Towards the Machine Learning Algorithms in Telecommunications Business Environment

Moisés Loma-Osorio de Andrés[1], Aneta Poniszewska-Marańda[2(✉)], and Luis Alfonso Hernández Gómez[1]

[1] Universidad Politécnica de Madrid, Madrid, Spain
M.loma-osorio@alumnos.upm.es,
luisalfonso.hernandez@upm.es
[2] Institute of Information Technology, Lodz University of Technology, Lodz, Poland
aneta.poniszewska-maranda@p.lodz.pl

Abstract. We live in times where companies and individuals are dealing with extremely large amounts of data coming from all different kind of sources. This data includes a lot of very valuable information, which, most of the time, cannot be inferred at first sight. Therefore, in today's businesses there is a growing necessity of discovering efficient and useful information out of the data that has been gathered. This is the reason why Machine Learning, a technology that has been developed since mid-20th century, is one of the biggest growing technologies in this last decade, being one of its most popular applications in the field of data. The paper presents an analysis what techniques are available for starting with a Data Science project, how easy they are to implement, and how they can be applied in a real world case. The data that was worked with for this project was gathered from a telecommunications company.

Keywords: Machine learning · Data processing · Machine learning model

1 Introduction

We live in times where companies and individuals are dealing with extremely large amounts of data coming from all different kind of sources. This data includes a lot of very valuable information, which, most of the time, cannot be inferred at first sight. Therefore, in today's businesses there is a growing necessity of discovering efficient and useful information out of the data that has been gathered. This is the reason why Machine Learning (ML) is one of the biggest growing technologies in last decade, being one of its most popular applications in the field of data. Although it can help to interpret the data that has been obtained and getting useful information regarding existing trends, one of the most valuable applications is the ability of predicting future behaviors and trends based on the current ones (predictive model). This way companies not only know how to react to the current situation, but also are able to prepare for any future problems that they might encounter.

In order to get a deep understanding of how these technologies are being used even in traditional companies, we analyzed the data gathered from a company that offers

M. Themistocleous et al. (Eds.): EMCIS 2020, LNBIP 402, pp. 79–96, 2020.
https://doi.org/10.1007/978-3-030-63396-7_6

telecommunication services to their clients, such as phone and internet services, television and movie streaming, and device protection. With this data, different Machine Learning techniques were developed and applied in order to find out if they work properly, to identify which are the main reasons that clients stop hiring the company's services and to predict how likely a new client is to do so, as well as finding out how the company can act in order to stop this from happening.

The main objective of the project is to get a deep understanding of why Machine Learning techniques are being so widely used nowadays in the field of data, why every single company is starting to implement them and what benefits can they obtain from their application, and how a Machine Learning model can be tuned to achieve different results.

The paper presents the study of what techniques are available for starting with a data science project, how easy they are to implement, and how they can be applied in a real world case. The data that was worked with for this project was gathered from a telecommunications company.

The paper is structured as follows. Section 2 presents a short overview of machine learning and data science algorithms. Section 3 describes the model building process with determining the resources and data processing techniques. Section 4 deals with presentation of research methodology and implementation of ML selected methods while Sect. 5 presents the analysis of results of ML methods implementation in the given case.

2 Machine Learning and Data Science Algorithms

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves [1–5].

Machine learning approach involves mostly supervised or semi-supervised techniques. Standard classification methods are applicable, however sequence-based methods that use the whole sentences as sequences of words instead of sets of single words, are more widely used. *Deep learning* is a sub-domain of machine learning that uses neural network architectures, in NER especially valuable are architectures that capture long-term dependencies and operate on data sequences.

Supervised Machine learning techniques are the most common application in Data Science. Starting with a dataset with several entries, we can select a target (output) to make predictions for, based on the rest of the features (inputs). When training the model, the algorithm will stablish relations between the features and the target, and then will use the acquired knowledge to predict new values when faced with new data.

Depending on the type of target that we are trying to predict, it is possible to identify two different categories of algorithms: regression and classification [18]. The main difference between them is the type of target they are trying to predict: regression algorithms are used when the target is a continuous variable, for example, trying to predict the price of a new house; while classification algorithms are used if the target is a discrete value, for example, trying to identify if an incoming email should be

classified as spam or not spam. In the problem that is being tackled by this project, the objective is trying to predict if a customer from a company will stop hiring its services or if they will stay, which can be easily identified as a classification problem. In the case of leaving customers in a business environment, this is known as churn prediction. In Machine Learning, classification refers to a predictive modeling problem where a class label is predicted for a given example of input data [2].

