**Bank Churn Prediction**

Using Pyspark

horizontal line

# 

# Introduction

Bank customer churn prediction is a critical task in the banking industry as it helps the banks to identify the customers who are likely to switch to other banks. It is important for banks to predict customer churn because losing customers can have a significant impact on their revenue and market share. The problem statement for this task is to develop a machine learning model that can accurately predict customer churn based on their historical transaction data, demographic information, and other relevant factors. The model should be able to identify the customers who are at the highest risk of churn so that the bank can take proactive measures to retain them. This involves collecting and analyzing data from various sources, selecting the relevant features, and training the model using appropriate algorithms. We will be using Machine Learning and some data analysis techniques to analyze the bank data which is open-sourced and can be found on Kaggle. We will be looking into the various factors that lead to the churn of a customer from a bank. We will be using python and Pyspark frameworks to complete this project.

## Objective

The ultimate goal of this project is to build a reliable and accurate model that can help banks to reduce customer churn and improve customer satisfaction.

### Research Questions to answer

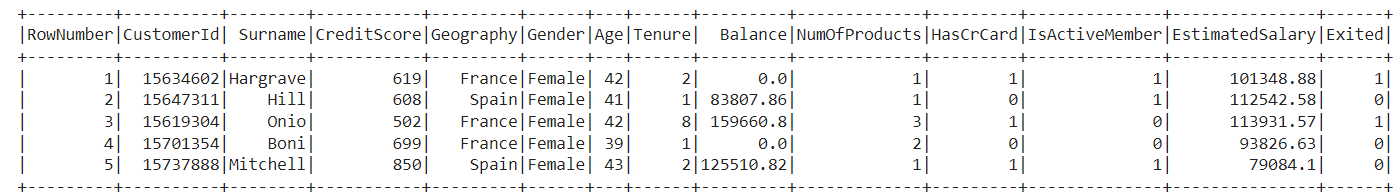
1. Analyze and obtain the factors which are responsible for a customer to churn
2. Is the churn of customers related to the gender of a customer?
3. Does geography or location of customer is also a factor for churn?
4. Does a person having a credit card churn?
5. Build a predictive model that tells us about the chances of a customer to churn so that some extra benefits can be provided by the bank

## 

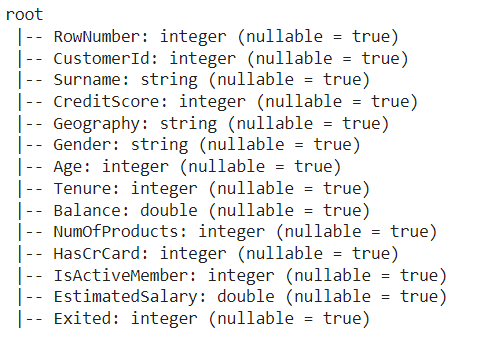
## Milestones

1. Data Loading/Ingestion
2. Exploratory Data Analysis
3. Data Preprocessing
4. Feature Engineering
5. Model Training and Evaluation
6. Insights and Conclusion
7. Comparing plots and analysis using Tableau

### Data Loading/Ingestion

We will be using pyspark for loading data. Pyspark needs to have a pyspark session created inorder to load data. So after creating a session, data is read using the read.csv function of pyspark and the inferschema and header function can be set to True. This will automatically infer the schema of the data and also the header column will also be loaded. Now some basic exploration can be done using the show() function, printschema() function. The show() function will show the first few rows of the data as shown below.  


The printschema() function will show the schema of the data



We have observed that there are around 10000 rows in the data with 14 columns.

