Simón Mijares

Udacity Nanodegree: Machine Learning Engineer Nanodegree

Domain: Telecommunications

May 2021

Capstone Project: Churn Predictor

Project Overview

Traditionally the base client grow rate was one of the main KPI in a service business, but currently is known that it does not matter if those clients will not stay long enough to develop some fidelity to your services. In fact, acquire a new customer cost five times more than maintain an existing client[1].

For this reason, monitor your levels of churn (or attrition) is important, more in current times when the pandemic took most people out of their regular routine, struggling with income stability, heath issues among many others. Not only clients are having difficult times, but the companies are also having difficulties in different stages of its own flows. Transport, distribution, or the sole challenge of working remote to name a few examples.

The Telco's are one case, and it will be the case of study for this project. The possible variables of study are certainly multiples but all orbit around the same concept, technical variables, commercial variables, interaction variables and commitment with the services variables. These will be explained bellow.

As part of this work, I will try to obtain a model in Sagemaker capable of predict when a client will most probably churn, so the company could try to reach him and try to keep him happy.

We will try to compare random forests with a neural networks model since these are popular proved methods to predict churn based on historical data[2].

[1] C. Grönroos, A service quality model and its marketing implications, Eur. J. Mark., 18 (1984), pp. 36-44

[2] https://www.sciencedirect.com/science/article/pii/S0167923621000518

Dataset

To train the model we will be using the data set available in Kaggle at: https://www.kaggle.com/barun2104/telecom-churn.

Parameters:

In this dataset we have the following columns:

- **Churn**: 1 if customer cancelled service, 0 if not. *This will be the variable to predict.*
- **AccountWeeks**: number of weeks customer has had active account. The first weeks after a service installation could imply high call rates to customer services, with the number of week we could identify this case.
- **ContractRenewal**: 1 if customer recently renewed contract, 0 if not. If the user have a yearly contract, might be tempted to look to a different provider close to the end.
- **DataPlan**: 1 if customer has data plan, 0 if not. Usually, the pay as you go plans have a quite different behavior compared to user with a data plan.
- **DataUsage**: gigabytes of monthly data usage. The user might not be using the service or barely using it. This probably will lead to a churned user.
- **CustServCalls**: number of calls into customer service. The more the user call, the more upset it might be.
- **DayMins**: average daytime minutes per month. Like the DataUsage, this is a feature to make a profile of the user.
- **DayCalls**: average number of daytime calls. Another feature to help make a profile of the user.
- **MonthlyCharge**: average monthly bill. Also, to build a profile of the user based on what the user is been charged. The minimal plan user is quite different to the more premium one.
- **OverageFee**: largest overage fee in last 12 months. The user might need a different plan to prevent overage.

Data Análisis

Table 1: First 10 rows

n	Churn	Account Weeks	Contract Renewal	Data Plan	Data Usage	CustServ Calls	Day Mins	Day Calls	Monthly Charge	Overage Fee	Roam Mins
0	0	128	1	1	2.7	1	265.1	110	89	9.87	10
1	0	107	1	1	3.7	1	161.6	123	82	9.78	13.7
2	0	137	1	0	0	0	243.4	114	52	6.06	12.2
3	0	84	0	0	0	2	299.4	71	57	3.1	6.6
4	0	75	0	0	0	3	166.7	113	41	7.42	10.1
5	0	118	0	0	0	0	223.4	98	57	11.03	6.3
6	0	121	1	1	2.03	3	218.2	88	87.3	17.43	7.5
7	0	147	0	0	0	0	157	79	36	5.16	7.1
8	0	117	1	0	0.19	1	184.5	97	63.9	17.58	8.7
9	0	141	0	1	3.02	0	258.6	84	93.2	11.1	11.2

In the data we can find a total of 3333 samples. Churns: 483 (14 %) and Not Churns: 2850.

Table 2: Features Statistics

Feature	Churn	Account Weeks	Contract Renewal	Data Plan	Data Usage	CustServ Calls	Day Mins	Day Calls	Monthly Charge	Overage Fee	Roam Mins
count	3,333	3,333	3,333	3,333	3,333	3,333	3,333	3,333	3,333	3,333	3,333
mean	0.14	101.06	0.9	0.28	0.82	1.56	179.78	100.44	56.31	10.05	10.24
std	0.35	39.82	0.3	0.45	1.27	1.32	54.47	20.07	16.43	2.54	2.79
min	0	1	0	0	0	0	0	0	14	0	0
25%	0	74	1	0	0	1	143.7	87	45	8.33	8.5
50%	0	101	1	0	0	1	179.4	101	53.5	10.07	10.3
75%	0	127	1	1	1.78	2	216.4	114	66.2	11.77	12.1
max	1	243	1	1	5.4	9	350.8	165	111.3	18.19	20