In the problem that is being tackled by the project, the objective is trying to predict if a customer from a company will stop hiring its services or if they will stay, which can be easily identified as a classification problem. In the case of leaving customers in a business environment, this is known as churn prediction. Many of machine learning algorithms are used for this kind of problems, such as logistic regression, Naïve Bayes algorithm, k-nearest Neighbors (KNN) algorithm, decision trees. Four of them were implemented for the data gathered from a telecommunications company: random forest [6–8], k-nearest Neighbors [9], extreme gradient boosting [10, 11] and logistic regression [12, 13].

3 Model Building – Resources and Data Processing

Since the nature of the project was data science, there are a few questions needed to answer firstly. First, where do we get the data that will be used in the project? Which programming language should be used to work with it? Which development environment is best suited for this kind of project?

The obvious first step in any data science project is getting the data that will be analyzed and used to train the machine learning model. Though this might seem like a trivial step, sometimes this data is not immediately accessible, depending on its nature. For example, data concerning medical records of patients is usually protected, and this means that it can be a challenge to start a data science project about this topic. Since the objective of this project was to analyze the behavior of clients of a Telecom company, we needed access to personal data from companies, which is rarely shown to the public. Such data was found at Kaggle.com. It contains the following information [14]:

- Customers who left within the last month – column called *Churn*.
- Services that each customer has signed up for – phone, multiple lines, internet, online security, online backup, device protection, tech support, and streaming TV and movies.
- Customer account information – how long they have been a customer, contract, payment method, paperless billing, monthly charges, and total charges.
- Demographic info about customers – gender, age range, and if they have partners and dependents.

The project was developed in *Python* language with the use of *Sckit-lear*n library (integrating a wide range of machine learning algorithms for medium-scale supervised and unsupervised problems) and *pandas* library (providing fast, flexible, and expressive data structures designed to make working with “relational” or “labelled” data) [15].

Before starting with the testing of Machine Learning models, it is very important to understand the nature of the data we are working with. A first analysis can reveal some parameters that do not contribute too much to the final result (in this case, whether the customer is going leave the company and hire the services of a different company). Right now, the data is stored in CSV (comma separated values) format, which does not allow for an easy interpretation, so we have to load the data and transform it into an easy to read and manipulate format.

To load the data, we just need to specify the path to the CSV file where it is stored and pass it as an argument to the pandas library method *read_csv*. To check what the data looks like, we can call the DataFrame method *head()*, which returns the first five columns of the DataFrame (Fig. 1). The important thing is to identify which of these columns is the target to predict: in this case, it is the column *Churn*. The fields in this column take only two values: *Yes*, if the customer did not continue hiring the services of the company; or *No*, if they stayed with the company.

```
In [1]: # Load the data as a pandas Data Frame class
        full_data = pd.read_csv('telco-customer-churn.csv')

        # Check the first 5 rows of the data to take a peak at the data
        full_data.head()
```

Out[1]:

	customerID	gender	SeniorCitizen	Partner	Dependents	tenure	PhoneService	MultipleLines	InternetService	OnlineSecurity	...	DeviceProtection	TechSup
0	7590-VHVEG	Female	0	Yes	No	1	No	No phone service	DSL	No	...	No	
1	5575-GNVDE	Male	0	No	No	34	Yes	No	DSL	Yes	...	Yes	
2	3668-QPYBK	Male	0	No	No	2	Yes	No	DSL	Yes	...	No	
3	7795-CFOCW	Male	0	No	No	45	No	No phone service	DSL	Yes	...	Yes	
4	9237-HQITU	Female	0	No	No	2	Yes	No	Fiber optic	No	...	No	

5 rows × 21 columns

Fig. 1. First five rows of the DataFrame in pandas library

Some of the algorithms used to build the machine learning models in our approach can only predict numerical values, so we need to change the values ‘Yes’ for ‘1’ and the ‘No’ values for ‘0’.

When working with a classification problem, it is important to train the Machine Learning models with data that is balanced, meaning that there needs to be a balance between positive classes (in this case, 1 in the Churn column) and negative classes (0 in the Churn column). In the case of the problem being tackled we had the situation where there is a larger number of non-Churn customers than Churn ones:

- Number of non-Churn customers in the dataset: 5163.
- Number of Churn customers in the dataset: 1869.
- Proportion of Churn vs No Churn: 0.27.

The exact percentage of leaving customers is 27% of the total that is not as extreme as we would like to have, but the performance of the models will still see improvement if we balance this data.

It is important to clarify that data balancing cannot be done to the whole dataset. This is because data has to be divided in training and testing – to train the algorithm and to measure its performance. Both of them are obtained by dividing the original dataset, and balancing the data can only be done to the dataset that will train the model. Otherwise, this would cause a problem known as Data Leakage, since there would be some samples that have been created from already existing data in the testing dataset, which would cause the model to seem much more effective than it really is (fake results). This means that, even though data balancing is being explained before splitting the data, this can only be done afterwards.

