Machine Learning Engineer Nanodegree
Capstone Project - Prediciting the Churn for a newspaper media

Deividi Pansera

Definition

Project Overwiew

Businesses in the consumer market and, in fact, in all enterprise sectors have to deal with churn. Very often, churn is excessive and influences policy decisions. The traditional solution that can be found is to predict high-propensity churners and address their needs via something like a concierge service, marketing campaigns, etc. It can vary from industry to industry.

Well, the common factor is that businesses need to minimize these special customer retention efforts. Therefore, given this perspective, a natural methodology would be to score a churn probability to every customer and to address the top N ones. The top customers might be the most profitable ones or not. It depends of a lot of variables.

The domain background of this project is churn prediction for a Brazilian newspaper media called Gazeta do Povo (GP). The main source of business income for GP is through payed subscription at its website. Therefore, the churn problem is crucial for its business.

For this project, we will use a dataset provided from a Brazilian Newspaper Media, *Gazeta do Povo*, which we shall call GP. GP is an old Brazilian Newspaper media, based in South Brazil. Recently, within the technological revolution happening in the world, GP decided to "leave behind what does not lead you forward." That is, they have stopped to sell their product in a printed version and decided to sell it a digital version of it. This is a way, at least for them, to be adaptive and competitive in the new "Technological Era." This means that now their product is completely online and one of the most important way for them to get money is from the payed subscriptions to their website. Given this scenario, once someone decided to pay a subscription, it would be nice to not let this user to cancel the subscription. In this project we will use supervised learning algorithms to predict the Churn, i.e., to predict whether or not a subscriber of GP will cease to pay a subscription.

In what follows, it is included some papers on the specific problem of the Churn analysis:

- Predictive churn analysis with machine learning methods;
- How to Leverage AI to Predict (and Prevent) Customer Churn;
- The Challenges of Building a Predictive Churn Model;
- Churn Prediction- Machine Learning Starter;

- Introduction to Churn Prediction in Python

Problem Statement

The main source of business income for GP is through payed subscription, as it was said before. So, for this project, the purpose is to build a Machine Learning model to predict the churn based on a handful of algorithms that we will test (e.g. Random Forest, AdaBoostingClassifier, Gradient Boosting). We shall conduct experiments and with the results we will demonstrate the performance of the models by using statistical metrics like F_β-score, precision, recall, etc. With the higher scoring of these metrics, we shall be able to judge the success of these models in predicting the churn.

Metrics

GP is particularly interested in predicting who will practiced the Churn accurately. It would seem that using *accuracy* as a metric for evaluating a particular model's performance would be appropriate. Additionally, identifying someone who will not practice the Churn as someone who will would be detrimental to GP, since they are looking to find individuals who shall leave. Therefore, a model's ability to precisely predict those that will practice the Churn is *more important* than the model's ability to **recall** those individuals. We can use **F-beta score** as a metric that considers both precision and recall:

$$F_\beta = (1 + \beta^2) \cdot \frac{precision \cdot recall}{(\beta^2 \cdot precision) + recall}$$

In particular, when $\beta = 0.5$, more emphasis is placed on precision. This is called the $F_{0.5}$ score (or F-score for simplicity).

Note: Recap of accuracy, precision, recall

Accuracy measures how often the classifier makes the correct prediction. Its the ratio of the number of correct predictions to the total number of predictions (the number of test data points).

Precision tells us what proportion of messages we classified as spam, actually were spam. It is a ratio of true positives (words classified as spam, and which are actually spam) to all positives (all words classified as spam, irrespective of whether that was the correct classification), in other words it is the ratio of

$$\frac{\textbf{True Positives}}{(\textbf{True Positives} + \textbf{False Positives})}$$

Recall(sensitivity) tells us what proportion of messages that actually were spam were classified by us as spam. It is a ratio of true positives (words classified as spam, and which are actually spam) to all the words that were actually spam, in other words it is the ratio of

$$\frac{\textbf{True Positives}}{(\textbf{True Positives} + \textbf{False Negatives})}$$

For classification problems that are skewed in their classification distributions like in our case, for example if we had a 100 text messages and only 2 were spam and the rest 98 weren't, accuracy by itself is not a very good metric. We could classify 90 messages as not spam (including the 2 that were spam but we classify them as not spam, hence they would be false negatives) and 10 as spam (all 10 false positives) and still get a reasonably good accuracy score. For such cases, precision and recall come in very handy. These two metrics can be combined to get the F1 score, which is weighted average (harmonic mean) of the precision and recall scores. This score can range from 0 to 1, with 1 being the best possible F1 score (we take the harmonic mean as we are dealing with ratios).

