	43	Middle-Age Adult	0
...	...	...	...	...	...
15606229	France	Male	39	Adult	0
15569892	France	Male	35	Adult	0
15584532	France	Female	36	Adult	1
15682355	Germany	Male	42	Middle-Age Adult	1
15628319	France	Female	28	Adult	0

Churn Vs Country

```
#Churn Vs Country
Churn_Vs_Country = Bank_Demographics_Data.groupby(['country', 'churn'])['churn'].count()
Churn_Vs_Country
```

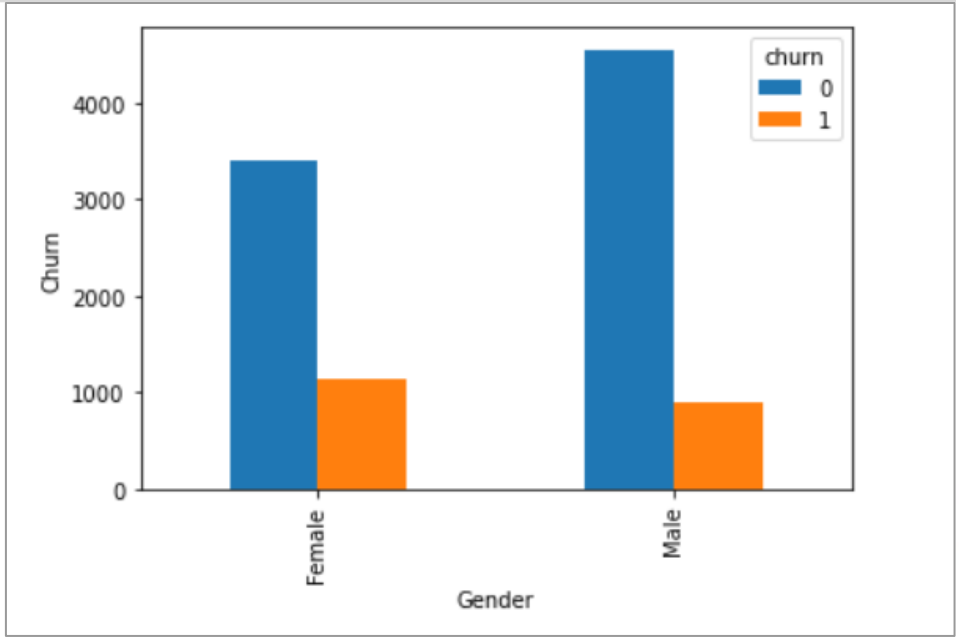
```
country  churn
France   0        4204
         1         810
Germany  0        1695
         1         814
Spain    0        2064
         1         413
Name: churn, dtype: int64
```



Churn Vs Gender

```
#Churn Vs Gender
Churn_Vs_Gender = Bank_Demographics_Data.groupby(['gender', 'churn'])['churn'].count()
Churn_Vs_Gender
```

```
gender  churn
Female  0        3404
        1        1139
Male    0        4559
        1         898
Name: churn, dtype: int64
```



Churn vs Age

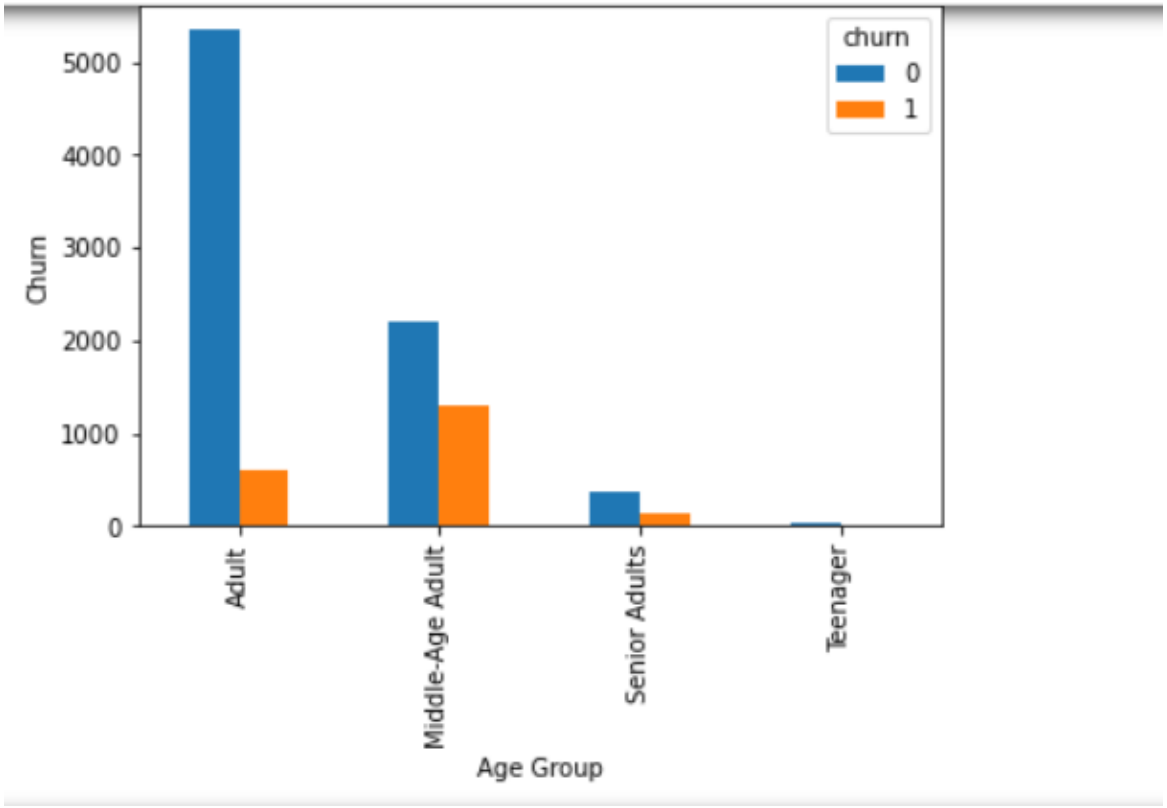
```
#Churn Vs Age
Churn_Vs_Age = Bank_Demographics_Data.groupby(['age', 'churn'])['churn'].count()
Churn_Vs_Age
```

```
#Churn Vs Age Groups
Churn_Vs_Age_Group = Bank_Demographics_Data.groupby(['age group', 'churn'])['churn'].count()
Churn_Vs_Age_Group
```

```
age  churn
18   0        20
     1         2
19   0        26
     1         1
20   0        38
              ..
84   0         1
     1         1
85   0         1
88   0         1
92   0         2
Name: churn, Length: 128, dtype: int64
```

```
age group           churn
Adult               0        5344
                    1         594
Middle-Age Adult    0        2194
                    1        1293
Senior Adults       0         379
                    1         147
Teenager            0          46
                    1           3
Name: churn, dtype: int64
```