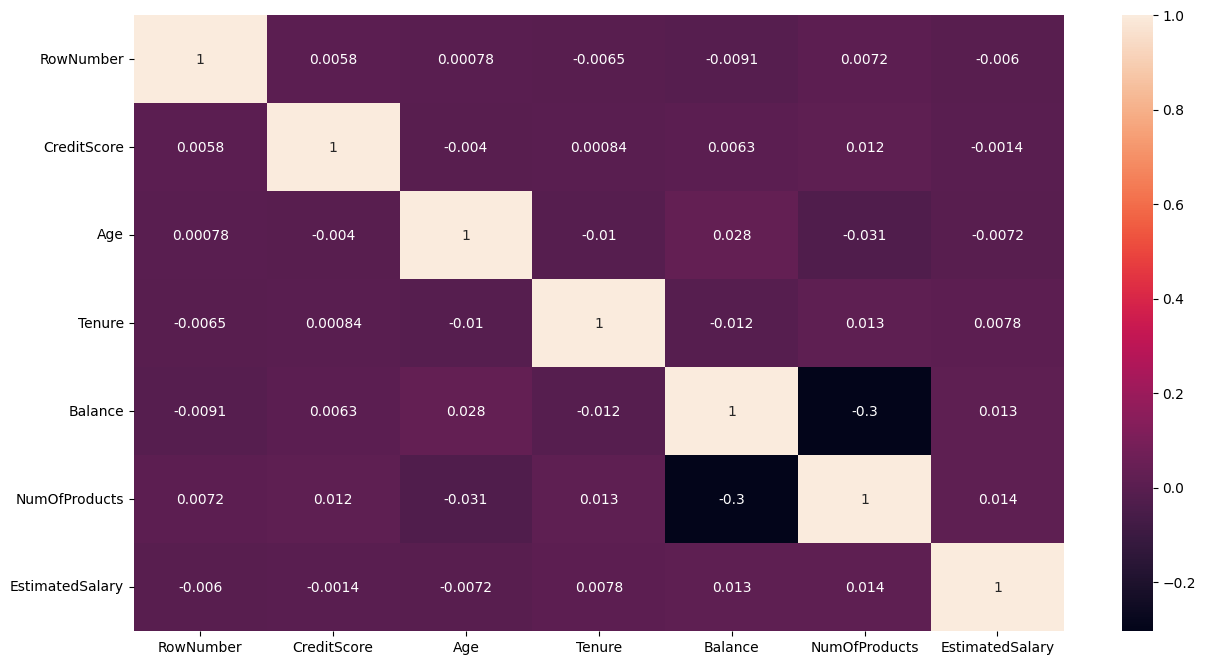
### Exploratory Data Analysis

Exploratory Data Analysis (EDA) is an approach to analyzing data that involves visually and statistically exploring and summarizing the main characteristics of a dataset, such as its distribution, central tendency, spread, correlations, outliers, and missing values.

The main goals of EDA are to understand the underlying patterns and relationships in the data, to identify potential issues or errors in the data, and to generate hypotheses for further investigation or modeling. So now we will perform some EDA on this data. Pyspark dataframe is not well suited for plotting graphs and visualizations so we would convert this to pandas dataframe using toPandas() function and then perform EDA on it.

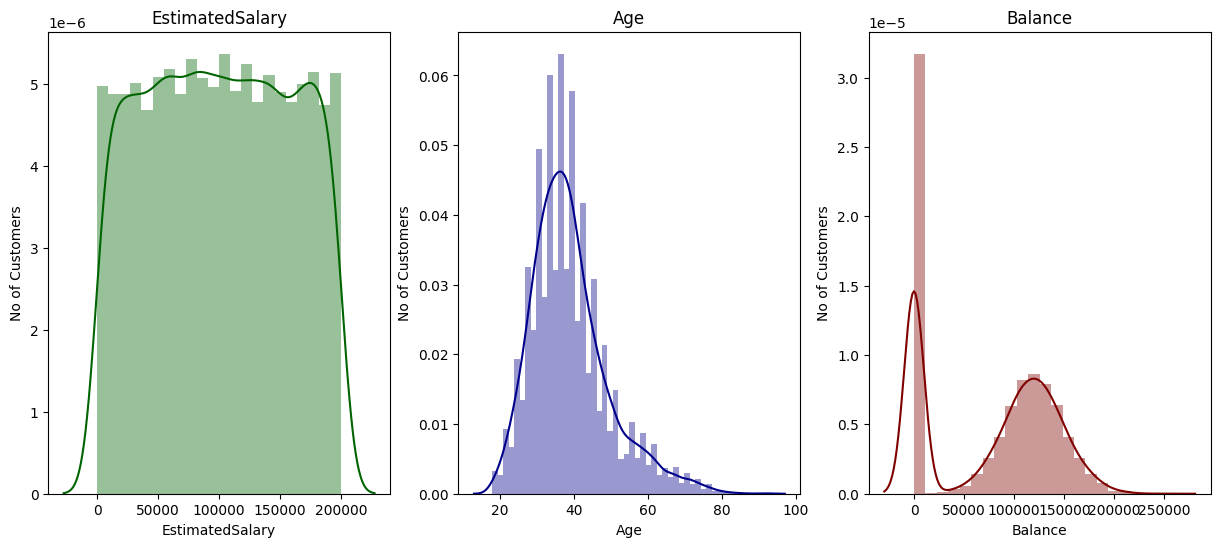
First, we have obtained some statistical analysis using the data.describe() function, we obtained some meaning insights:

* The average credit score for the customers is 650.5, with a standard deviation of 96.6, indicating a moderate variation in credit scores across the customers.
* The average age of the customers is 38.9, with a standard deviation of 10.5, indicating a relatively diverse age range among the customers.
* The average tenure (number of years a customer has been with the bank) is 5.01, with a standard deviation of 2.89, indicating that most customers have been with the bank for at least a few years.
* The average balance across the customers is 76,485, with a standard deviation of 62,397, indicating a large variability in the balance amounts.
* The average number of products held by the customers is 1.53, with a standard deviation of 0.58, indicating that most customers hold between 1 and 2 products with the bank.
* The percentage of customers who have a credit card is 70.6%, while the percentage of active members is 51.5%.
* The average estimated salary for the customers is 100,090, with a standard deviation of 57,510, indicating a wide range of salaries among the customers.
* The churn rate (percentage of customers who left the bank) is 20.4%, indicating a relatively high rate of customer attrition.

Then we obtained the correlation plot using corr() function to observe relations between different features in the data.   


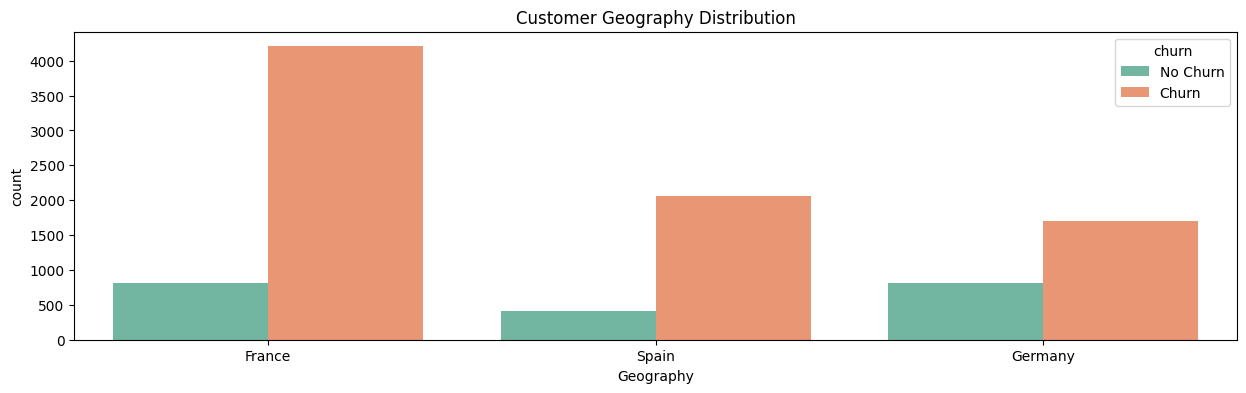
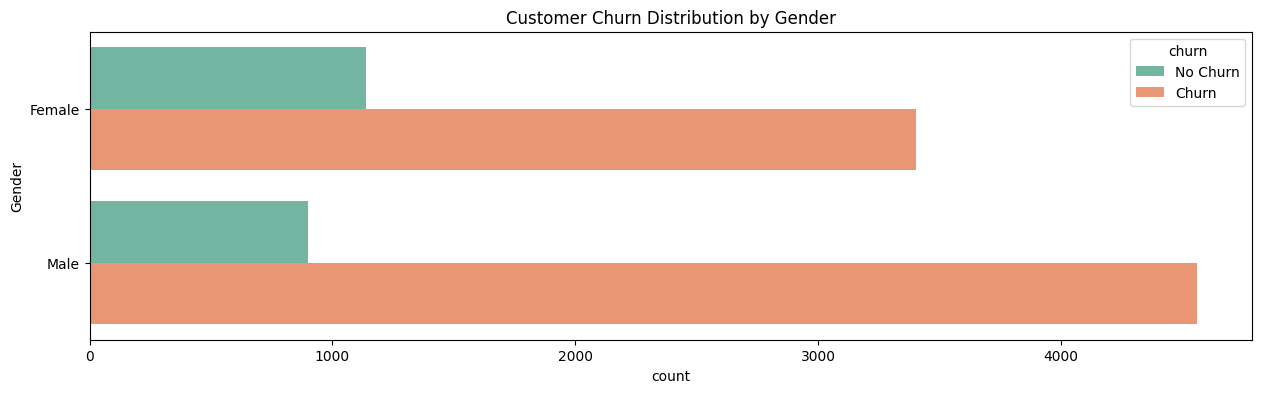
We have also obtained some meaningful observations from this plot:

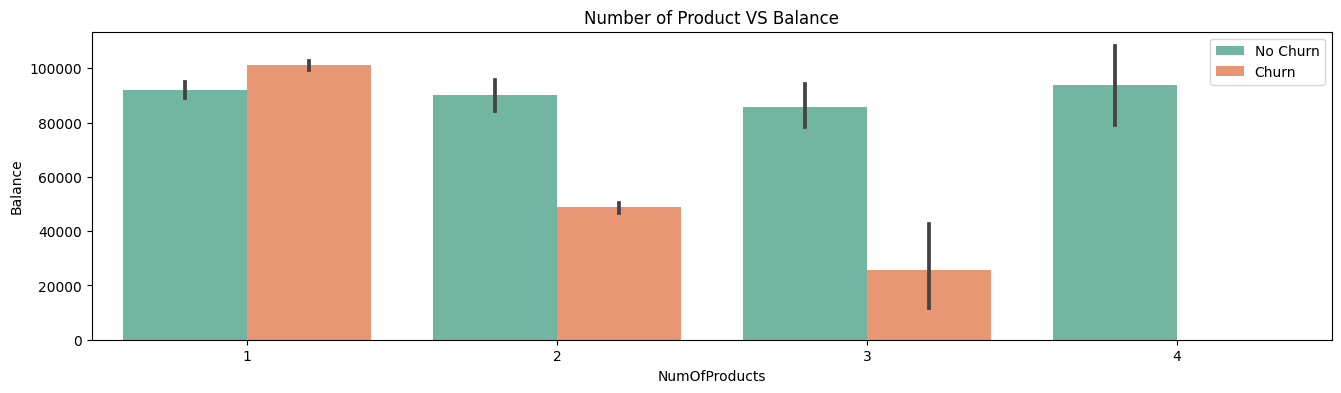
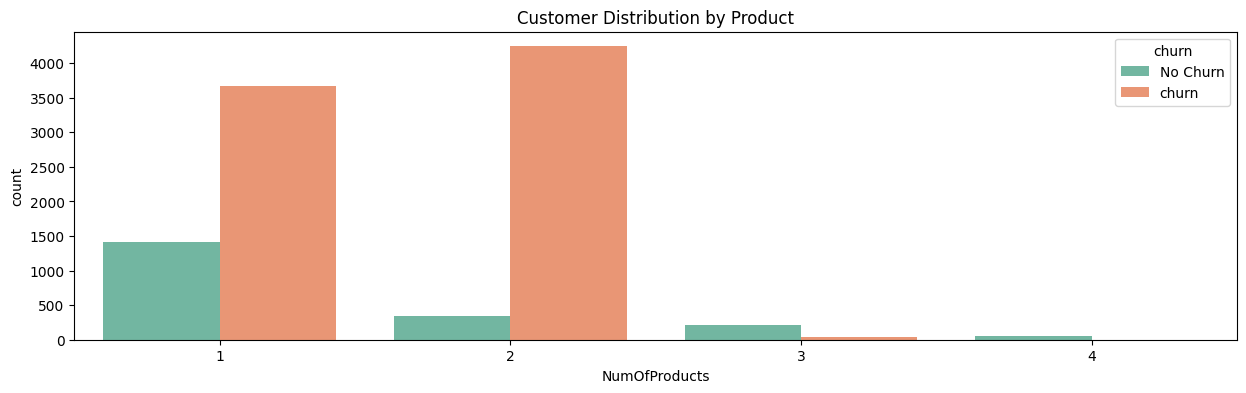
* CreditScore has a weak positive correlation with Age (0.005840), which suggests that older customers tend to have slightly higher credit scores.
* Balance has a weak negative correlation with RowNumber (-0.009067), which suggests that customers with higher row numbers (later in the dataset) tend to have slightly lower balances.
* NumOfProducts has a moderate negative correlation with Balance (-0.304180), which suggests that customers with more products tend to have lower balances.
* Age has a weak negative correlation with NumOfProducts (-0.030680), which suggests that older customers tend to have slightly fewer products.
* Tenure has a weak positive correlation with NumOfProducts (0.013444), which suggests that customers who have been with the bank longer tend to have slightly more products.
* EstimatedSalary has a weak positive correlation with NumOfProducts (0.014204), which suggests that customers with higher estimated salaries tend to have slightly more products.