Using the SMOTE (Synthetic Minority Oversampling TEchnique) algorithm for balancing the data produced the results presented in Fig. 2.

```
Length of original data is 5625
Length of oversampled data is 8244
Number of no Churn in oversampled data: 4122
Number of Churn: 4122
Proportion of Churn data in oversampled data is 0.5
Proportion of no Churn data in oversampled data is 0.5
```

Fig. 2. Proportion of customers in the training dataset after balancing both classes

4 Research Methodology and Implementation of Selected Methods

After loading, analyzing and preparing the data to work with, it is time to test the selected Machine Learning algorithms indicated in Sect. 2.

The Machine Learning process can be (greatly) simplified in two steps: *training* and *testing* of the model. Since we implement supervised learning algorithms, this means that the training step consists in feeding the model with data where a finite number of feature columns map to a single target column whose values are known in advance.

At this point, we have the modified data split in two different sets: training and validation sets. Both of these are divided in features (X) and target (y – Churn column). The training set is used to fit the different models, which means to stablish relationship between features and target. The validation set is used to evaluate the performance of the

model: it uses the validation features set to predict a value for the target set, and then compare these predictions with the real validation targets set, and use different metrics (that was already explained beforehand) of choice to return a performance measure.

This process was done for each of the four models, while changing a few parameters to optimize the results, and then compare the different performance metrics of each models to select the one that gives us the most satisfying results. When trying to improve the performance of each individual model, we performed two main operations: *feature selection*, which consists in finding out which features have lower contribution to the model and getting rid of them to see if it improves its performance; and *hyperparameter tuning*, which means trying several parameter configurations for each algorithm (these parameters are different depending on the way the algorithm works) and find out which combination of parameters works best for each model.

4.1 Random Forest Model

The first model implements Random Forest (RF) classifier algorithm. There are a few parameters that can be set in order to better adapt the model to our data – the considered parameters are: *n_estimator*s (number of trees in the forest), *max_dept*h (maximum depth of the tree) and *max_feature*s (number of features to consider when looking for the best split). Firstly, the model with all parameters set at their default values was considered, in order to test if the changes that are made improve on the performance of this basic model. The results are presented in Fig. 3 (top part). The first results with AUC_ROC score equals to 0.7087 are not satisfying. It is because some of the lower ranked features might introduce noise and causing the model to overfit (there are way too many variables to consider).

Therefore, only the first few features were kept to see if the performance of the model improves (Fig. 3, middle part). The recall metric value is now 0.64, an increase of 0.07 point compared to the last case (before deleting some features), which is definitely an improvement. The AUC_ROC value also increases about 0.02 points, once again proving that a large amount of the features were more harmful than helpful to the model.

Finally, some of the parameters were tuned to check to get further improvement. In general, the higher the value of *n_estimators* (number of trees in the ensemble) the more reliable the prediction will be, but the computational cost will also be higher (Fig. 3, bottom part). These results are much more promising than the first ones but Random Forest Algorithm is one of the simpler to implement, so better results with other algorithms are expected.

	precision	recall	f1-score	support
0	0.85	0.85	0.85	1041
1	0.57	0.57	0.57	366
accuracy			0.77	1407
macro avg	0.71	0.71	0.71	1407
weighted avg	0.78	0.77	0.77	1407

AUC_ROC score for this model: 0.7087

	precision	recall	f1-score	support
0	0.87	0.82	0.84	1041
1	0.55	0.64	0.59	366
accuracy			0.77	1407
macro avg	0.71	0.73	0.72	1407
weighted avg	0.78	0.77	0.78	1407

AUC_ROC score for this model: 0.7279

	precision	recall	f1-score	support
0	0.89	0.80	0.84	1041
1	0.55	0.72	0.62	366
accuracy			0.77	1407
macro avg	0.72	0.76	0.73	1407
weighted avg	0.80	0.77	0.78	1407

AUC_ROC score for this model: 0.7556

Fig. 3. Metrics report for Random Forest model and modified Random Forest model with the most important feature

Figure 4. how the ROC curve improved after all the changes we made. The green curve represents the basic model, and the blue one is the model after all the improvements we made to the model and the training data. The red line represents an estimator with 0.5 AUC, which is the score that an estimator that predicted random results (50% of the time class A and 50% class B) would get.