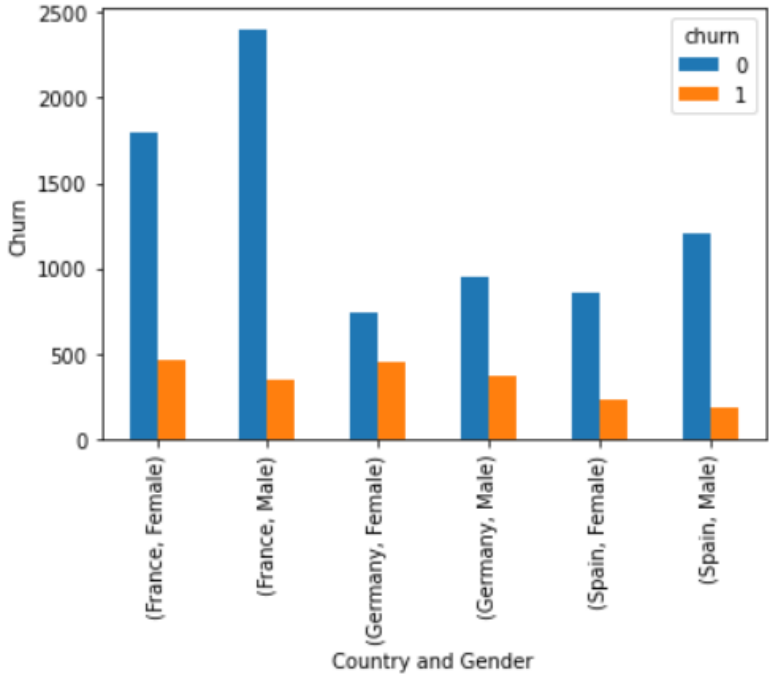
Churn vs Age Group Plot



Churn vs Country and Gender

```
#Pivot table of Churn Vs Country and Gender
print("Below is a pivot table representing the distribution of Churn with respect to Country")
Bank_Demographics_Data.pivot_table(index = 'country', columns = 'gender', values = 'churn', aggfunc='sum')
```

