

# Churn Prediction with Logistic Regression

## Intelligent Systems Technologies

Alberto Castañeda Arana

**Abstract**—Churn prediction is critical for many businesses that need to identify customers that are at risk of cancelling the business service. By predicting these customers, businesses can reduce costs of acquire new clients by retaining the existing ones. Given that the churn prediction is a binary classification problem, in this project a Logistic Regression model made from scratch to predict was proposed to predict if a customer will cancel a service or 'churn' given his or her recorded attributes.

### I. INTRODUCTION

Churn refers to a regular, quantifiable process or rate of change that occurs in a business over a period of time as existing customers are lost and new customers are added [1]. While every customer that cancels a business service has a different reason, some even personal reasons, there are factors that contribute to this decision that businesses look for in order to predict and find out improvement areas of their operations so that their churn rate can become lower.

Given the previously mentioned factors of customers, the purpose of this project is to create a tool that predicts whether a customer is likely to cancel the service or 'churn' so that further efforts can be made to retain that customer since it is cheaper to retain current customers in businesses rather than obtaining new ones.

The problem can be categorized as a binary classification problem where what is trying to be predicted is if a customer is likely to churn or not. To tackle this problem there is multiple classification machine learning algorithms that given data are able to predict this binary outcome. For this project, a logistic regression model made from scratch was proposed.

For this project, a Logistic Regression model made from scratch is proposed to handle this classification problem. Additionally, the model will be fitted alongside other implementations from the Python SciKit library to compare the accuracy and fitting speed of each one of them.

### II. THEORY

#### II-A. Logistic Regression

Logistic Regression is a statistical method used to obtain the probability of a certain class existing based on input variables that are defined for the model.

In Machine Learning, logistic regression can be used as a tool to create accurate predictions of outcomes given some predictor variable data.

While there is multiple types of logistic regression depending on what needs to be achieved, for this case a binary logistic regression is needed in order to predict a churn or not churn prediction. Like the name of the model implies, a binary result will be given to represent the outcome; the value being either 1, or 0. As previously mentioned, logistic regression outputs a continuous number between 0 and 1, which is why we must define a threshold (defined as 0.5 by default) in order to transform this continuous value to a 0 or 1 value.

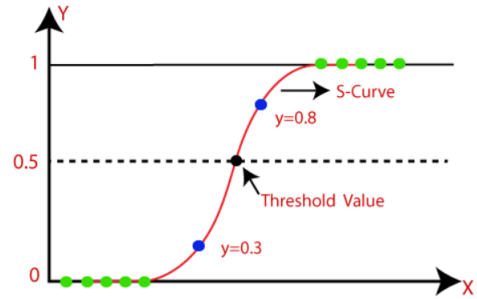


Fig. 1. Logistic Regression graph with threshold for classification.

Figure 1 could be represented by a sigmoid function, which has the formula:

$$y = g(x) = \frac{1}{1 + e^{-x}} \quad (1)$$

Fig. 2. Sigmoid Function

The function for the logistic regression would be defined as:

$$h_{\theta}(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}} \quad (2)$$

Fig. 3. Logistic Regression Function

#### II-B. Cost Function

Cost functions are used to determine the minimization needed to make models have a better performance. The function reflects how much the predicted output differs from the original output with the goal of optimizing the model. In other words, we use the cost function to measure how close the model's predictions are to the real outputs.

$$J(\theta) = -\frac{1}{m} \sum_{i=1}^m y^i \log h_{\theta}(x^i) + (1-y^i) \log(1-h_{\theta}(x^i)) \quad (3)$$

Fig. 4. Cost function formula

### II-C. Gradient Descent

Gradient Descent is an iterative optimization algorithm which finds the minimum of a diferentiabe function. In this case, Gradient Descent can be applied to the cost function of logistic regression to find an optimizal solution. By applying the previously established hypothesis function to the gradient descent formula we would get:

$$\theta_j = \theta_j - \alpha \frac{1}{m} \sum_{i=1}^m ((h_{\theta}(x^i) - y^i) x_j^i) \quad (4)$$

Fig. 5. Gradient descent with logistic Regression formula formula

## III. DATA

### III-A. Dataset

For this project, the dataset chosen to train the models is the Telco customer churn dataset from Kaggle [2]. The dataset contains the Telco Company customer's characteristics and contract information while also indicating whether the customer departed within the last month.