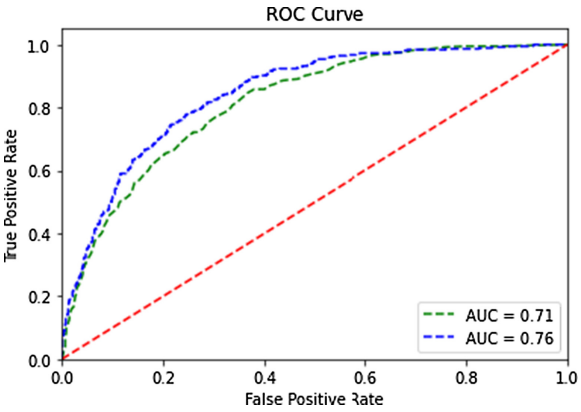

Fig. 4. ROC curves for the basic and improved RF models

4.2 K-Nearest Neighbors Model

K-Nearest Neighbors (KNN) model is the most popular classification algorithm. The same, firstly, default model was tested and then it was improved. By default, the model sets k = 5 (k being number of neighbors). Looking at the values of precision and recall (0.48 and 0.67 respectively) it seems like this model predicts positive classes much more often than it should (less than 50% of predicted positives are actual positives). It finds a fair amount of actual positive classes, but it cannot be considered a good model when looking at these results (Fig. 5, top part).

KNN implements a lazy learning model, which means that it waits for the training data to perform any calculations. This means that removing features from the set can only be done by intuition, so we will use the same ranking as in Random Forest model. Even though it might not be the optimal approach, it is still better than removing features randomly. This time, the best result is achieved when using only the best 8 features (Fig. 5, middle part). Results are improved – recall and AUC_ROC score improve respectively 0.04 and 0.03 points, so even if the deleted features might not have been the optimal ones, the performance of the model is improving rapidly.

So far, all the model tests have been done with a value of k = 5 neighbors which, with such a large training set (around 8000 entries), it is a really low number and can easily lead to wrong predictions. It is necessary to check which K number grants better results for this dataset. In this case, a number between 100 and 120 seems to consistently return the best possible values for both metrics, so when setting the value of K to 117 neighbors, we obtain the results presented in Fig. 5 (bottom part). Setting the number of neighbors to a higher value improves the results of pretty much all the metrics. Focusing on the recall and AUC_ROC, the results have improved greatly, achieving a recall score of 0.81, which is even better than the best score achieved with the Random Forest model. The difference in ROC curves for both KNN models presented in Fig. 6 shows how drastic the improvement actually was.

	precision	recall	f1-score	support
0	0.86	0.75	0.80	1041
1	0.48	0.67	0.56	366
accuracy			0.73	1407
macro avg	0.67	0.71	0.68	1407
weighted avg	0.77	0.73	0.74	1407

AUC_ROC score for this model: 0.7075

	precision	recall	f1-score	support
0	0.88	0.76	0.82	1041
1	0.51	0.71	0.59	366
accuracy			0.75	1407
macro avg	0.69	0.73	0.70	1407
weighted avg	0.78	0.75	0.76	1407

AUC_ROC score for this model: 0.7333

	precision	recall	f1-score	support
0	0.92	0.73	0.81	1041
1	0.51	0.81	0.63	366
accuracy			0.75	1407
macro avg	0.72	0.77	0.72	1407
weighted avg	0.81	0.75	0.77	1407

AUC_ROC score for this model: 0.7721

Fig. 5. Metrics report for K-Nearest neighbors model and modified K-Nearest neighbors model with removed features

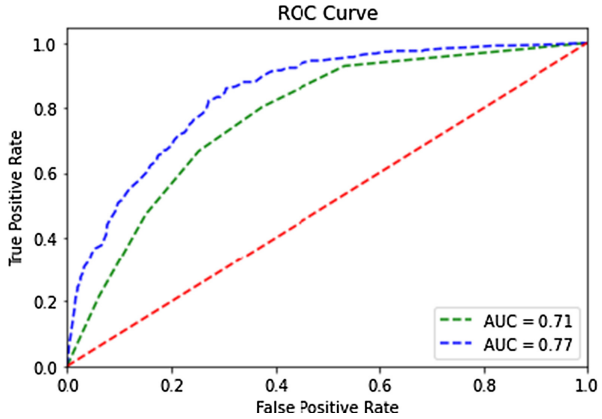

Fig. 6. ROC curves for the basic and improved KNN models

4.3 Extreme Gradient Boosting Model

Extreme gradient boosting (XGB) is a variation of original gradient boosting, one of the more powerful machine learning algorithms developed, and widely used in many data science projects nowadays.