We have also used some univariate plots to observe the distribution of features  


We can observe that

* EstimatedSalary: The distribution of the estimated salary seems to be a plateau distribution.
* Age: This has a normal distribution that is right skewed. Most customers lie in the range of 30-45 years of age.
* Balance: This has a bimodal distribution. A considerable number of customers with a low balance are there, which seems to be an outlier.

Similarly from the Bivariate analysis, we observed  




Observations,

* Comparatively, more female customers have churned over male customers
* We can observe that the difference between the number of customers that churned and those that did not churn is lesser for Germany and Spain compared with France.
* We can also observe that as the number of products increases, the balance for churned customers remains very high.

### Data Preprocessing

We started preprocessing by observing the data imbalance in data between the churned and not-churned customers.

We also observed that the datatypes of several columns are deviating from what they should be so we have used cast() function to cast the data types as shown below:

| from pyspark.sql.types import StringType data = data.withColumn("HasCrCard",data["HasCrCard"].cast(StringType())) data = data.withColumn("churn",data["churn"].cast(StringType())) data = data.withColumn("IsActiveMember",data["IsActiveMember"].cast(StringType())) |
| --- |

We have also checked for null values in the data and found that there were no null values in the data. We have also tried to observe for duplicates using the duplicated() function but there were no duplicates in the data.

### Feature Engineering

Feature engineering is very important in any machine learning project as this would enable the model to understand the data. We have used different feature techniques for both categorical and numerical attributes:

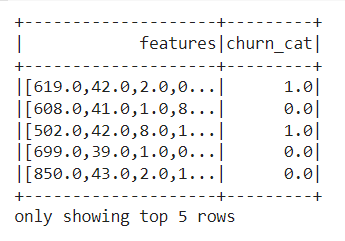
For Categorical attributes - We have used stringindexer to handle the categorical features and transform them. StringIndexer in Pyspark is a data preprocessing technique that is used to convert categorical string features into numerical features. It assigns a unique numerical index to each distinct category in the feature column based on their frequency of occurrence. This transformation is useful for machine learning models that require numerical inputs, as it allows categorical data to be used in the models.

| from pyspark.ml.feature import StringIndexer indexer=StringIndexer(inputCol='churn',outputCol='churn\_cat') indexer.setHandleInvalid("error") indexed=indexer.fit(indexed).transform(indexed) |
| --- |

For Numerical attributes - We have used vectorassembler to handle the numerical features. VectorAssembler is a Pyspark feature that helps combine multiple input columns into a single output column of vectors. It takes a list of input column names and produces a vector column, where each element of the vector corresponds to the value of a specific input column. This can be useful for preparing data for machine learning algorithms that require input data to be in vector form.

| assembler=VectorAssembler(inputCols=[  'CreditScore',  'Age',  'Tenure',  'Balance',  'NumOfProducts',  'EstimatedSalary',  'Geography\_cat',  'Gender\_cat',  'HasCrCard\_cat',  'IsActiveMember\_cat'],outputCol="features") output=assembler.transform(indexed) output.select(['features','churn\_cat']).show(5) |
| --- |

And the output of the vectorassembler is a vector as shown below:



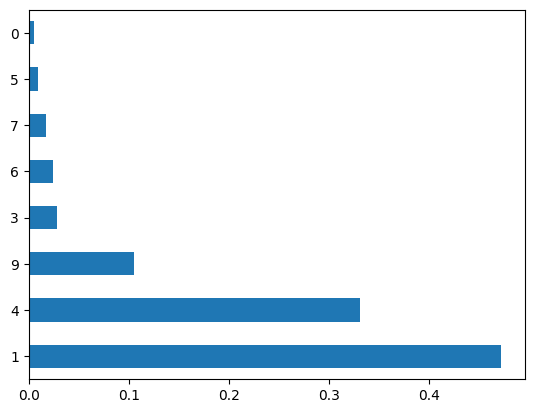
Further we have split our data and tried to apply feature selection to select the best features that can be given to the model.