### III-B. Data Analysis

From the dataset, the proportion of how many customers were classified as churn and not churn is as follows:

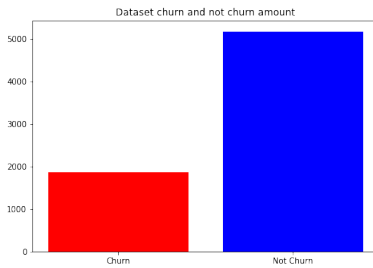


Fig. 6. Churn / Not Churn proportion.

As we can see on the previous figure, we have an approximate 1:3 ratio of churn and not churn customers which makes the training contemplate both of these scenarios with sufficient data.

By doing exploratory data analysis on the dataset, the following observations were made.

- CustomerID is a primary key and does not give any useful information.
- Gender does not seem to be a good predictor.
- Customer doing paper-less billing or not does not seem to be a good predictor.
- Senior Citizens are more likely to churn.

- Customers with no partners or dependents are more likely to churn.
- Short-term customers, or customers with lower tenure, are more likely to churn.
- Customers that do not have services (online security, online backup, device protection, tech support) are more likely to churn.
- Customers with lower total charges are more likely to churn.

By doing the previous observations, we can conclude that most of the features in the dataset can be used as a predictor with the exception of:

- CustomerID
- Gender
- Paper-less billing

As such, these columns will be dropped from the feature extraction for the models fitting.

Additionally, empty cells in the TotalCharge column were found in some rows. For this, an approximation calculation of tenure \* MonthlyCharge was done to approximately represent the TotalCharge and to fill these empty cells.

## IV. PREPROCESSING

After analyzing the data and making the previous observations, it is now time to clean and transorm the data in order to train our proposed models.

As mentioned in the previous section, we first drop columns that do not seem to give good predictors. Afterwards, since machine learning algorithms require purely numerical values for fitting, we must transform each categorical feature to a numerical type. To do this, we first transform each Yes/No classification column as a binary representation (0 for no, 1 for yes) which also includes our target variable Churn.

One problem is presented with some categorical values which have more than 2 possible labels. For example: the InternetService feature can possibly have three labels (DLS, Fiber Optic, No Internet Service). To tackle this problem, one-hot encoding was used to transform these features into multiple numerical features.

id	color			
1	red			
2	blue			
3	green			
4	blue			

One Hot Encoding →

id	color_red	color_blue	color_green
1	1	0	0
2	0	1	0
3	0	0	1
4	0	1	0

Fig. 7. One Hot Encoding transformation

Additionally, as mentioned in the previous section there are some empty cells that need to be replaced. From the dataset analysis, only the total charge feature has these

empty cells so a  $\text{MonthlyCharge} * \text{tenure}$  calculation is used to fill these cells.

Lastly, since our model will be working with a sigmoid function, we must normalize our values to prevent overflow errors that can be thrown with the exponential divider. Initially, the model did not scale the features which caused multiple overflow warnings and lower accuracy overall.

To do this, the `StandardScaler` class from the SciKit library is used to scale the training and test data.

## V. RESULTS

### V-A. Fitting

In order to test our models, cross validation was used with 20 % of the data used for testing.

The Logistic Regression model made from scratch was trained with 8000 epochs and a learning rate of 0.001. The following chart represents the cost function compared to each epoch.