Analysis

Data Exploration

The datasets to be used in the project were provided by GP and contains browsing histories of users. Information such as how the dataset was obtained, and the characteristics of the dataset should be included as well. Basically, there will be four datasets. Two of them containing data for the month of June and two of them for the month of July. For the both months, there will be one dataset consisting of browsing history of people who did not practice the churn and the other one consisting of browsing history of people who did practice the churn. The features contained in all datasets are listed and explained bellow:

1. *Subscription_id*: An "id" to identify an user at the database;

2. *type*: "T" for titular account and "D" for dependent account member;

3. *Recency*: A variable that varies from 0 to 30. It measures how recent a user have visited the website, where 0 means that he never had visited in the last 30 days and 30 that he came yesterday;

4. *last_access_date*: Last day of user's access at the website;
5. *lst_date*: a string containing all the days that the user have accessed the website;
6. *subscription_date*: the precise day that the user have subscribed;
7. *freq*: How many days the user have visited the website;
8. *qt_page_view*: quantity of page views a user has;
9. *qt_page_view_week*: quantity of page views during the weeks;
10. *qt_page_view_weekend*: quantity of page views during the weekends;
11. *qt_page_view_mornig*: quantity of page views during the mornings;
12. *qt_page_view_afternoon*: quantity of page views during the afternoons;
13. *qt_page_view_nigth*: quantity of page views during the nights;
14. *qt_art_read*: quantity of articles that were read by a user;
15. *qt_source_sm_cpc_facebook*: how many times a user came to the website via a paid campaign from facebook;
16. *qt_source_sm_cpc_others*: how many times a user came to the website via a paid campaign from others social medias;
17. *qt_source_sm_cpc_others*: how many times a user came to the website via a paid campaign from others social medias;
18. *qt_source_sm_facebook*: how many times a user came to the website via facebook (no paid campaign);
19. *qt_source_sm_facebook*: how many times a user came to the website via facebook (no paid campaign);
20. *qt_source_sm_other*: how many times a user came to the website via other social media (no paid campaign);
21. *qt_source_nedeal*: how many times a user came to the website via actions from a outsourced;
22. *qt_source_email*: how many times a user came to the website via other social media (no paid campaign);

23. *qt_source_others*: how many times a user came to the website via other social media (no paid campaign);

24. *qt_gazeta_capa*: how many times a user accessed the website frontpage;

25. *qt_gazeta_esportes*: how many times a user accessed the website section "esportes";

26. *qt_gazeta_politica*: how many times a user accessed the website section "politica";

27. *qt_gazeta_economia*: how many times a user accessed the website section "esportes";

28. *qt_gazeta_curitiba*: how many times a user accessed the website section "curitiba";

29. *qt_agronegocio*: how many times a user accessed the website section "agronegocio";

30. *qt_haus*: how many times a user accessed the website section "haus";

31. *qt_bom_gourmet*: how many times a user accessed the website section "bom_gourmet";

32. *qt_viver_bem*: how many times a user accessed the website section "viver_bem"';

33. *qt_guia*: how many times a user accessed the website section "guia";

34. *qt_viver_bem*: how many times a user accessed the website section "viver_bem"';

35. *qt_access_others*: how many times a user accessed other pages of the website;

36. *qt_comments*: how many times a user have made some comment at the website comment section;

37. *qt_explicar*: how many times a user have read some article with the tag "explicar";

38. *qt_alegre*: how many times a user have read some article with the tag "alegre";

39. *qt_provocar*: how many times a user have read some article with the tag “provocar”;

40. *qt_triste*: how many times a user have read some article with the tag “triste”;

41. *qt_inspirar*: how many times a user have read some article with the tag “inspirar”;

42. *qt_moderno*: how many times a user have read some article with the tag “moderno”;

43. *qt_facilitar*: how many times a user have read some article with the tag “facilitar”;

44. *qt_surpreendente*: how many times a user have read some article with the tag “surpreendente’;

45. *qt_informar*: how many times a user have read some article with the tag “informar”;

46. *qt_hardnews*: how many times a user have read some article with the tag “hardnews”;

47. *qt_softnews*: how many times a user have read some article with the tag “softnews”;

48. *qt_appGazeta*: how many times a user have used the app section “gazeta” to access the website;

49. *qt_appGuia*: how many times a user have used the app section “guia” to access the website;

50. *qt_mobile*: how many times a user have used a mobile to access the website;

51. *qt_other_devices*: how many times a user have used other devices rather than mobile to access the website;

52. *qt_login*: how many times a user have logged in;

53. *browser*: the user most common used browser to access the website;

54. *has_club*: if the user has club, which is another product of the newspaper media;

55. *has_print*: if the user has a printed version of the newspaper;

56. *Churn*: the output variable; if the user practiced the Churn or not.

The dataset contains 62581 data points, where 1371 data points for the class of people who did practice the churn and 61210 who didn't.