The results obtained for default XGB model are not impressive (Fig. 7, first part). This is because one of the strengths of XGB models are the numerous variables that can be tweaked in order to improve the performance of the algorithm. Some of these parameters include the following [16]: *n_estimator*s (number of estimators in the

```
              precision    recall  f1-score   support

           0       0.85      0.86      0.85      1041
           1       0.58      0.55      0.57       366

    accuracy                           0.78      1407
   macro avg       0.71      0.71      0.71      1407
weighted avg       0.78      0.78      0.78      1407

AUC_ROC score for this model: 0.7058
              precision    recall  f1-score   support

           0       0.92      0.74      0.82      1041
           1       0.52      0.82      0.64       366

    accuracy                           0.76      1407
   macro avg       0.72      0.78      0.73      1407
weighted avg       0.82      0.76      0.77      1407

AUC_ROC score for this model: 0.7783
              precision    recall  f1-score   support

           0       0.97      0.45      0.61      1041
           1       0.38      0.96      0.54       366

    accuracy                           0.58      1407
   macro avg       0.67      0.70      0.58      1407
weighted avg       0.81      0.58      0.59      1407

AUC_ROC score for this model: 0.702
              precision    recall  f1-score   support

           0       0.93      0.71      0.81      1041
           1       0.51      0.85      0.64       366

    accuracy                           0.75      1407
   macro avg       0.72      0.78      0.72      1407
weighted avg       0.82      0.75      0.76      1407

AUC_ROC score for this model: 0.7831
```

Fig. 7. Metrics report for Extreme gradient boosting model and modified Extreme gradient boosting model with deleted features

ensemble), *eta* (weight of each iteration of the loop), *max_depth* (maximum depth of the decision trees), *subsample* (subsample ratio of the training instances).

After a few tests, the following combination of values seems to significantly improve the previous results: n_estimators = 10, eta = 0.1, max_depth = 4, subsample = 0.92. The new obtained results are improved in most metrics, but especially those that we are focusing on. Specifically, recall improved by 0.27 points (which is an extremely high improvement) and AUC_ROC score improved by around 0.07 points (Fig. 7, second part).

Following the tuning of parameters, we will once again try to delete some features in order to see if it can improve the model. Deleting some of the less important columns causes positive effect in the performance of the algorithm. Running the same loop as in previous models returns the value '26' as the number of features to delete in order to maximize the recall value. If we focused only on predicting every single Churn and disregarding every other metric it might be a viable solution, but it is important to find good balance between high recall and rest of the metrics so it seems like deleting features will not help in this instance (Fig. 7, third part).

One alternative option to be considered is setting a value for another parameter: *colsample_bytree*. This works similarly to the subsample feature, but instead of sampling a percentage of the rows in the training data for each iteration of the loop, it does so but for the features (columns). Testing some values, the best results were obtained with a value for this parameter between 0.45 and 0.60. Settling for a value of 0.5 in the end (which at first already seemed like a good choice, since sampling 50% of features each iteration would definitely help in this case, where the training dataset is made of such a large number of features) will grant the following results. Both main metrics have improved after the implementation of this last parameter. So far, this model outperforms the previous one (Fig. 7, fourth part).

Comparing both ROC curve shapes (Fig. 8), it can be appreciated how improved the shape of the final looks compared to the original. This is expected, seeing how XGB is a very popular machine learning algorithm to use in this kind of projects, one of the reasons being the high number of hyperparameters that can be set in order to perfectly fit the data.

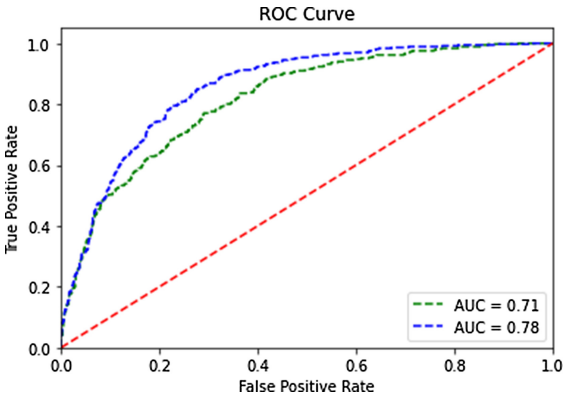

Fig. 8. ROC curves for the basic and improved Extreme Gradient Boosting model

4.4 Logistic Regression

Finally, the Logistic regression (LR) model, commonly used in this type of data science projects, was tested. Firstly, a model with default parameters was trained to get a first grasp of what to expect (Fig. 9, top part). The results are actually pretty decent, even before removing less useful features or tuning some of the hyperparameters. A recall score of 0.81 and an AUC_ROC score of 0.77 are better results than those from previous models even after tuning them.

In this case, the *Recursive Feature Elimination (RFE)* was used to tune the logistic regression model. This was achieved by fitting the algorithm used in the core of the model, ranking features by importance, discarding the least important features, and re-fitting the model. This process is repeated until a specified number of features remains [17]. In order to see which value of features left make for the best improved model, we can run a loop and find the best resulting value for ROC_AUC. In this case, it happens to be nine features out of the 43 original ones.