#### Feature Selection using Random Forest

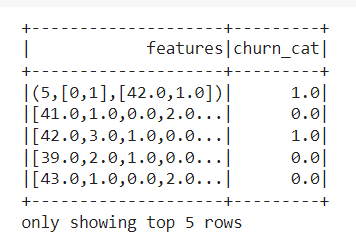
We have used the Random Forest model to select the best features that can be used for the modeling phase. RandomForestClassifier is a machine learning algorithm available in PySpark that can be used for feature selection. It works by creating an ensemble of decision trees, where each tree is trained on a subset of the features. The importance of each feature is then calculated based on the reduction in impurity that it provides, and less important features can be discarded.

| rf = RandomForestClassifier(featuresCol = 'features', labelCol = 'churn\_cat') rfModel = rf.fit(train\_data) predictions = rfModel.transform(test\_data) importances=rfModel.featureImportances |
| --- |

By using the above code we can obtain the feature importance of each column and further depending on the complexity of the model, we can use the top-n features.



Here we can observe that we have the features 1,4,9,6,3 as our top 5 features. These are Age,NumofProducts,IsActiveMember\_cat,Geography\_cat and Balance. So we will be using these features to build our Logistic regression model. We can observe the top-5 features along with the target class as shown below:



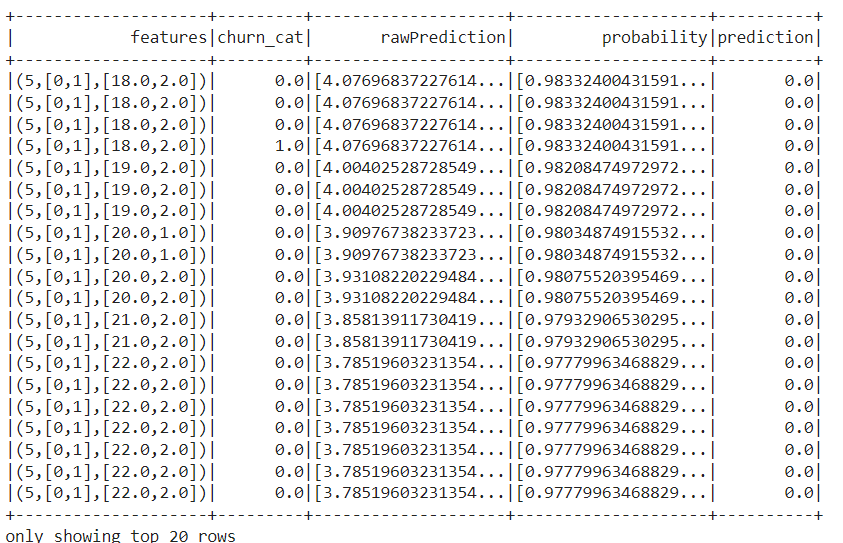
### Model Training and Evaluation

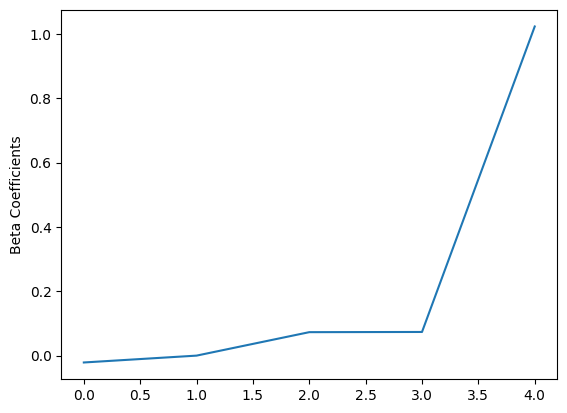
We will be using the top 5 features that we have found using feature selection to build the machine learning model. We will be using the logistic regression model for training.

#### Logistic Regression:

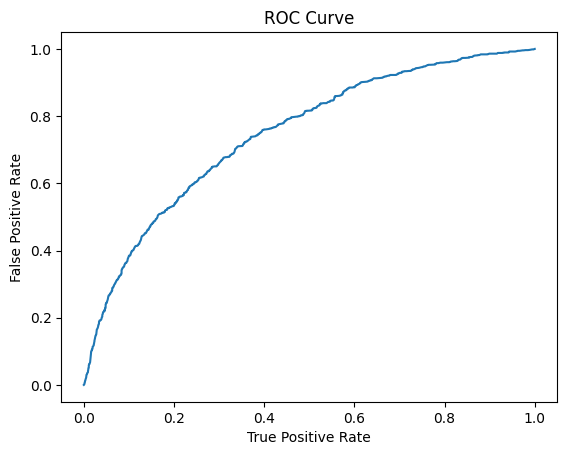
Logistic regression is a statistical method used to analyze and model the relationship between a dependent variable and one or more independent variables. It is a type of regression analysis commonly used to solve binary classification problems, such as predicting whether a customer will churn or not. Logistic regression assumes that the relationship between the dependent variable and independent variables is linear, and it uses a logistic function to convert the linear output into probabilities. These probabilities can be used to make predictions or make decisions based on a given threshold. Logistic regression is a popular method in machine learning and data science due to its simplicity, interpretability, and effectiveness in a wide range of applications.