gender	Female	Male
country		
France	460	350
Germany	448	366
Spain	231	182



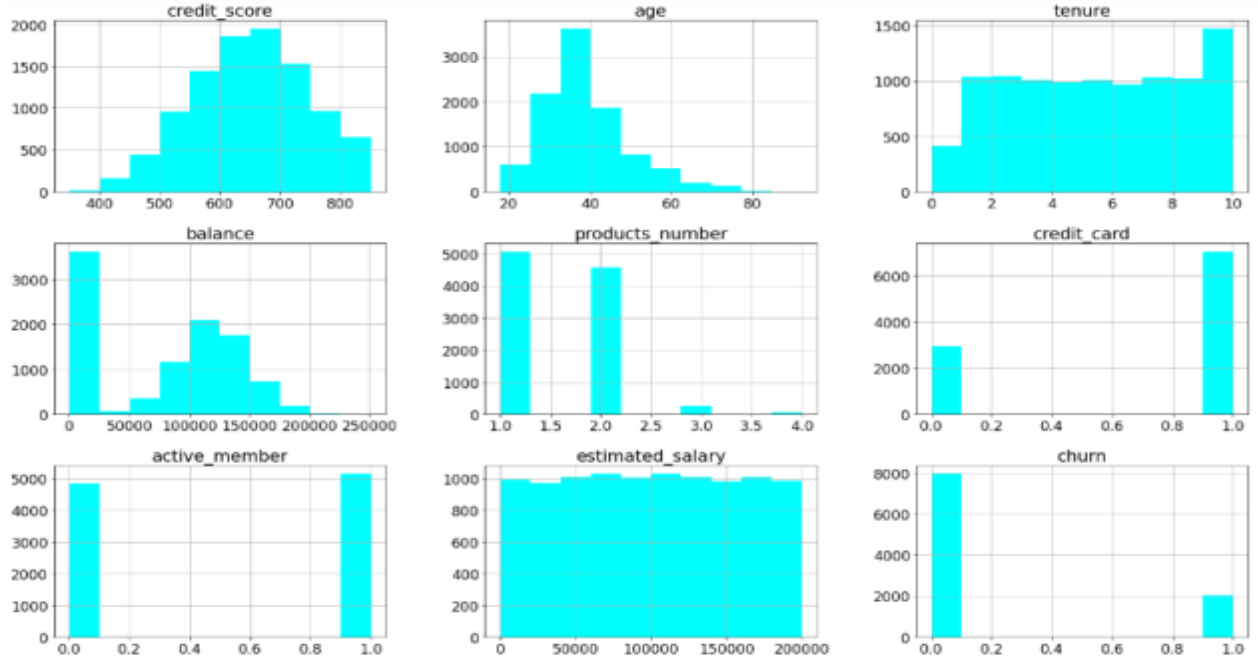
Correlation Matrix

```
corr=bank_data.corr()
corr.style.background_gradient(cmap='coolwarm')
```

	credit_score	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn
credit_score	1.000000	-0.003965	0.000842	0.006268	0.012238	-0.005458	0.025651	-0.001384	-0.027094
age	-0.003965	1.000000	-0.009997	0.028308	-0.030680	-0.011721	0.085472	-0.007201	0.285323
tenure	0.000842	-0.009997	1.000000	-0.012254	0.013444	0.022583	-0.028362	0.007784	-0.014001
balance	0.006268	0.028308	-0.012254	1.000000	-0.304180	-0.014858	-0.010084	0.012797	0.118533
products_number	0.012238	-0.030680	0.013444	-0.304180	1.000000	0.003183	0.009612	0.014204	-0.047820
credit_card	-0.005458	-0.011721	0.022583	-0.014858	0.003183	1.000000	-0.011866	-0.009933	-0.007138
active_member	0.025651	0.085472	-0.028362	-0.010084	0.009612	-0.011866	1.000000	-0.011421	-0.156128
estimated_salary	-0.001384	-0.007201	0.007784	0.012797	0.014204	-0.009933	-0.011421	1.000000	0.012097
churn	-0.027094	0.285323	-0.014001	0.118533	-0.047820	-0.007138	-0.156128	0.012097	1.000000

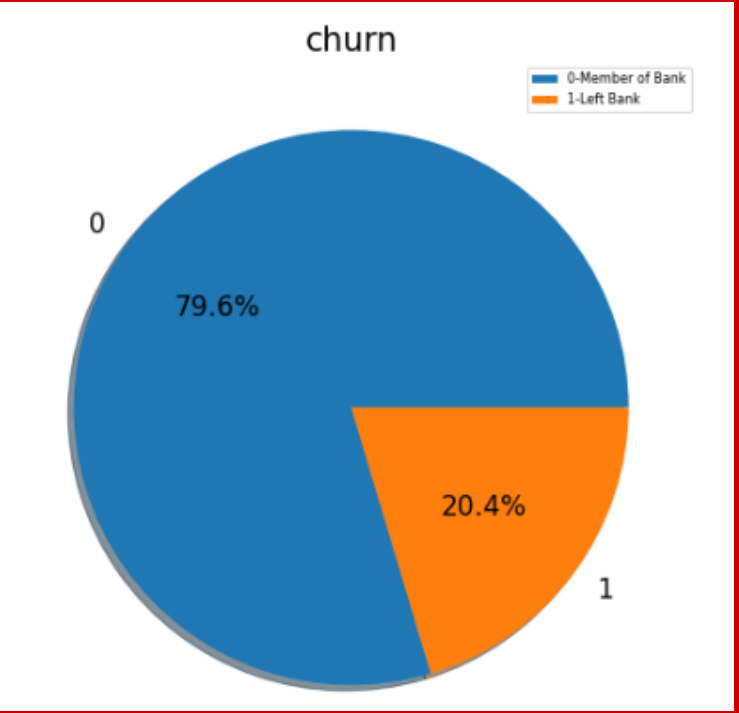
Distribution of all Columns

```
import matplotlib.pyplot as plt
color = '#00FFFF'
bank_data.hist(bins=10,figsize=(25,15),color=color)
plt.rcParams['font.size'] = 17
plt.show()
```



Majority of customers are still with the bank

```
import matplotlib.pyplot as plt
plt.figure(figsize=(12,8))
bank_data['churn'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True);
plt.title('churn');
plt.ylabel('');
plt.legend(['0-Member of Bank','1-Left Bank'],prop={'size': 8})
```