There is an improvement on the performance, but not as impactful as we could have predicted (since the original result was pretty good to start with). This is as good as it can get, because tuning some of the Logistic Regression hyperparameters does not seem to improve the performance of the model as it did with the rest of the algorithms (Fig. 9, bottom part). The final and original ROC curves are represented in Fig. 10.

	precision	recall	f1-score	support
0	0.92	0.73	0.81	1041
1	0.51	0.81	0.63	366
accuracy			0.75	1407
macro avg	0.72	0.77	0.72	1407
weighted avg	0.81	0.75	0.76	1407

AUC_ROC score for this model: 0.7712

	precision	recall	f1-score	support
0	0.93	0.71	0.81	1041
1	0.51	0.84	0.63	366
accuracy			0.75	1407
macro avg	0.72	0.78	0.72	1407
weighted avg	0.82	0.75	0.76	1407

AUC_ROC score for this model: 0.7754

Fig. 9. Metrics report for Logistic regression model and modified Logistic regression model with deleted feature

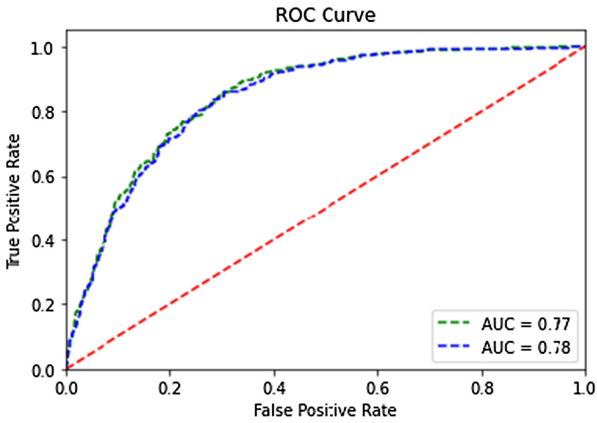

Fig. 10. ROC curves for the basic and improved Logistic regression model

As expected, the performance of the model didn't improve that much even after trying to get the best possible results. This speaks of Logistic Regression as a fairly simple algorithm, that works very well if there is no need to optimize the result for a specific metric.

5 Results and Discussion – Choosing a Machine Learning Model and Its Implementation

After testing four different algorithms and obtaining performance results for all of them, they were compared and one of them was selected to be implemented to tackle the problem of detecting leaving customers in a Telecommunications company environment. While comparing the algorithms, following aspects should be kept in mind:

- There are many metrics used to evaluate the performance of a machine learning model. In our case, the focus was on two metrics: *recall* (to be able to detect as many leaving customers as possible) and *ROC_AUC score* (to get a grasp at how good the model is at distinguishing between positive and negative classes). When testing the different models, we aimed to achieve good balance between these two metrics.
- Only algorithm's metrics performances were taken into account when choosing between models. Other variables, such as overall training and testing speed, do not really matter for this problem, since it does not have real time implications (there is no need for the algorithms to run as fast as possible, sacrificing overall performance).

The results obtained for performances of tested machine learning models are presented in Table 1.

Table 1. Performance metrics results for all tested ML models

	Random forest	KNN	XGBoost	Logistic regression
Recall	0.72	0.81	0.85	0.84
ROC_AUC	0.7556	0.7721	0.7831	0.7754

As expected, Extreme Gradient Boost algorithm performed the best results out of all the tested. The possibility of fine-tuning the hyperparameters to adapt it as well as possible to the data, as well as reinforcing an original weak estimator and transforming it into a strong one are some of the reasons why it is being considered as one of the best performing machine learning algorithms for data science projects. Thus, the chosen final model implements this algorithm, since it is the one which was expected to grant the most reliable results when exposed to new customer's data.

In order to test how this project could be implemented in a real-world case, we created a Python simulation that creates basic synthetic data based on the dataset that has been used to train the model. Originally, the idea behind this project was to use this data as the core dataset for the project, but this idea was quickly discarded, as it is not the best way to evaluate the performance of the different machine learning algorithms. Instead, we used the Kaggle.com dataset to train and test all the models, and then use own program to generate synthetic data bases on this same dataset.

Some considerations were made when developing this program, but in the end this was just a symbolic step in the project, to simulate a real situation where the Machine Learning model is being tested with new data where we don't know yet if the customer will leave or stay. This means that there is no real performance measure to be made with this data, and it is just to recreate a real-world business situation. Still, the objective was to make the synthetic somewhat resembling of the dataset that was used to train the model so that the predictions could be as reliable as they can be. Some of the considerations that were made are the following:

- Gender distribution is 50% between male and female.
- The customer must hire at least one service from the company.
- Probability distribution of the values in different features is similar to the one of the original dataset.
- Monthly charges are calculated based on the number of services the customer has hired.
- Neither monthly nor total charges exceed the minimum or maximum values that were represented in the original data.
- There are some variations added in the calculations of the prices (to add some variance in the dataset).