| from pyspark.ml.classification import LogisticRegression final\_data\_top=output.select(['features','churn\_cat']) train\_data,test\_data=final\_data\_top.randomSplit([0.7,0.3]) lr = LogisticRegression(featuresCol = 'features', labelCol = 'churn\_cat', maxIter=30) lrModel = lr.fit(train\_data) |
| --- |

We have used the Logistic regression model from ml.classification. This model requires two main inputs - one is the featurecol and the other is the labelcol to show the labels. We would then fit this model on the training data. It is a bit different from sklearn implementation of logistic regression as there we can initialize the model directly without any parameters. We can also obtain the predictions from the summary module   


We can also observe the beta coefficients which settle down at 3.0 as shown below  


We have also plotted the ROC curve along with the AUC score of 0.74.



We have further observed all the evaluation metrics as shown below

| trainingSummary = lrModel.summary accuracy = trainingSummary.accuracy falsePositiveRate = trainingSummary.weightedFalsePositiveRate truePositiveRate = trainingSummary.weightedTruePositiveRate fMeasure = trainingSummary.weightedFMeasure() precision = trainingSummary.weightedPrecision recall = trainingSummary.weightedRecall print("Accuracy: %s\nFPR: %s\nTPR: %s\nF-measure: %s\nPrecision: %s\nRecall: %s"  % (accuracy, falsePositiveRate, truePositiveRate, fMeasure, precision, recall)) |
| --- |

And we observed the following scores for each metric

| Metric | Score |
| --- | --- |
| Accuracy | 0.8062767475035664 |
| FPR | 0.6838834518251322 |
| TPR | 0.8062767475035664 |
| F-measure | 0.7572353203010409 |
| Precision | 0.7709674391484447 |
| Recall | 0.8062767475035664 |

Based on the above table, it seems like the model has a decent overall accuracy of 0.81, meaning that 81% of the predictions made by the model were correct. However, the high false positive rate (FPR) of 0.68 suggests that the model may be incorrectly classifying a significant number of negative instances as positive. The true positive rate (TPR) is equal to the accuracy, indicating that the model is effectively identifying positive instances.

The F-measure, which is the harmonic mean of precision and recall, is 0.76, indicating that the model has a good balance between precision and recall. The precision of 0.77 indicates that when the model predicts a positive instance, it is correct 77% of the time. The recall of 0.81 suggests that the model is able to identify 81% of the positive instances in the dataset.

Overall, while the model has a good accuracy and balance between precision and recall, it may require further optimization to reduce the high false positive rate.

