|  |
| --- |
| Technical Document  Les Clairvoyants |
| **Group members:**   * Chadi Ghosn * Kirran Kayani * Rita Starzyk * YEVGENIY DENKOVYCH |



Contents

[Project Diagram: 3](#_Toc119173347)

[ETL Process: 4](#_Toc119173348)

[Tableau: 5](#_Toc119173349)

[ML Models: 6](#_Toc119173350)

[1. Logistic Regression: 6](#_Toc119173351)

[2. K nearest neighbor: 7](#_Toc119173352)

[3. Random Forest Classifier: 7](#_Toc119173353)

[AI Selenium code: 8](#_Toc119173354)

# A picture containing diagram Description automatically generatedProject Diagram:

The project will start by ingesting a csv dataset into an ETL process for data curation. Once done, a csv of the curated data will be exported to be used with Tableau for data visualization. On the other hand, the curated data will as well be pushed to a PostgreSQL database for machine learning usage. Three different algorithms of machine learning will be applied to try and reach the highest model score available. Finally, the selenium code will contact an online website for AI, where users can input information, as well as the model accuracy achieved, and the website will return insights and percentages of customer retention as well as revenue saving.

# ETL Process:

We start by importing necessary libraries for data curation. Then we read the csv file downloaded from [Kaggle](https://www.kaggle.com/competitions/customer-churn-prediction-2020/data).

The dataset was already split into train, test and sample submission, so we had to append all of the data together to be able to control the split between training and testing data.

Afterwards, we checked that the columns did not have any null values and checked the columns data types to make sure that the int or float columns did not have any strings value that would affect the column data type.

Moreover, we mapped the churn column by replacing yes by 1 and no by 0. Then we dropped irrelevant columns like “State” and “Are code”.

Next, we changed all string values to numerical values to prepare the data for machine learning model usage. And we checked the number of categories that we have by checking the unique values in the columns. We then checked again for null values and data types to make sure data now is ready to be ingested.

Finally, we exported the data as a csv file that will be used with Tableau, and we loaded the data into a PostgreSQL database called “churn\_pred” into a fact table by the name of “fact\_churn”.

This file as well include the selenium code that we will touch on later.

# Tableau:

We created a [Tableau](https://public.tableau.com/app/profile/chadi.ghosn/viz/Project4_16680253761120/Dashboard1?publish=yes) dashboard that helps the business check for any correlation or trend in their data.

The dashboard consists of 7 visuals in total:

* The first 4 visuals are comparing the account lengths (in months) of the account holders, with the total number of minutes used by day, eve, night and international, separately.
* Then we are looking at the number of churns by the date the customers signed up for a contract with the telecom company.
* In addition, we looked at the number of customer service calls by the account length to check for any trend.
* Finally, we compared the churn by day, and the count of sign ups by day, as well as the percentage churn by day for reference.

Based on the visuals created, we were able to conclude the following:

* There is no positive or negative correlation between the total number of minutes and the account lengths.
* Between 2012 and 2014:
  + The company registered the highest number of customers.
  + These customers made the highest numbers of calls until 2020, and also the highest number of calls to customer service.
  + The number of customers who churned was also highest between mentioned years.
* The churn rate remained relatively stable for these customers (avg. 14%), thus:
  + No relationship between the number of months the customer has stayed with the company and the churn rate.

# ML Models:

## Logistic Regression:

We randomly split and scaled the data for training and testing, keeping 25% of the data for testing the model. Both training and testing scores were relatively high and close together after applying the Logistic Regression model, at 83.3% and 81.4% respectively, and a total accuracy of 81%. This was achieved through tuning the parameters of the model: ‘max\_iter’, ‘solver’, ‘penalty’, and ‘l1\_ratio’.

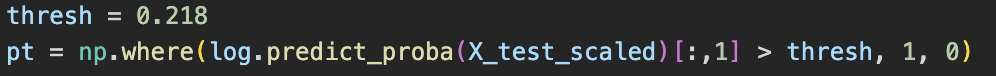
Text

Description automatically generated

Chart, treemap chart

Description automatically generatedHowever, the proportion of False Negative results was 16.7%. These customers would churn (leave the company) while our model predicted them to stay, and so would be lost customers. In addition, only 2% of the results were True Positives (where our model correctly predicted the customers likely to churn). In comparison, the proportion of False Positive results was low at 2%; these customers are predicted to churn however would remain with the company. As it is more important for our model to predict those who leave, so the company can carry out any necessary retention strategies, we decided to optimize our model to increase its sensitivity.

Chart, treemap chart

Description automatically generatedThrough amending the probability threshold from the default of 0.5 to 0.218, we were able to more than halve the proportion of False Negative results to 8%, while increasing the proportion of True Positives to 11%. Although total accuracy reduced to 77% and the proportion of False Positive results increased, the benefit received from the higher sensitivity would outweigh any cost incurred from the additional False Positive customers.

## K nearest neighbor:

Chart

Description automatically generated

Data was randomly split for training and testing. 75% was split for training and 25% for testing. Data was scaled and initial run of the model was initialized to access the performance of the model with various n neighbors hyperparameter, resulting in an Elbow graph (Figure1). Model has scored quite well with its performance plateau nearing n neighbors of 6. Models’ accuracy resulted nearing 83.5% for training and 81.1% for testing. To optimize model hyperparameters such as: leaf size, n neighbors, and p value we used GridSearchCV library, after it optimized the necessary values, model was run with leaf size =1, n neighbors = 11, and p value = 1. This model’s accuracy was the same as the one that was run the first time. This tells us that model is already performing as best as it can. Finally, for the final model Confusion Matrix was created.

Chart, treemap chart

Description automatically generatedChart, treemap chart

Description automatically generatedAs it can be seen from the confusion matrix heatmap (Figure 2), there is a significant proportion of clients that were predicted to stay but instead they would churn, also only a tiny percentage of people were predicted to churn correctly. To address the issue, sensitivity threshold was implemented. Through trial-and-error threshold value of 0.23 was used. Which drastically improved the rate of True Positives and reduced the rate of False Negatives, though the rate of False Positives were increased. We believe that the improvement to the model outweighs the negatives. (Figure 3)

## Random Forest Classifier:

Data was randomly split for training and testing: 75% for training and 25% for testing. Then data was scaled by using Standard Scaler. Random Forest Classifier was created with Training Data Score of 95% and Testing Data Score of 88%. That was achieved by tuning the hyper-parameters:   
criterion – entropy, max depth of 10, random state of 42 and balanced class weigh.

Chart, treemap chart

Description automatically generated

Although the accuracy of 88% was achieved, the false negatives were quite high, therefore model was optimized with the threshold of 0.33. By making this change, false negatives were lowered to 91, which is 7% of all customers.

Although false positives increased, we believe that improvement we made outweighs the extra cost company will face.

Chart, treemap chart

Description automatically generated

# AI Selenium code:

We started by importing Chrome Driver to access the browser.

Then we’ve set the chrome options to have a “headless” browser, which means the process will be hidden.

Once the hidden browser launches, it will access the URL specified and start applying the set buttons:

* It will accept the cookies based on the html XPath.
* It will ask 5 questions and temporarily store them as variables.
* Then it will find the relevant fields by XPath and fill in the fields with the respective variables.
* Finally, it will click the button to process the numbers and bring back the result and print them.

Text, letter

Description automatically generated