Variable Graphic Distributions (given by python's *profile_report*)

Churn
Categorical

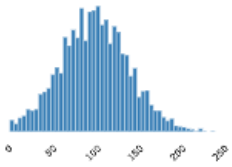
Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%

AccountWeeks

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	212
Distinct (%)	6.4%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	101.0648065

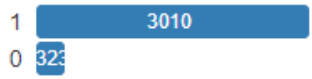
Minimum	1
Maximum	243
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



ContractRene…

Categorical

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%



DataPlan

Categorical

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	2
Distinct (%)	0.1%
Missing	0
Missing (%)	0.0%
Memory size	26.2 KiB

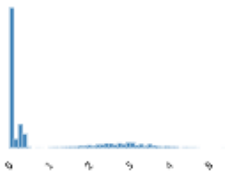
DataUsage

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
ZEROS

Distinct	174
Distinct (%)	5.2%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	0.8164746475

Minimum	0
Maximum	5.4
Zeros	1813
Zeros (%)	54.4%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



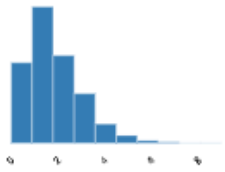
CustServCalls

Real number (ℝ≥0)

ZEROS

Distinct	10
Distinct (%)	0.3%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	1.562856286

Minimum	0
Maximum	9
Zeros	697
Zeros (%)	20.9%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



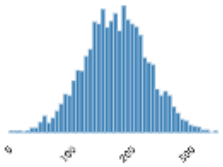
DayMins

Real number (ℝ≥0)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	1667
Distinct (%)	50.0%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	179.7750975

Minimum	0
Maximum	350.8
Zeros	2
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB

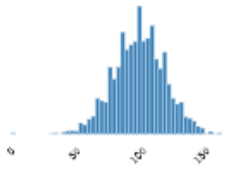


DayCalls

Real number (ℝ≥0)

Distinct	119
Distinct (%)	3.6%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	100.4356436

Minimum	0
Maximum	165
Zeros	2
Zeros (%)	0.1%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



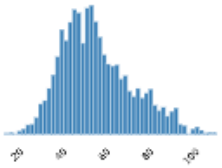
MonthlyCharge

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION
HIGH CORRELATION

Distinct	627
Distinct (%)	18.8%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	56.30516052

Minimum	14
Maximum	111.3
Zeros	0
Zeros (%)	0.0%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB

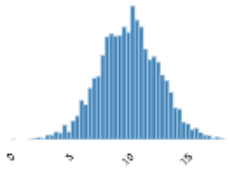


OverageFee

Real number ($\mathbb{R}_{\geq 0}$)

Distinct	1024
Distinct (%)	30.7%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	10.05148815

Minimum	0
Maximum	18.19
Zeros	1
Zeros (%)	< 0.1%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



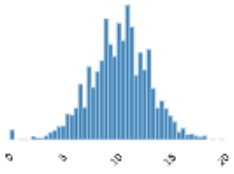
RoamMins

Real number ($\mathbb{R}_{\geq 0}$)

HIGH CORRELATION

Distinct	162
Distinct (%)	4.9%
Missing	0
Missing (%)	0.0%
Infinite	0
Infinite (%)	0.0%
Mean	10.23729373

Minimum	0
Maximum	20
Zeros	18
Zeros (%)	0.5%
Negative	0
Negative (%)	0.0%
Memory size	26.2 KiB



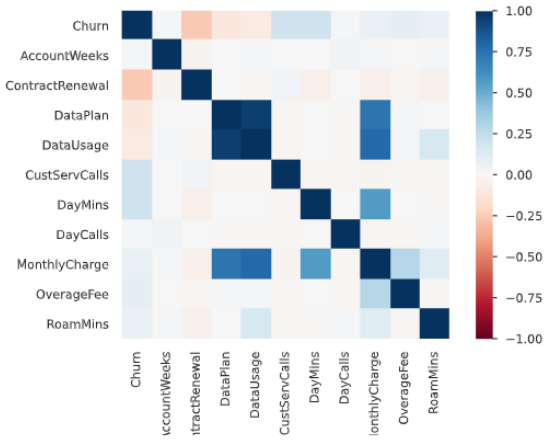


Figure 1. Correlation Matrix

In the matrix of the Figure 1 we can appreciate a high correlation between *DataPlan* and *DataUsage*. This is a good indicator that one could be discarded, since DataPlan is binary and offer less information, we probably could discard this feature for de analysis.