The data has many datapoints with no browsing history and has to be treated. Also, it has some missing values as well. There is just one non-numerical feature. It is the feature *browser*, which is categorical.

A sample of one of the datasets is given bellow.

	subscription_id	type	Recency	last_access_date	lst_date	subscription_date	freq	qt_page_view	qt_page_view_week	qt_page_view_weekend	qt_page_viev
0	1000465	T	27.0	2018-05-30	2018-05-03,2018-05-04,2018-05-05,2018-05-06,20...	1994-11-26	26	225	169	56	
2	1001140	T	30.0	2018-06-06	2018-05-07,2018-05-07,2018-05-08,2018-05-12,20...	1994-12-25	21	109	51	58	
9	1005513	T	10.0	2018-06-04	2018-05-25,2018-06-04,2018-06-24	1995-04-25	1	14	14	0	
15	1014499	T	21.0	2018-06-06	2018-05-16,2018-06-06,2018-06-15	1995-01-25	1	10	10	0	
27	1023969	T	26.0	2018-05-29	2018-05-03,2018-05-29,2018-06-02	2004-01-12	1	3	3	0	

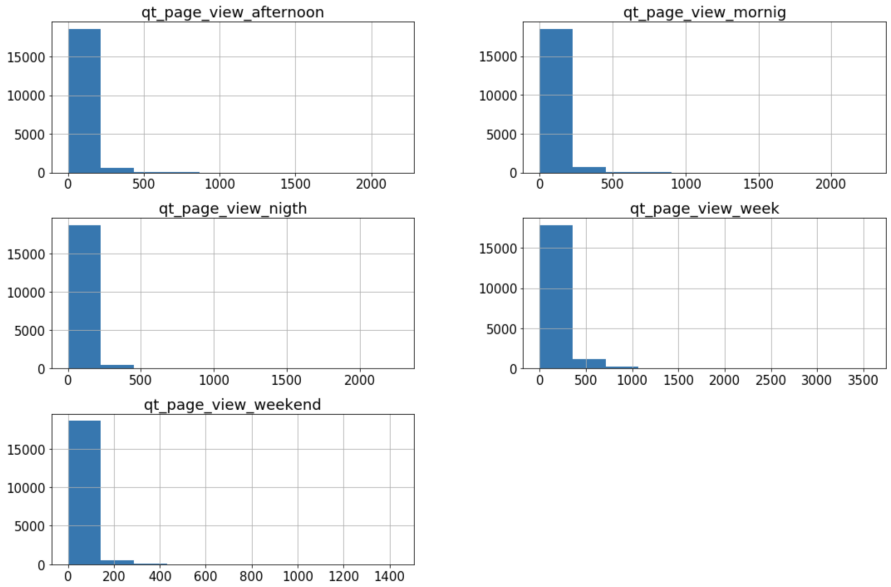
Also, in what follows there are the statistics of some variables, which gives us a glimpse of how the data is distributed and, hence, how it needs some treatment before we apply any Machine Learning Algorithm.

	Recency	freq	qt_page_view	qt_page_view_week	qt_page_view_weekend	qt_page_view_mornig	qt_page_view_afternoon
count	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000
mean	25.015304	13.142924	138.567130	111.581345	26.985785	49.757937	44.944283
std	7.335434	9.486268	227.473971	186.818567	56.043122	90.916882	87.862962
min	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	0.000000
25%	23.000000	4.000000	12.000000	9.000000	0.000000	3.000000	2.000000
50%	29.000000	12.000000	53.000000	42.000000	6.000000	15.000000	13.000000
75%	30.000000	21.000000	176.000000	141.000000	30.000000	60.000000	51.000000
max	30.000000	30.000000	3788.000000	3566.000000	1436.000000	2261.000000	2174.000000