The Kaggle.com dataset was again used (this time, in its entirety) as training data for the model, while the synthetic data was used as testing set (this time, it is not really labelled as validation data, since it is not used to validate the performance of the model). The steps that are performed to prepare the data are, in order:

- Changing the format of tenure feature values into categorical (from months to range of years, as we already did before) for both training and testing datasets.
- Applying one-hot encoding to both datasets, once again getting rid of the useless features (those that represented no phone service/no internet service).
- Normalizing the numerical columns (once again, monthly and total charges) in both datasets with a MinMax scaler in.
- Applying SMOTE algorithm *only to the training dataset*, in order to balance the data (same number of occurrences for both Churn and no Churn).

Once the data is prepared for the model, next step is defining the latter. As mentioned earlier, the final decision was to create a model that runs an Extreme Gradient Boosting algorithm, with the same parameters that were defined previously when testing the model.

After training the model and computing the predictions, they were added to the synthetic dataset (these steps are represented in Fig. 11) to have the data as organized as possible. The random set of customers and the predicted outcome are presented in Fig. 12.

```
final_model = XGBClassifier(learning_rate=0.1, max_depth=4,
                            subsample=0.92, colsample_bytree=0.33)
xgb_model.fit(final_X_train, final_y_train,
          early_stopping_rounds=10,
          eval_set=[(X_val, y_val)],
          verbose=False)

predictions = xgb_model.predict(final_X_test)

new_data['PredictedChurn'] = predictions
```

Fig. 11. Code cell to obtain the new predictions and joining them with the testing dataset

gender	Male	gender	Female	gender	Male
SeniorCitizen	Yes	SeniorCitizen	No	SeniorCitizen	Yes
Partner	Yes	Partner	Yes	Partner	No
Dependents	No	Dependents	Yes	Dependents	No
tenure	(3, 4)	tenure	(0, 1)	tenure	(1, 2)
PhoneService	Yes	PhoneService	Yes	PhoneService	No
MultipleLines	Yes	MultipleLines	Yes	MultipleLines	No phone service
InternetService	DSL	InternetService	Fiber optic	InternetService	DSL
OnlineSecurity	No	OnlineSecurity	Yes	OnlineSecurity	No
OnlineBackup	No	OnlineBackup	Yes	OnlineBackup	No
DeviceProtection	Yes	DeviceProtection	Yes	DeviceProtection	No
TechSupport	No	TechSupport	No	TechSupport	Yes
StreamingTV	Yes	StreamingTV	Yes	StreamingTV	Yes
StreamingMovies	Yes	StreamingMovies	Yes	StreamingMovies	Yes
Contract	One year	Contract	Month-to-month	Contract	Month-to-month
PaperlessBilling	Yes	PaperlessBilling	Yes	PaperlessBilling	No
PaymentMethod	Credit card (automatic)	PaymentMethod	Electronic check	PaymentMethod	Mailed check
MonthlyCharges	97.82	MonthlyCharges	118.75	MonthlyCharges	65.55
TotalCharges	3691.73	TotalCharges	104.5	TotalCharges	855.43
PredictedChurn	0	PredictedChurn	1	PredictedChurn	1

Fig. 12. Random set of customers and predicted outcome in synthetic dataset

For this first customer, whose features are represented in Fig. 12 the model predicted a value of 0, which means No Churn, or that he will stay with the company. Looking at some of the feature values, we observe he has been in the company for a decent amount of time (between 3 and 4 years), that he has hired DSL over Fiber Optic (which latter correlates to higher churn rates), he has a one year contract (month-to-month usually meant higher churn probability), does not pay by electronic check, and its monthly rates are not in the lower side of the spectrum (which we saw correlated with much higher leaving rates). Thus, the prediction that this customer will stay with the company seems pretty safe.

The second customer (Fig. 12) was predicted to leave by machine learning model. When looking at the data for her, it is no wonder why: tenure value belonging in the first group (which it's indicative of very high churn probability), optic fiber as internet service type, month-to-month contract, electronic check as payment method, low total charges (which were indicative of low tenure and high monthly charges), etc. She ticks almost all the boxes to qualify as a surely leaving customer.

The final customer we are going to look at was also predicted to leave. This time, it is not as obvious as to why, so we have to trust the model on this one. Being a senior citizen and having a monthly contract are some of the most common features in leaving customers, but the rest of parameters should not be too alarming. In this situation, it is important to remember that this model was optimized to maximize its recall value (while still maintaining good overall performance, measured by the AUC_ROC metric), which means that the main objective was to catch as many leaving customers as possible, while probably classifying some customers that had no intention of leaving as Churn.

The model predicted 36% of the customers to be leaving, which is higher than the 27% of confirmed Churn that were present in the original data. This reinforces the notion that the model is over-predicting leaving customers, which is a sacrifice that has to be made in order to catch as many true churn classes.