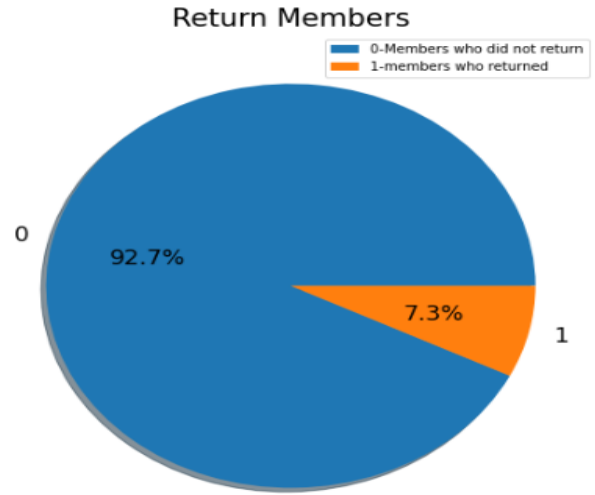
Adding a new metric 'return_member'

```
import numpy as np
conditions = [
    (bank_data['active_member'] == 1) & (bank_data['churn'] == 1)
        ]
values = ['1']
bank_data['return_member']  = np.select(conditions, values)
bank_data.head()
```

customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn	return_member
15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1	1
15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	0
15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	0
15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0	0
15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	0

The bank is only able to re-attract a small percentage of old customers

```
import matplotlib.pyplot as plt
plt.figure(figsize=(10,8))
bank_data['return_member'].value_counts().plot.pie(autopct='%1.1f%%',shadow=True);
plt.title('Return Members');
plt.ylabel('');
plt.legend(['0-Members who did not return','1-members who returned'],prop={'size': 8})
```



Members who returned are defined as customers who left the bank(churn = 1) & returned(active_member = 1)