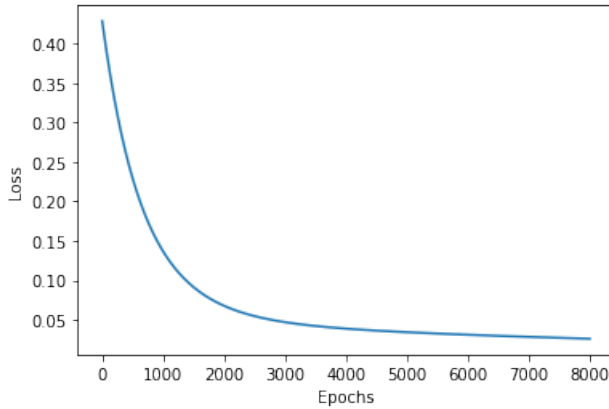


Fig. 8. Epochs vs Cost Function

While the model can be trained with more epochs, the training time would become too big for the scope of this project.

### V-B. Accuracy

After fitting each model, the following prediction accuracy percentages were given per model:

As seen on the plot, our logistic regression model got approximately 78.1 % accuracy during testing, slightly lower than the SciKit framework implementation. However, the previous two models consistently got 6 % higher accuracy than the SciKit framework implementation of a Decision Tree Classifier.

### V-C. Training time

As seen on the plot, our logistic regression model was approximately 10x time slower than other models from the SciKit framework with the same data. The reason for

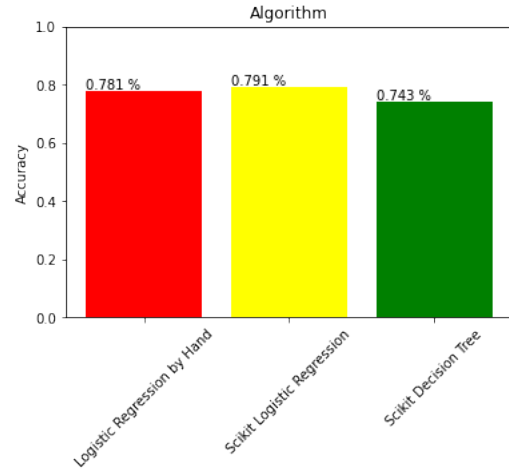


Fig. 9. Accuracy per model

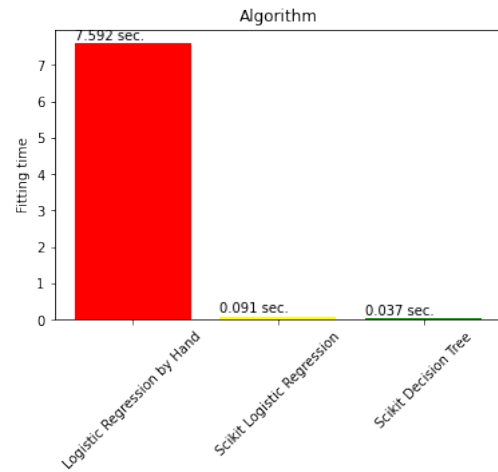


Fig. 10. Training time per model

this drastic difference could be attributed to the lack of parallelization in the model made from scratch, which SciKit uses to improve training speed performance.

### V-D. Confusion Matrix

As seen on the previous vector of confusion matrix, the scratch logistic regression model has approximately equal False Negatives compared to False Positives. On the other hand, the SciKit implementation has a higher ratio of False Negatives compared to False Positives. Lastly, the SciKit implementation of Decision Tree ratio of False Negatives to False Positives is similar to the first model but higher.

### V-E. Receiver Operating Characteristic Curve

A Receiver Operating Curve (ROC) is a graph showing the performance of a classification model at all classification thresholds. This curve plots two parameters: True Positive Rate and False Positive Rate.

The curve is compared with the threshold and a larger the area under the curve (AUC) is usually better.

