# Customer Churn Prediction with Python

**Customer churn** is a financial term that refers to the loss of a client or customer—that is, when a customer ceases to interact with a company or business. Similarly, the **churn rate** is the rate at which customers or clients are leaving a company within a specific period of time. A churn rate higher than a certain threshold can have both tangible and intangible effects on a company's business success. Ideally, companies like to retain as many customers as they can.

With the advent of advanced data science and machine learning techniques, it's now possible for companies to identify potential customers who may cease doing business with them in the near future. In this article, you'll see how a bank can predict customer churn based on different customer attributes such as age, gender, geography, and more. The details of the features used for customer churn prediction are provided in a later section.

**Project Statement :-**

Customer churn is when a company’s customers stop doing business with that company. Businesses are very keen on measuring churn because keeping an existing customer is far less expensive than acquiring a new customer. New business involves working leads through a sales funnel, using marketing and sales budgets to gain additional customers. Existing customers will often have a higher volume of service consumption and can generate additional customer referrals.

Customer retention can be achieved with good customer service and products. But the most effective way for a company to prevent attrition of customers is to truly know them. The vast volumes of data collected about customers can be used to build churn prediction models. Knowing who is most likely to defect means that a company can prioritise focused marketing efforts on that subset of their customer base.

### Overview: Using Python for Customer Churn Prediction

Taking a closer look, we see that the dataset contains 14 columns (also known as features or variables). The first 13 columns are the independent variable, while the last column is the dependent variable that contains a binary value of 1 or 0. Here, 1 refers to the case where the customer left the bank after 6 months, and 0 is the case where the customer didn't leave the bank after 6 months. This is known as a **binary classification problem**, where you have only two possible values for the dependent variable—in this case, a customer either leaves the bank after 6 months or doesn't.

It's important to mention that the data for the independent variables was collected 6 months before the data for the dependent variable, since the task is to develop a machine learning model that can predict whether a customer will leave the bank after 6 months, depending on the current feature values.

Here's an overview of the steps we'll take in this article:

1. Importing the libraries
2. Loading the dataset
3. Selecting relevant features
4. Converting categorical columns to numeric ones
5. Preprocessing the data
6. Training a machine learning algorithm
7. Evaluating the machine learning algorithm
8. Evaluating the dataset features

#### Step 1: Importing the Libraries. The first step, as always, is to import the required libraries.

#### Step 2: Loading the Dataset. The second step is to load the dataset from the local CSV file into your Python program. Let's use the read\_csv method of the pandas library.

#### Step 3: Feature Selection.As a reminder, there are 14 columns total in our dataset.

1. **RowNumber** orresponds to the record (row) number and has no effect on the output. This column will be removed.
2. **CustomerId** ontains random values and has no effect on customer leaving the bank. This column will be removed.
3. **Surname** he surname of a customer has no impact on their decision to leave the bank. This column will be removed.
4. **CreditScore** an have an effect on customer churn, since a customer with a higher credit score is less likely to leave the bank.
5. **Geography**  customer's location can affect their decision to leave the bank. We'll keep this column.
6. **Gender I**t's interesting to explore whether gender plays a role in a customer leaving the bank. We'll include this column, too.
7. **Age** his is certainly relevant, since older customers are less likely to leave their bank than younger ones.
8. **Tenure r**efers to the number of years that the customer has been a client of the bank. Normally, older clients are more loyal and less likely to leave a bank.
9. **Balance** lso a very good indicator of customer churn, as people with a higher balance in their accounts are less likely to leave the bank compared to those with lower balances.
10. **NumOfProducts** refers to the number of products that a customer has purchased through the bank.
11. **HasCrCard d**enotes whether or not a customer has a credit card. This column is also relevant, since people with a credit card are less likely to leave the bank.
12. **IsActiveMember a**ctive customers are less likely to leave the bank, so we'll keep this.
13. **EstimatedSalary I**s with balance, people with lower salaries are more likely to leave the bank compared to those with higher salaries.
14. **Exited w**hether or not the customer left the bank. This is what we have to predict.

After careful observation of the features, we'll remove the RowNumber, CustomerId, and Surname columns from our feature set. All the remaining columns do contribute to the customer churn in one way or another.