Creating a new column 'iso_code' for plots

```
conditions = [
     (bank_data['country'] == 'France'),(bank_data['country'] == 'Spain'),(bank_data['country'] == 'Germany')
     ]
values = ['FRA','ESP','DEU']
bank_data['iso_code']  = np.select(conditions, values)
bank_data.head()
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn	iso_code
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1	FRA
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	ESP
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	FRA
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0	FRA
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	ESP

Calculating the number of customers for each country only for churned customers

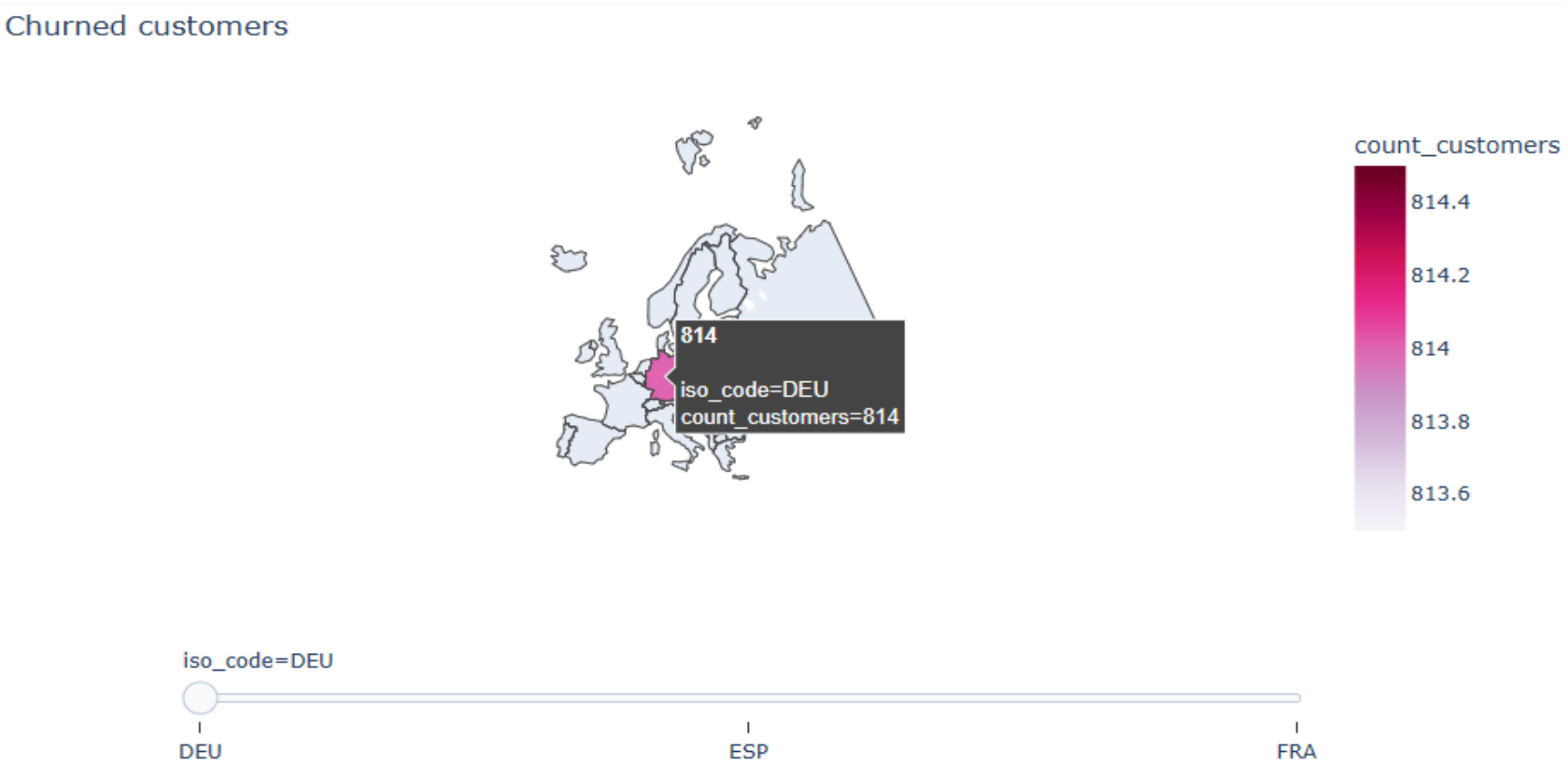
```
map_data1 = bank_data[bank_data['churn']== 1]
map_data = pd.DataFrame()
map_data['count_customers'] = map_data1.groupby('iso_code')['customer_id'].count()
map_data.reset_index(inplace=True,level = ['iso_code'])
print(map_data)
```

	iso_code	count_customers
0	DEU	814
1	ESP	413
2	FRA	810

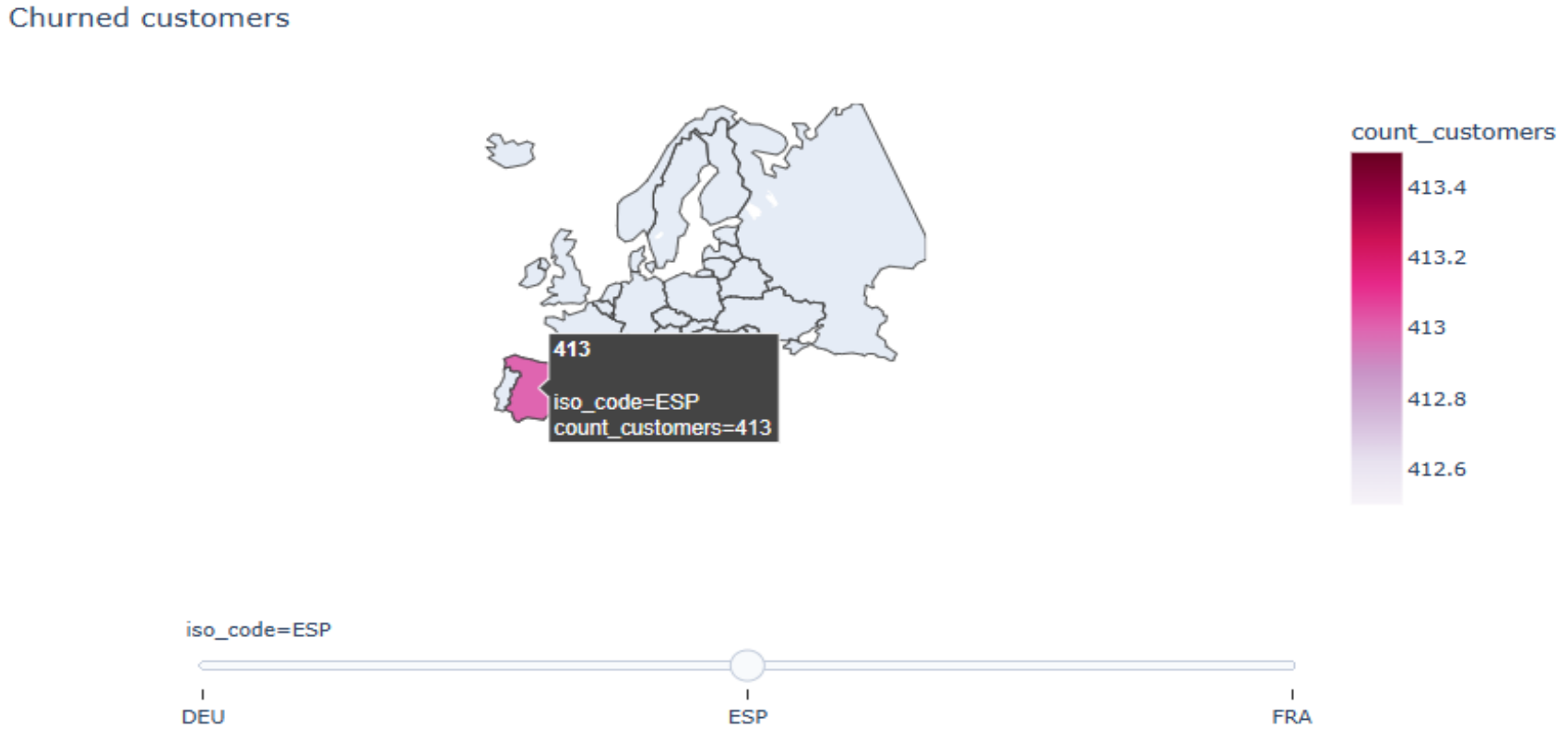
Plotting the number of customers for each country only for churned customers