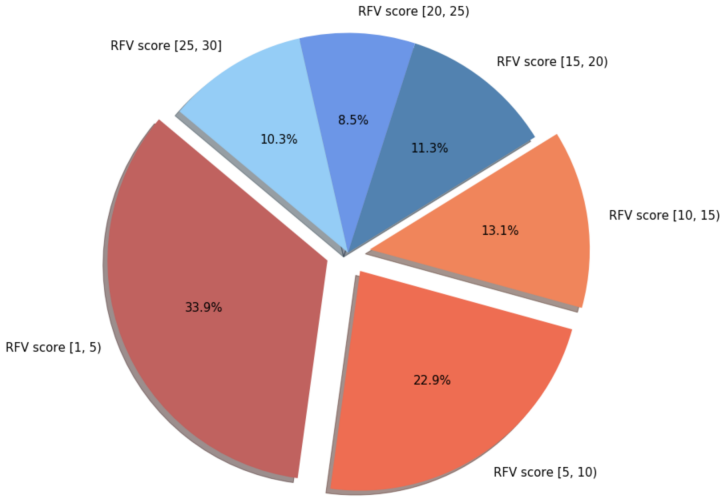
qt_moderno	qt_facilitar	qt_surpreendente	qt_informar	qt_hardnews	qt_softnews	qt_appGazeta	qt_appGuia	qt_mobile	qt_other_devices
19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000	19276.000000
5.057585	3.374870	5.319309	25.977329	19.400913	5.920782	29.883793	0.003113	30.823822	77.856402
9.723402	6.141924	8.948348	44.326666	35.028546	10.779348	103.529545	0.224309	84.539098	186.478727
0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.000000	0.000000	0.000000	2.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
2.000000	1.000000	2.000000	10.000000	7.000000	2.000000	0.000000	0.000000	1.000000	8.000000
6.000000	4.000000	7.000000	33.000000	24.000000	7.000000	6.000000	0.000000	18.000000	74.000000
345.000000	151.000000	398.000000	1169.000000	907.000000	304.000000	3174.000000	25.000000	1943.000000	3788.000000

Exploratory Visualization

In what follows, we plot how skewed is some of the variables of our dataset. It shows, among other things, how the data should be treated before we could apply any machine learning algorithm.



Another important visualization is the distribution of the data in terms of our benchmark model (see later), which divide the data in six groups and these division will be useful to balance the classes to be predicted later on.



Algorithms and Techniques

We shall use some classification algorithms (Gradient Boosting Classifier, Ada Boost Cassifier, Random Forest Classifier etc.) to solve this problem.

The first big problem that we have to deal with is the imbalaced data problem. We have 18718 people who did not practice the Churn and just 558 who did. The first thing we will do, though, is to use the *Local Outlier Factor* (LOF) algorithm to detect anomalies in the dataset of people who did not practice the churn. For more information of this algorithm, we recommend the follwing links:

1. Lof outlier detection;
2. LOF - Wikipedia;
3. http://www.dbs.ifi.lmu.de/Publikationen/Papers/LOF.pdf.

After that, we shall use the k-means algorithm (see this and this) to extract samples from the data in order to balance the training set. The number of clusters, k, is given by the RFV score, our benchmark (see the next subsection). It is a suggestion that we have 6 important intervals of RFV score ($[1, 5)$, $[5, 10)$, $[10, 15)$, $[15, 20)$, $[20, 25)$, $[25, 30]$). Then we take the k for k-means to be equal to 6. After that, we use three supervised learning algorithms which are good to deal with this kind of problem, namely, Ada Boost Classifier, Gradient Boosting Classifier and Random Forest Classifier. We compare the performance of the three algorithms and choose the

one that had a better performance. After that, we use gridsearch to tune the model and use joblib to save the model.

In what follows, we briefly describe the math behind two of the three algorithms used in this project to build the model. If the reader wants to learn more about it, we recommend the excellent book An Introduction to Statiscal Learning (freely available).

- **Adaboost Classifier:**

  Basically, Adaboost classifier combines weak classifier algorithm to form a strong classifier. A single algorithm may classify the objects poorly. But if we combine multiple classifiers with selection of training set at every iteration and assigning right amount of weight in final voting, we can have a much better result for the classifier. Each weak classifier is trained using a random subset of the training set. Mathematically, we have the following function

  $$H(x) = sign\left(\sum_{t=1}^{T} \alpha_t h_t(x)\right),$$

  where $h_t(x)$ is the output of weak classifier t for input x, α_t is the weight assigned to the classifier and it is calculated as follows

  $$\alpha_t = 0.5 \ln\left(\frac{(1-E)}{E}\right),$$

  where E is the error rate. After the weak classifier is trained, we update the weight of each training example with the following formula

  $$D_{t+1}(i) = \frac{D_t(i)\exp(-\alpha_t y_i h_t(x_i))}{Z_t},$$

  where D_t is the weight at a previous level (so it is an iterative function), Z_t is the sum of all weights (so, it is a normalization of the weights what we are doing when we divide by Z_t).