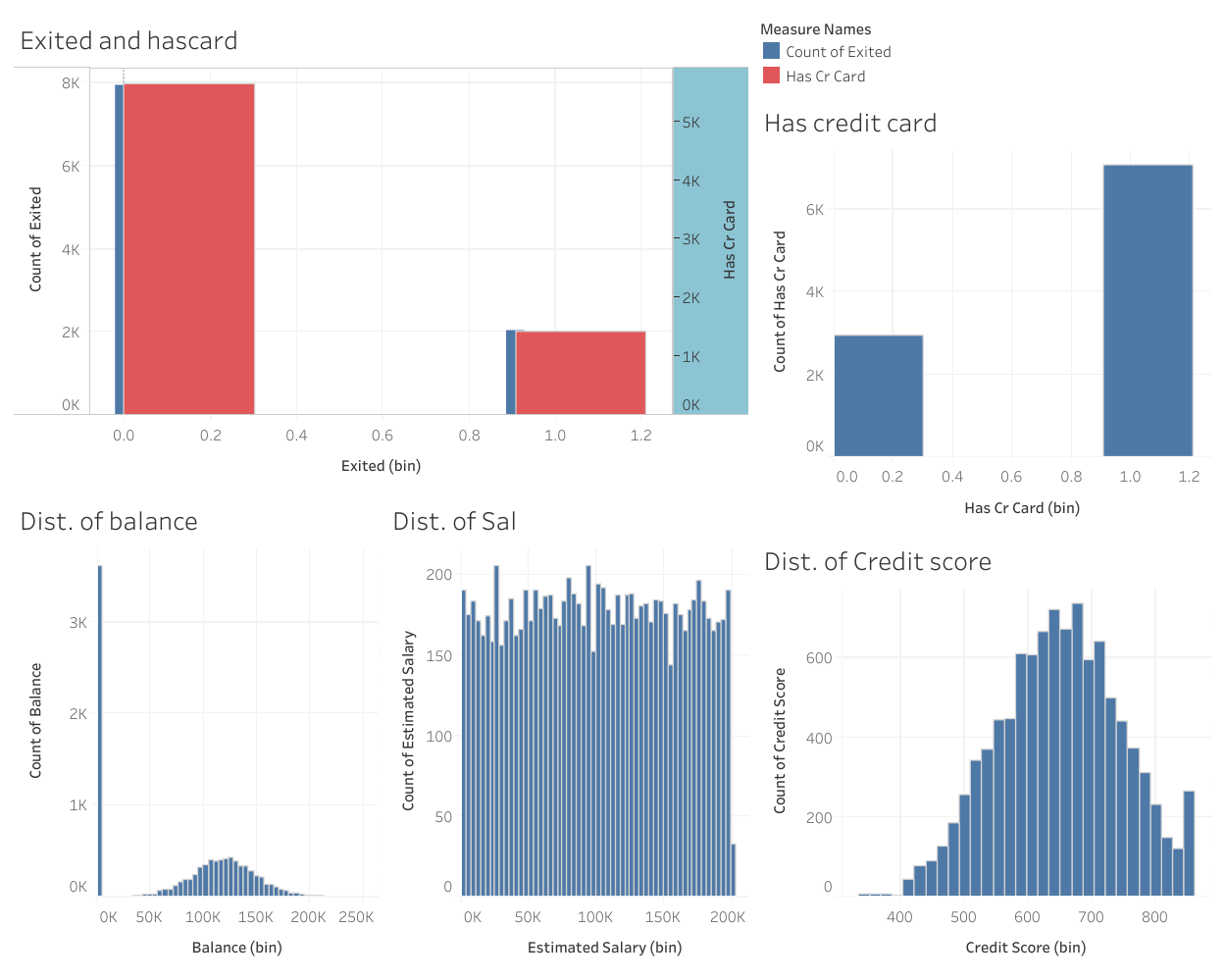
### Insights

* The data indicates that 20.37% or 2,037 customers churned, while 79.63% or 7,963 customers did not churn.
* The analysis reveals that the average credit score for churned customers is 645.35 and their average age is 44.83 years.
* The average balance and estimated salary for churned customers were found to be higher than those of customers who did not churn, at 911,108.53 and 101,465.67, respectively.
* The correlation analysis reveals that there is a positive correlation of 29% between customer age and churn.
* The data indicates a positive correlation of 12% between balance and churn.
* The correlation analysis shows a negative correlation of 30% between the number of products and balance.
* The difference in churn rates between customers in Germany and Spain compared to France is noteworthy.
* The data reveals that more female customers have churned and that customers with 3-4 products are more likely to churn.
* The analysis indicates that customers with a negative-to-low balance are less likely to churn compared to customers with a balance of 75,000–150,000.
* The most important features selected using a Random forest-based feature selection approach are Age, NumOfProducts, IsActiveMember, Geography\_cat, and Balance.

### Conclusion

* Marketing teams often rely on predicting customer churn to optimize their campaigns and allocate resources effectively. It is a crucial application of analytics in the field.
* This project explores the data science pipeline and its application in machine learning. The focus is on logistic regression, and the differences between it and linear regression are explained.
* The data is prepared for exploration using pandas DataFrame, and data scrubbing techniques such as missing value imputation, column renaming, and data type manipulation are used.
* Various data visualization techniques, such as univariate, bivariate, and correlation plots, are used to extract insights from the data and identify useful patterns.
* Feature selection is essential in building a good machine learning model. A Random-forest-based classifier is employed to identify the most important features.
* Logistic regression is implemented to calculate the likelihood of customer churn, which is the final step in the process.

### Some Comparison Plots using Tableau



From this dashboard, we can clearly visualize the distribution of credit score which seems to be gaussian and the distribution of balance which even seems to be gaussian and the number of people using credit cards and the comparison of people having cards and churned from the bank.

## Research questions answered simply after analysis:

1. The female customers seem to churn more than male customers.
2. More customers seem to churn from spain than any other country
3. People having credit cards seem to churn more than people not using credit cards.

## Tech Stack used:

Jupyter notebook, Python, Pyspark, Pandas, Tableau, Machine Learning frameworks.