```
fig = px.choropleth(map_data, locations="iso_code",
                    color="count_customers",
                    hover_name="count_customers",
                    animation_frame="iso_code",
                    scope="europe",
                    title = "Churned customers",                    color_continuous_scale=px.colors.sequential.PuRd)
fig["layout"].pop("updatemenus")
fig.show()
```

Number of churned customers for Germany

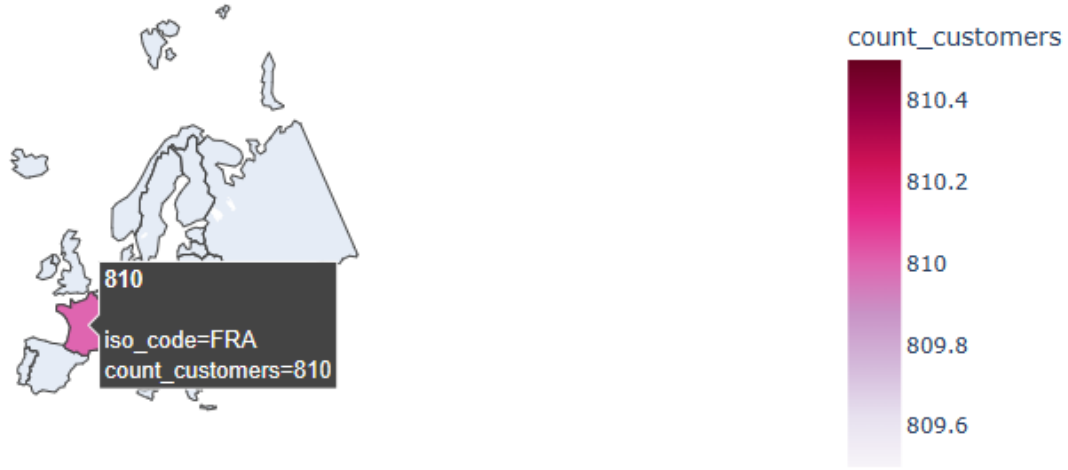


Number of churned customers for Spain



Number of churned customers for France

Churned customers



iso_code=FRA

DEU ESP FRA

Data Preprocessing

```
#Convert 'gender' and 'country' to categorical
bank_data['gender_new']=bank_data.gender.map({'Female':0,'Male':1})
bank_data['country_new'] = bank_data.country.map({'France':1,'Spain':2,'Germany':3})

bank_data.head(5)
```

	customer_id	credit_score	country	gender	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn	ge
0	15634602	619	France	Female	42	2	0.00	1	1	1	101348.88	1	
1	15647311	608	Spain	Female	41	1	83807.86	1	0	1	112542.58	0	
2	15619304	502	France	Female	42	8	159660.80	3	1	0	113931.57	1	
3	15701354	699	France	Female	39	1	0.00	2	0	0	93826.63	0	
4	15737888	850	Spain	Female	43	2	125510.82	1	1	1	79084.10	0	

```
bank_data_new = bank_data.drop(['country','gender'],axis=1)
bank_data_new.head()
```

	customer_id	credit_score	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn	gender_new	country
0	15634602	619	42	2	0.00	1	1	1	101348.88	1	0	
1	15647311	608	41	1	83807.86	1	0	1	112542.58	0	0	
2	15619304	502	42	8	159660.80	3	1	0	113931.57	1	0	
3	15701354	699	39	1	0.00	2	0	0	93826.63	0	0	
4	15737888	850	43	2	125510.82	1	1	1	79084.10	0	0	

STEP 1

Update Categorical column values

Data Preprocessing

```
#Standardize the data
scaling_cols = ['tenure','balance','estimated_salary']