- **Gradient Boosting Classifier:**

  In what follows, we describe the pseudo algorithm of Gradient Boosting. We start with a training set $\{(x_i, y_i)\}_{i=1}^{n},$, a differentiable loss function $L(y, F(x))$, and a number of iterations M. Then, the algorithms goes as it follows:

– Initialize model with a constant value:

$$F_0(x) = \arg\min_{\gamma} \sum_{i=1}^{n} L(y_i, \gamma)$$

– For $m = 1$ to M:

* Compute the so-called pseudo-residuals:

$$r_{im} = -\left[\frac{\partial L(y_i, F(x_i))}{\partial F(x_i)}\right]_{F(x)=F_{m-1}(x)} \quad \text{for } i = 1, \ldots, n.$$

* Fit a base learner (e.g. tree) $h_m(x)$ to pseudo-residuals, i.e. train it using the training set $\{(x_i, r_{im})\}_{i=1}^{n}$.
* Compute the multiplier γ_m by solving the following one-dimensional optimization problem:

$$\gamma_m = \arg\min_{\gamma} \sum_{i=1}^{n} L\left(y_i, F_{m-1}(x_i) + \gamma h_m(x_i)\right).$$

* Update the model:

$$F_m(x) = F_{m-1}(x) + \gamma_m h_m(x).$$

– Output $F_M(x)$.

To understand the functions used and a little bit more about the algorithm, we recommend a gentle introduction to Gradient Boosting as well.

- **Random Forest:**

For Random Forest, we describe it in layman's terms as it follows: Suppose that we want to predict if a patient Z entering an ER is sick at a high risk or not. In order to decide that, we let the nurse A_1 to decide whether Z is sick at a high risk or not. A_1, then, first consider whether Z is over 50 years old or not (the first split). Given A_1 starts to consider if Z blood pressure is over 150 (the second split on a branch). Given no, A_1 looks to other vitals. After a finite number of consideration and splits, A_1 decides whether Z is sick at a high risk or not. A_1, in Machine Learning terms, is a decision tree. But her predictive power, alone, is not that powerfull. We need more opinion and different question asked in order to predict with high accuracy if

Z is sick at a high risk or not. In Machine Learning terms, we need more decision trees and to compare their decisions. So we call nurses $A_1, \ldots, A_n$ (n-nurses) to do this classification job. After all of them have classified (with different questions most probably) the sickness of Z, we compare their predictions and the more voted one is our choice.

Random Forest, roughly speaking, does precisely that that is described in the paragraph above. It uses a lot of decision trees (an ensemble), where each tree is different from the others. When we have a patient, we take the majority vote of the ensemble to get a final result. To guarantee that the trees are different from each other, we used random samples of the given data to train them, i.e., each tree only "sees" a part of the training data, in other words, we use a subset of the features for each tree. Hence, if each tree comes with a different answer, we will get better results.

Benchmark model

The Benchmark model for this problem will be the classical RFM analysis but with an adaptation for the needs of GP. Basically, we have three variables, *Recency*, *Frequency* and *Volume*. *Recency* means how recent a user has visited the website (it varies from 0 to 30, where 0 means that the user did not visit the website in thirty days and any other number x means that the user has visited the website $31 - x$ days ago); *Frequency* means, during the thirty days, how many days the user has visited the website; *Volume* means how many articles and page views, according with an specific formula, the user has consumed at the website.

The *Volume* is calculated in the following way:

$$Vol = \frac{(pv - ra)}{2} + ra,$$

where pv is page views and ra is read articles. After 10 seconds of a hit in some webpage of the website that is not a home, the hit counts as a read article. And, since every read article is also a page view, we subtract the read articles from the page views in order to calculate the Vol variable. Then, after that, and with the approval of a committee of GP, it was formulated the following formula for the volume variable:

1. $Volume = 30$ if $Vol \geq 88$;
2. $Volume = \left\lceil \frac{Vol}{3} \right\rceil$ if $Vol < 88$.

Hence, the variables *Recency*, *Frequency* and *Volume* vary from 0 to 30. Then, with them, we calculate what we called the *RFV score* by taking the **Geometric mean** of these variables, i.e.,

$$\sqrt[3]{R.F.V},$$

where R stands for *Recency*, F for *Frequency* and V for *Volume*.

The company, then, have decided that a critical user is the one who has an *RFV score* under 14. For the base of June, 70% of the people who committed the churn had their RFV score under 14. We shall compare the predictive power of our method with the RFV model and prove that our model have a more satisfactory predictive power.