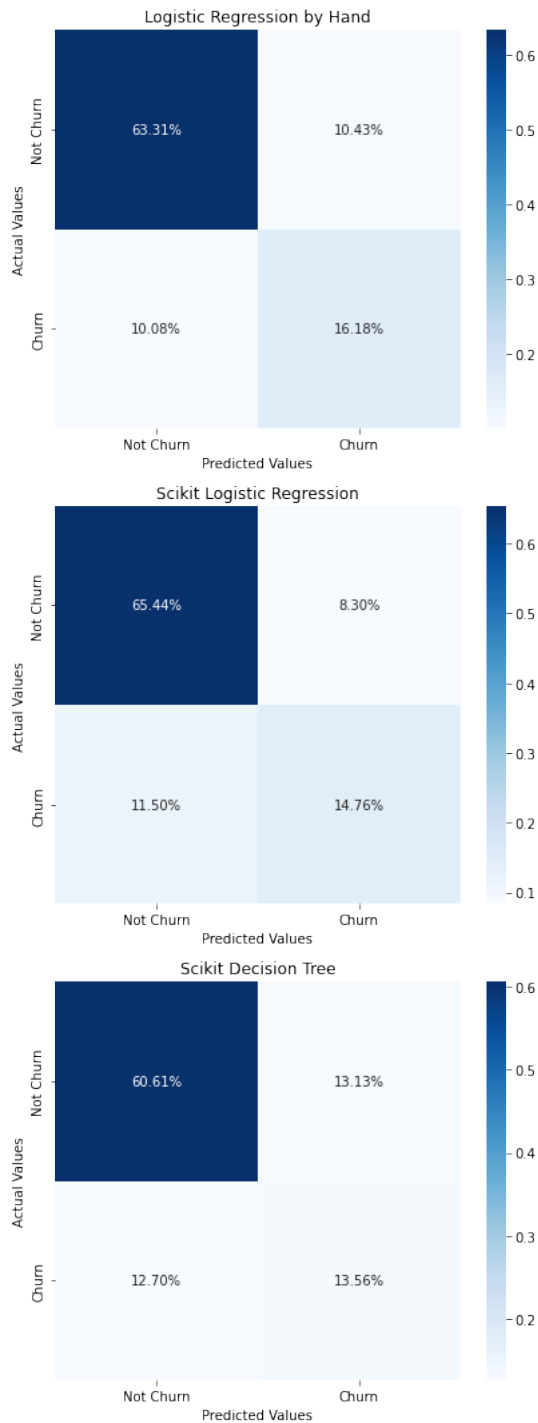


Fig. 11. Confusion matrix per model

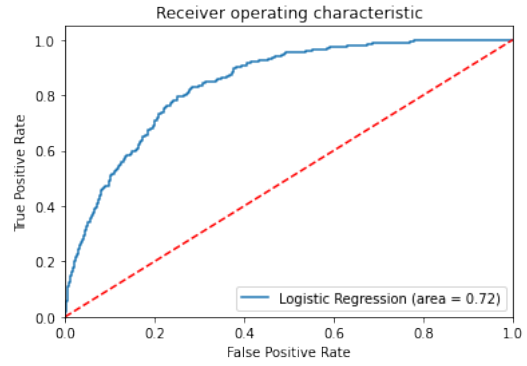


Fig. 12. Scratch Logistic Regression model ROC

## VI. DISCUSSION

While the theory was applied successfully and the scratch logistic regression model got pretty similar results to that of the SciKit implementation, all of the models could have better accuracy (with an average of 80 % at best). A standalone query script was developed additionally to obtain input data from the user to create a prediction with the scratch logistic regression model. Some tests were done with this query program to intentionally trigger Churn and Not Churn predictions. However, most of these queries resulted in False Negatives. The previous experiments imply that the models may be overfitting with the dataset. Other approaches in the state of the art which use the same dataset were investigated and it was found that these approaches resulted in approximately similar accuracy percentages. This implies that the problem could be from the dataset itself. Improvements could be made to the models by providing more data especially combining from different sources.

## REFERENCES

- [1] churn. 2011. In Merriam-Webster.com. Retrieved March 22, 2022, from <https://www.merriam-webster.com/dictionary/hacker>
- [2] Telco Customer Churn, 2018. In Kaggle.com. Retrieved March 22, 2022, from <https://www.kaggle.com/blatchar/telco-customer-churn>