Step 4: Converting Categorical Columns to Numeric Columns. Machine learning **algorithms work best with numerical data**. However, in our dataset, we have two categorical columns: Geography and Gender. These two columns contain data in textual format; we need to convert them to numeric columns..Let's first isolate these two columns from our dataset.One way to convert categorical columns to numeric columns is to replace each category with a number. For instance, in the Gender column, female can be replaced with 0 and male with 1, or vice versa. This works for columns with only two categories.For a column like Geography with three or more categories, you can use the values 0, 1, and 2 for the three countries of France, Germany, and Spain. However, if you do this, the machine learning algorithms will assume that there is an **ordinal relationship** between the three countries. In other words, the algorithm will assume that 2 is greater than 1 and 0, which actually is not the case in terms of the underlying countries the numbers represent.

A better way to convert such categorical columns to numeric columns is by using **one-hot encoding**. In this process, we take our categories (France, Germany, Spain) and represent them with columns. In each column, we use a 1 to designate that the category exists for the current row, and a 0 otherwise.

In this case, with the three categories of France, Germany, and Spain, we can represent our categorical data with just two columns (Germany and Spain, for example). Why? Well, if for a given row we have that Geography is France, then the Germany and Spain columns will both have a 0, implying that the country must be the remaining one not represented by any column. Notice, then, that we do not actually need a separate column for France.

Let's convert both the Geography and Gender columns into numeric columns. The get\_dummies method of the pandas library converts categorical columns to numeric columns. Then, .iloc[:,1:] ignores the first column and returns the rest of the columns (Germany and Spain). As noted above, this is because we can always represent "n" categories with "n - 1" columns.

Now if you open the **Geography** and **customer\_data** data frames in the Variable Explorer pane.

In accordance with our earlier explanation, the **Geography** data frame contains two columns instead of three. When the geography is France, both Germany and Spain contain 0. When the geography is Spain, you can see a 1 in the Spain column and a 0 in the Germany column. Similarly, in the case of Germany, you can see a 1 in the Germany column and a 0 in the Spain column.

#### Next, we need to add the ****Geography**** and ****Gender**** data frames back to the data set to create the final dataset. You can use the concat function from pandas to horizontally concatenate two data frames

#### Step 5: Data Preprocessing. Our data is now ready, and we can train our machine learning model. But first, we need to isolate the variable that we're predicting from the dataset. Here, X is our feature set; it contains all the columns except the one that we have to predict (Exited). The label set, y, contains only the Exited column. So we can later evaluate the performance of our machine learning model, let's also divide the data into a training and test set. The training set contains the data that will be used to train our machine learning model. The test set will be used to evaluate how good our model is. We'll use 20% of the data for the test set and the remaining 80% for the training set (specified with the test\_size argument)

#### Step 6: Machine Learning Algorithm Training.Now, we'll use a machine learning algorithm that will identify patterns or trends in the training data. This step is known as ****algorithm training****. We'll feed the features and correct output to the algorithm; based on that data, the algorithm will learn to find associations between the features and outputs. After training the algorithm, you'll be able to use it to make predictions on new data.

There are several machine learning algorithms that can be used to make such predictions. However, we'll use the [RANDOM FOREST ALGORITHM](https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html" \o "3.2.4.3.1. sklearn.ensemble.RandomForestClassifier 鈥" \t "/Users/akshatgaur/Documents\\x/_blank), since it's simple and one of the most **powerful algorithms for classification problems**. To train this algorithm, we call the fit method and pass in the feature set (X) and the corresponding label set (y). You can then use the predict method to make predictions on the test set.

Step 7: Machine Learning Algorithm Evaluation.

Now that the algorithm has been trained, it's time to see how well it performs. For evaluating the performance of a classification algorithm, the most commonly used metrics are the [F1 MEASURE, PRECISION, RECALL, AND ACCURACY](https://towardsdatascience.com/accuracy-precision-recall-or-f1-331fb37c5cb9" \o "Accuracy, Precision, Recall or F1? 鈥" \t "/Users/akshatgaur/Documents\\x/_blank). In Python's scikit-learn library, you can use built-in functions to find all of these values.The results indicate an accuracy of **86.35%**, which means that our algorithm successfully predicts customer churn 86.35% of the time. That's pretty impressive for a first attempt!

#### Step 8: Feature Evaluation

As a final step, let's see which features play the most important role in the identification of customer churn. Luckily, RandomForestClassifier contains an attribute named feature\_importance that contains information about the most important features for a given classification.Based on this data, we can see that age has the highest impact on customer churn, followed by a customer's estimated salary and account balance.

Conclusion.

Customer churn prediction is crucial to the long-term financial stability of a company. In this article, you successfully created a machine learning model that's able to predict customer churn with an accuracy of 86.35%. You can see how easy and straightforward it is to create a machine learning model for classification tasks.