Methodology

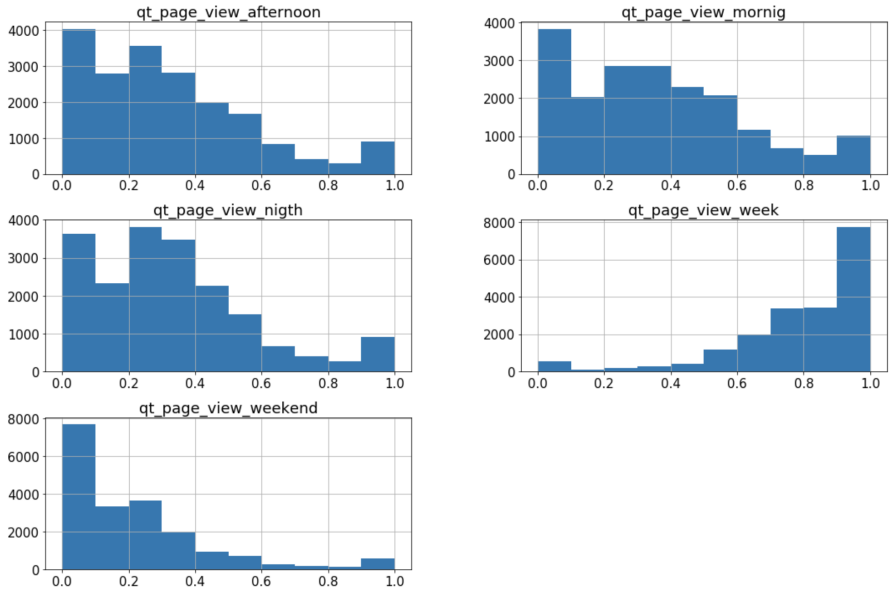
Data Preprocessing

From the the numerical data, we first check how many informations we can get from the columns containing them. So, we check the percentage of non-zero values on those columns. Some features with too many zeros do not add too many significant value to our analysis. The features that start with *qt_source_* mean how many times the user came to our website from that source. Since mostly the users come to the website via some other source than the ones listed, we decided to drop the following columns:

1. qt_source_sm_cpc_facebook, qt_source_sm_cpc_others, qt_appGuia,
2. qt_source_sm_facebook, qt_source_sm_other, qt_source_nedeal,
3. qt_source_email, qt_source_others.

After that, from the feature lst_date, which contains a string with all the days that a user visited the website, we calculate the maximum interval, in days, of non access of a user. Also, we extract another information from the collumns last_access_date and subscription_date. We would like to know how many days a user had to access the website. For that, if his subscription year is before 2018, we start to count since the beggining of 2018, because before this time the newspaper was not pretty much just online; if his subscription year is 2018, we count from his subscription date until his last access date.

At last, we correct the distribution of the numerical features. We already have shown a picture of how skewed some variables were. Here we show the same variables after some adjustments:



Also, we use log transformation to correct the distribution and we calculate the RFV score of the users and use the variable in our model. Besides that, we also normalize the data and apply the get_dummies function for the variable *browser*.

Implementation

In what follows, I describe the workflow of the implementations that were made:

- First, we create the RFV model;
- The we use the LOF algorithm to identify the Local Outliers;
- After that we use the k-means algorithm to extract samples to balance the data;
- We use train_test_split from scikitlearn to divide our dataset in training and test sets;
- We define a function, train_predict, to evaluate our model;
- We build models using Gradient Boosting Classifier, Ada Boost Classifier, and Random Forest Classifier;

- We select the best model (Gradient Boosting Classifier);

Also, it is worth to mention that the choice of the k for the k-means algorithm was a little bit complicated. But fortunately, the RFV model provided six class of people according to their level of engagement with the website.

Here we bring some figures of the training process which show why we chose the Gradient Boosting Classifier. The first picture is, precisely, about the Gradient Boosting Classifier:

`GradientBoostingClassifier`

	1%	10%	100%
acc_test	0.623418	0.750000	0.879747
acc_train	0.606667	0.860000	0.963333
f_test	0.450820	0.608466	0.802583
f_train	0.462810	0.824295	0.950096
prec_test	0.433071	0.666667	0.790909
prec_train	0.448000	0.853933	0.951923
pred_time	0.001279	0.002793	0.001909
recall_test	0.539216	0.450980	0.852941
recall_train	0.533333	0.723810	0.942857
train_time	0.025248	0.092163	0.327612