# Import the StandardScaler
from sklearn.preprocessing import StandardScaler

# Scale the features and set the values to a new variable
scaler = StandardScaler()
bank_data_new[scaling_cols] = scaler.fit_transform(bank_data_new[scaling_cols])
bank_data_new.head(2)
```

customer_id	credit_score	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	churn	gender_new	country_
15634602	619	42	-1.041760	-1.225848	1	1	1	0.021886	1	0	
15647311	608	41	-1.387538	0.117350	1	0	1	0.216534	0	0	

STEP 2

Standardizing the data

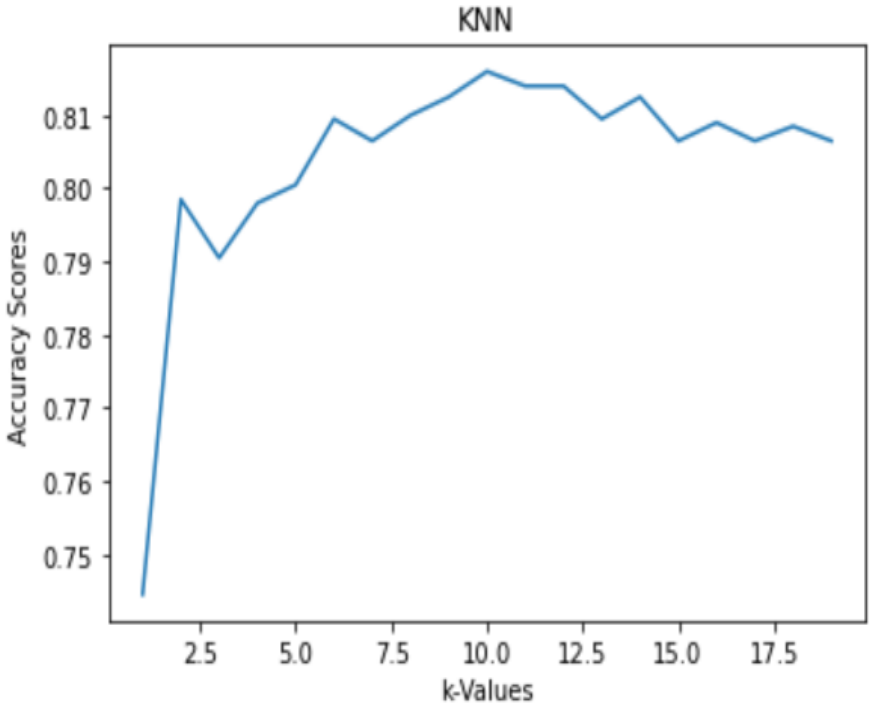
Logistic Regression

```
y = bank_data_new['churn']
X = bank_data_new.drop(['churn'],axis=1)

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

logreg_model = LogisticRegression(solver='liblinear')
logreg_model.fit(X_train,y_train)
y_pred = logreg_model.predict(X_test)
print(confusion_matrix(y_test, y_pred))
print(classification_report(y_test, y_pred))
#Accuracy score
from sklearn.metrics import accuracy_score
print("Accuracy:", accuracy_score(y_test, y_pred))
```

```
[[1543   64]
 [ 310   83]]
              precision    recall  f1-score   support

           0       0.83      0.96      0.89      1607
           1       0.56      0.21      0.31       393

    accuracy                           0.81      2000
   macro avg       0.70      0.59      0.60      2000
weighted avg       0.78      0.81      0.78      2000

Accuracy: 0.813
```

- Train - 80%
- Test - 20%

K-Nearest Neighbors



n_neighbors = 5

```
y = bank_data_new['churn']
X = bank_data_new.drop(['churn'],axis=1)
knn_model = KNeighborsClassifier(n_neighbors=5)
X_train_knn,X_test_knn,y_train_knn,y_test_knn = train_test_split(X,y,test_size=0.2,random_state=42)
#Fit
knn_model.fit(X_train_knn,y_train_knn)
#predict labels
predicted_y_5 = knn_model.predict(X_test_knn)
#Print confusion matrix & classification report
print(confusion_matrix(y_test_knn,predicted_y_5))
print(classification_report(y_test_knn,predicted_y_5))
print("Accuracy score when k=5", accuracy_score(y_test_knn,predicted_y_5))
```

```
[[1492  115]
 [ 284  109]]
              precision    recall  f1-score   support

           0       0.84      0.93      0.88      1607
           1       0.49      0.28      0.35       393

    accuracy                           0.80      2000
   macro avg       0.66      0.60      0.62      2000
weighted avg       0.77      0.80      0.78      2000

Accuracy score when k=5 0.8005
```

Probability of Churn

```
bank_data_new['Churn_Probability_logreg'] = logreg_model.predict_proba(bank_data_new[X_test.columns])[:,1]
bank_data_new['Churn_Probability_knn'] = knn_model.predict_proba(bank_data_new[X_test_knn.columns])[:,1]
prob = pd.DataFrame(bank_data_new, columns = ['churn','Churn_Probability_logreg','Churn_Probability_knn'])
prob = prob.sample(n=10)
print(prob)
```

customer_id	churn	Churn_Probability_logreg	Churn_Probability_knn
15717995	0	0.020444	0.0
15634218	0	0.174211	0.0
15591047	1	0.552893	0.8
15582276	0	0.061163	0.0
15790678	0	0.225681	0.2
15809777	0	0.512526	0.8
15808228	0	0.181978	0.4
15664881	0	0.135619	0.0
15622585	0	0.039266	0.0
15753332	1	0.544070	0.6

- Can enter details of a customer and compare the model's prediction of the probability of the customer leaving the bank under two different models
- Logistic regression = 81.30% accurate
  KNN model = 80.05% accurate
- Provides information but decision ultimately is based on management's experience & other qualitative factors

Churn Prediction using Mock Data