Benchmark Model

Results from *Talian Linda Chen* (https://www.kaggle.com/taliac/different-resampling-methods-for-trees)

Final	Accuracy	Precision	Recall
Random Forest downscaled	0.9041	0.6076	0.9796
Random Forest w/Smore	0.8921	0.6048	0.7653
Gradient Boost w/Smore	0.8906	0.6016	0.7551
Gradient Boost without SM	0.9370	0.8590	0.6837
Gradient Boost with 0.38 threshold	0.8770	0.5571	0.7959

Solution Statement

To solve the problem ML models were trained and tested. Bias due to statistical imbalance need to be taken in consideration, in this case recall will be considered without leaving out the precision.

The development is based on the Fraud Detection case of use of the course since the application is remarkably similar on spirit. In the mentioned example the target was detecting fraudulent operations avoiding as much as possible detecting false negatives. In our case we need to detect which clients pretend leaving the company, avoiding false positives as well. In the figure bellow we could see a representation on how disproportionate predictions could be due to imbalance in the classes. Causing bias in your model.

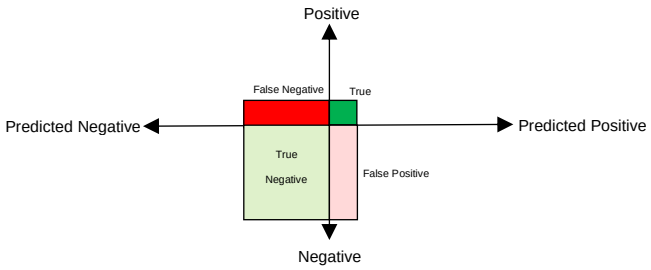


Figure 2. Imbalanced Training data and biased model

$$Recall = \frac{\Sigma \text{ True positive}}{\Sigma \text{ positive}}$$

$$Precision = \frac{\Sigma \text{ True positive}}{\Sigma \text{ predicted positive}}$$

Methodology

Data Preprocessing

In the *data exploration* we could evidence that there is no missing values or nulls included, so our next phase will be the correlation analysis. As showed below.

```
# Create correlation matrix for just Features to determine different models to test
corr_matrix = churn_df.corr().abs().round(2)

# display shows all of a dataframe
display(corr_matrix)
```

Table 3 Crosscorrelation table

	Churn	Account Weeks	Contract Renewal	Data Plan	DataUsage	Cust Serv Calls	Day Mins	Day Calls	Monthly Charge	Overage Fee	Account Weeks
Churn	1.00	0.02	0.26	0.10	0.09	0.21	0.21	0.02	0.07	0.09	0.07
Account Weeks	0.02	1.00	0.02	0.00	0.01	0.00	0.01	0.04	0.01	0.01	0.01
Contract Renewal	0.26	0.02	1.00	0.01	0.02	0.02	0.05	0.00	0.05	0.02	0.05
DataPlan	0.10	0.00	0.01	1.00	0.95	0.02	0.00	0.01	0.74	0.02	0.00
DataUsage	0.09	0.01	0.02	0.95	1.00	0.02	0.00	0.01	0.78	0.02	0.16
Cust Serv Calls	0.21	0.00	0.02	0.02	0.02	1.00	0.01	0.02	0.03	0.01	0.01
Day Mins	0.21	0.01	0.05	0.00	0.00	0.01	1.00	0.01	0.57	0.01	0.01

Day Calls	0.02	0.04	0.00	0.01	0.01	0.02	0.01	1.00	0.01	0.02	0.02
Monthly Charge	0.07	0.01	0.05	0.74	0.78	0.03	0.57	0.01	1.00	0.28	0.12
Overage Fee	0.09	0.01	0.02	0.02	0.02	0.01	0.01	0.02	0.28	1.00	0.01
Roam Mins	0.07	0.01	0.05	0.00	0.16	0.01	0.01	0.02	0.12	0.01	1.00

As it was stated before, the DataPlan and DataUsage are deeply correlated, so we will be dropping the ‘DataPlan’ column since is binary and has less information than DataUsage.

```
churn_decorr_df=churn_df.drop('DataPlan', axis=1)
```

Now we will form our data and label for training’.

```
x=churn_decorr_df.drop('Churn', axis=1)
x.head(10)
```

	AccountWeeks	ContractRenewal	DataUsage	CustServCalls	DayMins	DayCalls	MonthlyCharge	OverageFee	RoamMins
0	128	1	2.70	1	265.10	110	89.00	9.87	10.00
1	107	1	3.70	1	161.60	123	82.00	9.78	13.70
2	137	1	0.00	0	243.40	114	52.00	6.06	12.20
3	84	0	0.00	2	299.40	71	57.00	3.10	6.60
4	75	0	0.00	3	166.70	113	41.00	7.42	10.10
5	118	0	0.00	0	223.40	98	57.00	11.03	6.30
6	121	1	2.03	3	218.20	88	87.30	17.43	7.50
7	147	0	0.00	0	157.00	79	36.00	5.16	7.10
8	117	1	0.19	1	184.50	97	63.90	17.58	8.70
9	141	0	3.02	0	258.60	84	93.20	11.10	11.20

```
y=churn_decorr_df[['Churn']]
y.head(10)
```

	Churn
0	0
1	0
2	0
3	0
4	0
5	0
6	0
7	0
8	0
9	0

Initially we split the data in 60% samples for training, 20% for validation and 20% test. But probably due to the small size of the dataset this impacted the result significatively. To circumvent this issue, we instead split the data in 80% for training and 20% test/val.