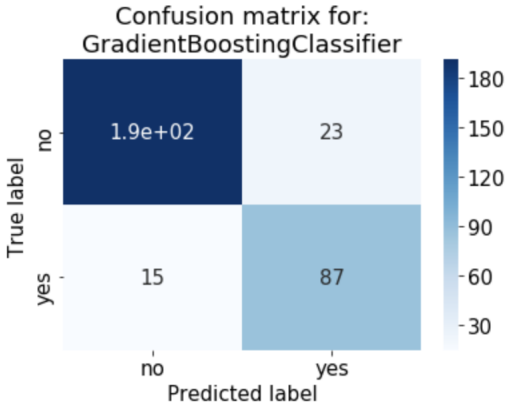
`AdaBoostClassifier`

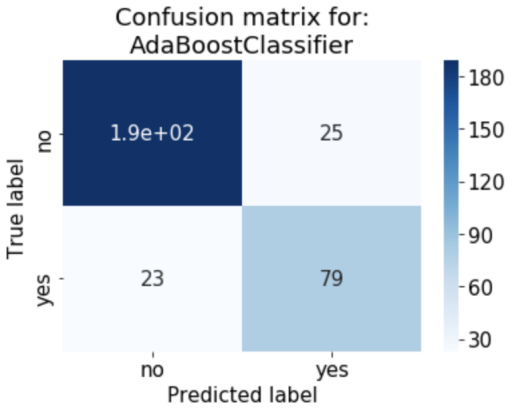
	1%	10%	100%
acc_test	0.693038	0.775316	0.848101
acc_train	0.626667	0.843333	0.843333
f_test	0.511111	0.650183	0.762548
f_train	0.453608	0.787321	0.766488
prec_test	0.528736	0.639640	0.759615
prec_train	0.463158	0.802083	0.754386
pred_time	0.010954	0.009249	0.009174
recall_test	0.450980	0.696078	0.774510
recall_train	0.419048	0.733333	0.819048
train_time	0.049331	0.077771	0.157091

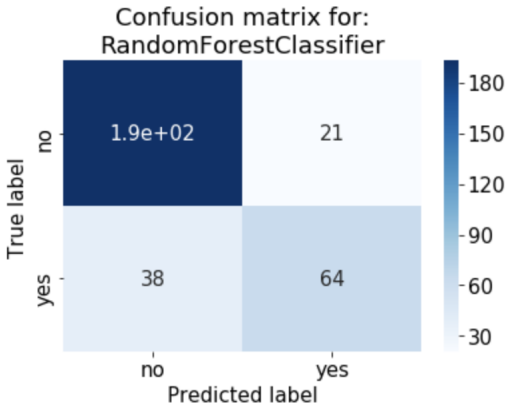
RandomForestClassifier

	1%	10%	100%
acc_test	0.629747	0.708861	0.813291
acc_train	0.613333	0.810000	0.993333
f_test	0.103093	0.477099	0.723982
f_train	0.170732	0.779037	0.996132
prec_test	0.173913	0.625000	0.752941
prec_train	0.280000	0.887097	1.000000
pred_time	0.001864	0.002081	0.002364
recall_test	0.039216	0.245098	0.627451
recall_train	0.066667	0.523810	0.980952
train_time	0.009974	0.011207	0.025759

And here we have the confusion matrices of the algorithms used:







Based on the above performance, although with a slightly difference, we can conclude that the Gradient Boosting Classifier (GBM) model is performing better than the other two models. With a good accuracy and F-score, we think that GBM is the most appropriate model for our problem.

Refinement

After we have chose GBM as our model, we tune the parameters. Many strategies exist on how to tune parameters. See for example this blog post on Machine Learning Mastery for some guidance from academic papers. Most data scientist see number of trees, tree depth and the learning rate as most crucial parameters.

We use grid search (‘GridSearchCV‘) to do the fine tuning. For that, we test the parameters “max_depth”, “min_samples_split”, and“subsample” with “random_state” initially equal to 30. What we get as result in terms of metrics is the following:

```
Unoptimized model
------
Accuracy score on testing data: 0.8797
Precision score on testing data: 0.7909
Recall score on testing data: 0.8529
F-score on testing data: 0.8026

Optimized Model
------
Final accuracy score on the testing data: 0.8892
Final precision score on testing data: 0.8073
Final recall score on testing data: 0.8627
Final F-score on the testing data: 0.8178
```

Results

Model Evaluation and Validation

As already was said, a validation set was used to evaluate the model. And the results are shown bellow. First, let's have a look at the following table:

Metric	RFV Model	GBM - Unoptimized Model	GBM - Optimized Model
Accuracy Score	0.5944	0.8797	0.8892
Precision Score	0.4507	0.7909	0.8023
Recall Score	0.6720	0.8529	0.8627
F-score	0.4825	0.8026	0.8178

The table express the improvement of the the metrics after the tunning. It also indicates that the model was chosen because of it. It is important to mention that the hyperparameters were chosen because they represented the best combination among the ones tested. To verify the robustness of the model, the metrics were utilized and comparison with the RFV model. Also, it was used some later dataset from GP, where we realize that, for a long period, the RFV model had a better performance. This means that our model didn't generalized well in time. I think that this happened because of the short period of time of data collected that we had for analysis and to create the model. This is discussed as well at the Improvement subsection of the Conclusion section.