```
##PREDICT
predicted_churn = logreg_model.predict(mock_data)
#print(predicted_churn)
mock_data['predicted_churn']=predicted_churn
mock_data.head(10)
```

customer_id	credit_score	age	tenure	balance	products_number	credit_card	active_member	estimated_salary	gender_new	country_new	predicted_churn
1	420	42	-1.573994	-0.831751	3	0	0	-0.851690	1	2	0
2	634	76	0.861990	1.715164	4	0	0	-0.660701	1	3	1
3	712	67	1.209988	-0.407173	3	1	0	-0.275257	1	3	1
4	842	78	0.861990	-1.145727	2	1	1	0.938087	0	2	1
5	796	57	-0.877998	1.286579	1	0	1	-0.482644	0	3	1
6	491	42	-0.877998	-0.008866	4	1	0	0.630363	0	3	0
7	411	58	-0.182003	0.501233	3	0	1	0.434174	1	2	0
8	844	89	1.557986	-0.711814	2	0	0	1.148677	0	2	1
9	383	28	1.557986	1.458033	4	1	1	1.192671	1	2	0
10	350	91	-0.877998	-0.385284	1	1	0	-1.159887	0	3	1

Decision Tree Classifier

```
from sklearn.tree import DecisionTreeClassifier

# Create Decision Tree classifer object
dt = DecisionTreeClassifier(criterion='entropy', max_depth=4)

# Train Decision Tree Classifer
dt = dt.fit(X, y)

#Predict the response for test dataset
y_pred = dt.predict(X)

# print out the accuracy of the classifier using `accuracy_score` function
print("Accuracy:", accuracy_score(y, y_pred))
```

```
Accuracy: 0.8468
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, random_state = 0)
```

```
tree_clf = DecisionTreeClassifier(random_state = 0)
tree_clf.fit(X_train, y_train)
```

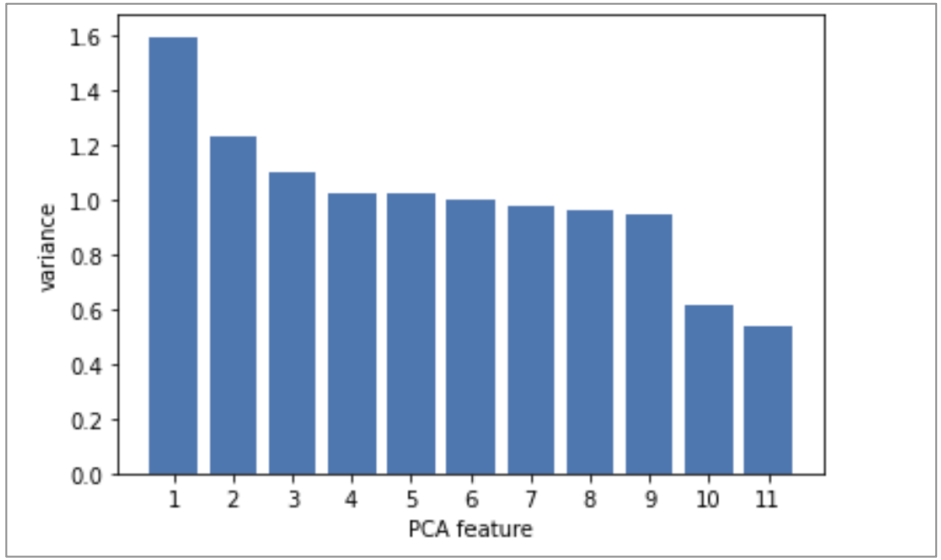
```
DecisionTreeClassifier(random_state=0)
```

```
class_rep = classification_report(y_test,y_pred[:2500])

print(class_rep)
```

	precision	recall	f1-score	support
0	0.80	0.94	0.86	1991
1	0.22	0.07	0.11	509
accuracy			0.76	2500
macro avg	0.51	0.50	0.48	2500
weighted avg	0.68	0.76	0.71	2500

PCA

```
from sklearn.pipeline import make_pipeline
# Import PCA
from sklearn.decomposition import PCA

# Create a PCA instance: pca
pca = PCA()

# Create pipeline: pipeline
pipeline = make_pipeline(scaler, pca)

samples = bank_data_new.values

# Fit the pipeline to 'samples'
pipeline.fit(samples)

# Plot the explained variances
features = range(1, 1 + pca.n_components_)
plt.bar(features, pca.explained_variance_)
plt.xlabel('PCA feature')
plt.ylabel('variance')
plt.xticks(features)
plt.show()
```



Conclusion

From our exploratory data analysis and data visualisation, we got good findings and insights.

Based on the models, we decided to pick the logistic regression model as it gives an accuracy score of 81.30% and is apt for the data since the dependent variable is categorical.

Since the model can calculate the probability of a customer leaving the bank, this information can be used by the management to target customers based on their churn probability to retain them.

Conclusion

- Detailed insights and analysis on bank's customers

- Based on the various transformations applied for analytics,
  the *logistic regression model* seems most apt to achieve our objective.
  This can be explained by accuracy score of 81.30% of the model along with the presence of categorical variables in the data

- The model predicts the probability of a customer no longer using the services of the bank, which can be used by the executives to come up with appropriate strategies to retain them

References

Dataset -

https://www.kaggle.com/datasets/gauravtopre/bank-customer-churn-dataset

Code Syntax -

https://www.ritchieng.com/machine-learning-k-nearest-neighbors-knn/

https://datatofish.com/random-rows-pandas-dataframe/

Class Slides (ELMS CANVAS)

Mock Data -

https://www.mockaroo.com/