Once the data was separated in label and data for training and test/val, the next step is to export them as csv and upload it to a S3 to be available to the classifier for training.

Implementation

Our main target is to train a predictor with the XGBoost algorithm, since is an efficient and powerful option for our case[3], but we are going to compare with a Linear Learner as follow:

a. Plain XGBoost

```
xgb = sagemaker.estimator.Estimator(container,         # The location of the container we wish to use
                                    role,                                    # What is our current IAM Role
                                    instance_count=1,                        # How many compute instances
                                    instance_type='ml.m4.xlarge',            # What kind of compute instances
                                    output_path='s3://{}/{}/output'.format(session.default_bucket(), prefix),
                                    sagemaker_session=session)


xgb.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        early_stopping_rounds=10,
                        num_round=500)
```

b. XGBoost with Scale Post Weight

```
xgb_tunned = sagemaker.estimator.Estimator(container,   # The location of the container we wish to use
                                    role,                                    # What is our current IAM Role
                                    instance_count=1,                        # How many compute instances
                                    instance_type='ml.m4.xlarge',            # What kind of compute instances
                                    output_path='s3://{}/{}/output'.format(session.default_bucket(), prefix),
                                    sagemaker_session=session)

# https://xgboost.readthedocs.io/en/latest/parameter.html#learning-task-parameters
# https://machinelearningmastery.com/xgboost-for-imbalanced-classification/
xgb_tunned.set_hyperparameters(max_depth=5,
                        eta=0.2,
                        gamma=4,
                        min_child_weight=6,
                        subsample=0.8,
                        silent=0,
                        objective='binary:logistic',
                        eval_metric='auc',
                        scale_pos_weight=6,
                        early_stopping_rounds=10,
                        num_round=500)
```

c. Linear Learner

```
# specify an output path
prefix = 'linearlearner_tunned'
output_path = 's3://{}/{}'.format(bucket, prefix)

# instantiate a LinearLearner
linear_recall = LinearLearner(role=role,
                              instance_count=1,
                              instance_type='ml.c4.xlarge',
                              predictor_type='binary_classifier',
                              output_path=output_path,
                              sagemaker_session=session,
                              epochs=15)
```

[3] https://ieeexplore.ieee.org/abstract/document/7937698

d. Linear Learner Balanced

```
# instantiate a LinearLearner
# include params for tuning for higher recall
# *and* account for class imbalance in training data
linear_recall_balanced = LinearLearner(role=role,
                            instance_count=1,
                            instance_type='ml.c4.xlarge',
                            predictor_type='binary_classifier',
                            output_path=output_path,
                            sagemaker_session=session,
                            epochs=15,
                            binary_classifier_model_selection_criteria='precision_at_target_recall', # target recall
                            target_recall=0.9, # 90% recall
                            positive_example_weight_mult = 'balanced')
```

Results

What we obtained can be summarize in the table below:

Table 4 Results by model

	Recall	**Precision**	**Accuracy**
Plain XGBoost	0.69	0.81	0.93
XGBoost with Scale Post Weight	0.79	0.60	0.89
Linear Learner	0.144	0.538	0.858
Linear Learner Balanced	0.907	0.227	0.538

Our best model for Accuracy was the “Plain XGBoost”, for Recall the “Linear Learner Balanced” and the best mid-point should be the “XGBoost with Scale Post Weight”.

The tunning process improved in both cases the recall as expected, maybe just too much in the case of the Linear Learner model.

In comparison with our benchmarks, we were able to reach similar results in accuracy but a little lower in recall.

Reflection

During the development, my first challenge was the low values obtained from the predictors and how to solve it. After some investigations, I realize that the size of the train set was impacting the accuracy and recall results, so the dataset for training was incremented and the results changed radically.

Improvement

The first improvement we could probably make is a bigger dataset, this will allow us more concise results.

If we plan to use the model for production environment, a deployed model could be integrated with the analytic flow and even a simple web interface to load a reduced set, in case the commercial department need to obtain a list of clients about to churn to contact then and offer special deals to keep then.

Another improvement possible is to change the model in a time series, this will might allow to follow the life cycle on the client and maybe change the binary classifier for a multiclass one. This way the commercial department can prioritize those close to the end of the cycle or avoid those beyond rescue.

If we need to squeeze even better results, Sagemaker allow us to finetuning al the hyperparameters of every model to get the best of everyone.