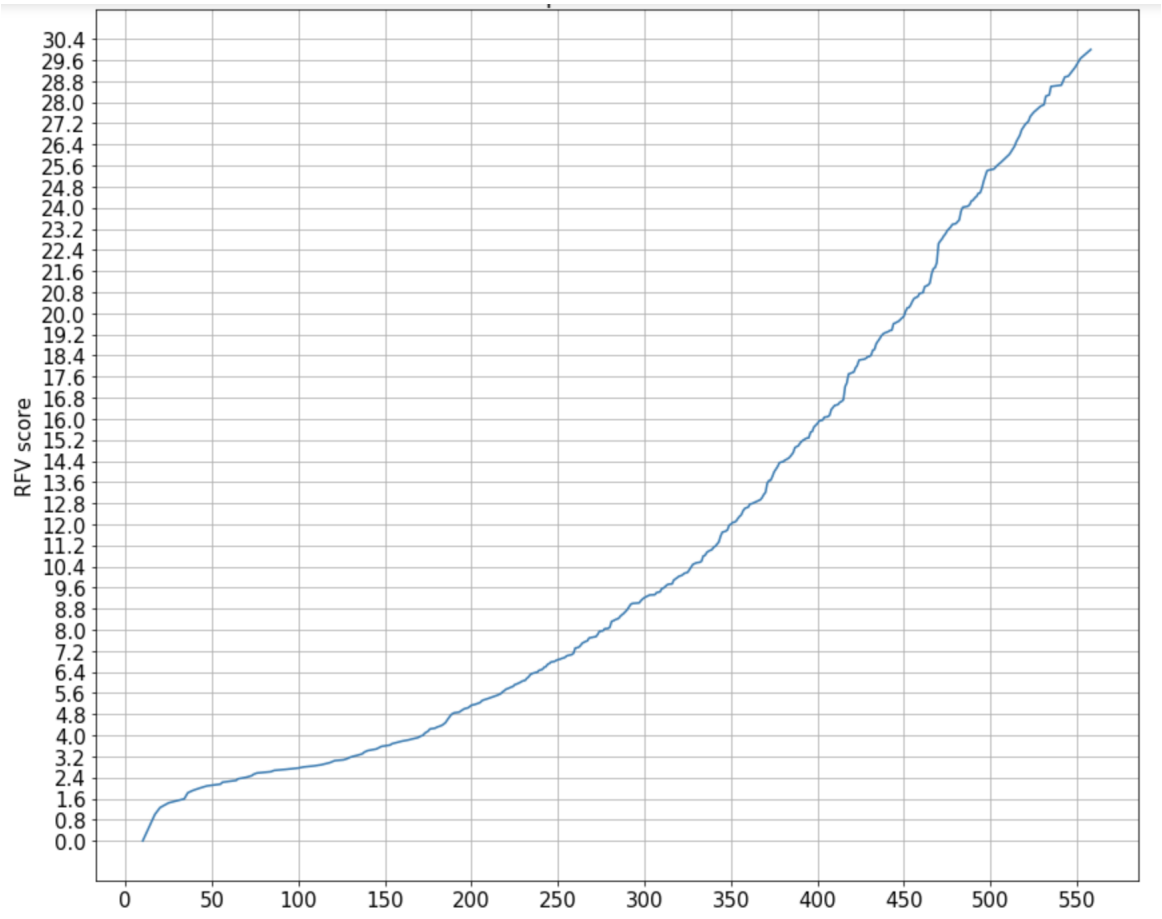
Justification

The RFV model, although a good model, is limited because it takes in consideration only three variables. It is a very rich model, though, and a more stable one if we take in consideration a long period of time. But, as it is shown is the table in the last subsection, the GBM model, for one month, performed much better than the RFV model. All the metrics, Accuracy Score, Precision Score, Recall Score, and F-Score, are much better in the GBM model rather than in the RFV model.

Conclusion

Free-Form Visualization

The following graphic express in y-axis the RFV score of the people and in the x-axis the quantity of people who practice the Churn and are under that score. Let's have a look.



As we can see, there are a considerable amount of people who practiced the churn but the RFV score model was not able to catch them. Our model, as the metrics improvement indicated, was able to catch a significant amount of the people who did not practice the Churn. Hence, we think this is a relevant characteristic of the model.

Reflection

The process used in this project can be describe as:

- Define an important and relevant problem for the company GP;
- Data preprocessing and cleaning;
  - Preprocess feature columns;
  - Data cleaning;
  - Feature Scaling - Standardization, Normalizing data;
- Creation of the RFV model;
- Used k-means and the RFV model to balance the data;
- Evaluate Algorithms:
  - Build models;
  - Select best model;
  - Make predictions on the validation set;
- Final conclusions.

For me, the most difficult item was the one that I had to deal with the imbalaced data problem, where k-means and the RFV model was used. Since we could lost a lot of information in this problem, it was difficult to think in some solution.

Improvement

To achieve an better solution, it would be necessary to collect more data. The dataset were collected during the month of July and the company, GP, had an internal tracker in production only since May. So, the amount of data collected was too little. And, as mentioned before, because of this, if we take in consideration a longer period of time, the RFV model perform better than our model.

Also, it would be interesting to get external information from the user, like the income, etc. So, in order to improve the model, I would go after more data and make tests.