6 Conclusions

Machine Learning is a technology that is starting to become present in almost every aspect of our lives. While there are uncountable applications of Machine Learning in many different fields, the objective was to explore its applications in the field of data, a field that is also always evolving, and whose applications are extremely important in many aspects of business' life. The goal was to explore how this technology can be used in order to prevent financial losses for a company, which in this case would be losing customers due to several different factors.

While gathering and storing data is a very important step in the data science process, the focus was exclusively on the analysis and obtention of results steps. Testing the performance of different algorithms that are used every day in many data science projects, comparing its results and finding out which one was better suited to tackle this problem was the main focus of the works.

We decided to optimize the model to detect as many leaving customers as possible (while also maintaining a good overall ratio), since this is probably what a company in

this situation would want to achieve by using machine learning to analyze its data. We made this assumption based on the thought that, for a company, it is probably much more cost-efficient to propose new offers (such as new physical terminals, new phone or reduction in services' prices) to a larger number of predicted leaving customers, even if some of them had no intention of leaving the company to start with, rather than assuming the loss in revenue if some these clients do end up leaving. This is something that should be discussed with the company, to determine which strategy should be followed to achieve the desired results.

The next works can be cover the implementation of cross-validation technique to improve the performance of the models. This would mean to also implement a pipelines technique to keep the data processing step much cleaner and organized. One possible final step would also be to use multiple machine learning models sequentially with the same data: first optimizing for recall and then for accuracy, in order to further solidify the prediction.

References

1. Alpaydin, E.: Introduction to Machine Learning. 4th edn. MIT Press, Cambridge (2020). ISBN 978-0-262-01243-0
2. Russell, S., Norvig, P.: Artificial Intelligence – A Modern Approach. Pearson, London (2009). ISBN 9789332543515
3. Urbanowicz, R.J., Moore, J.H.: Learning classifier systems: a complete introduction, review, and roadmap. J. Artif. Evol. Appl. **2009**(1), 1–25 (2009). ISSN 1687-6229
4. Zhang, J., Zhan, Z.-H., Lin, Y., Chen, N., Gong, Y.-J., Zhong, J.-H. et al.: Evolutionary computation meets machine learning: a survey. Comput. Intell. Mag. **6**(4), 68–75 (2011)
5. Allahyari, M., Pouriyeh, S., Assefi, M., Safaei, S., Trippe, E., Gutierrez, J.B., et al.: A Brief Survey of Text Mining: Classification, Clustering and Extraction Techniques, University of Georgia, Athens (2017)
6. Rokach, L., Maimon, O.: Data Mining with Decision Trees: Theory and Applications. World Scientific Pub Co Inc., Singapore (2008). ISBN 978-981277171
7. Shalev-Shwartz, S., Ben-David, S.: 18. Decision Trees. Understanding Machine Learning. Cambridge University Press, Cambridge (2014)
8. Rodriguez, J.J., Kuncheva, L.I., Alonso, C.J.: Rotation forest: a new classifier ensemble method. IEEE Trans. Pattern Anal. Mach. Intell. **28**(10), 1619–1630 (2006)
9. Cover, T.M., Hart, P.E.: Nearest neighbor pattern classification. IEEE Trans. Inf. Theory **13** (1), 21–27 (1967)
10. Mason, L., Baxter, J., Bartlett, P.L., Frean, M.: Boosting algorithms as gradient descent. In: Solla, S.A., Leen, T.K., Müller, K. (eds.). Advances in Neural Information Processing Systems, vol. 12, pp. 512–518. MIT Press, Cambridge (1999)
11. Hastie, T., Tibshirani, R., Friedman, J.H.: 10. Boosting and Additive Trees. The Elements of Statistical Learning, 2nd edn., pp. 337–384. Springer, New York (2009). ISBN 978-0-387-84857-0
12. Hosmer, D.: Applied Logistic Regression. Wiley, New Jersey (2013). ISBN 978-0470582473
13. Harrell, F.E.: Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis. Springer, New York (2010). ISBN 978-1-4419-2918-1

14. Telco Customer Churn, Kaggle. https://www.kaggle.com/blastchar/telco-customer-churn/data. Accessed 2020
15. Package Overview, Pandas. https://pandas.pydata.org/docs/\\getting_started/overview.html. Accessed 2020
16. XGBoost Parameters, XGBoost. https://xgboost.readthedocs.io/en/latest/parameter.html. Accessed 2020
17. Brownlee, J.: Recursive Feature Elimination (RFE) for Feature Selection in Python, Machine Learning Mastery. https://machinelearningmastery.com/rfe-feature-selection-in-python/. Accessed 2020
18. Śniegula, A., Poniszewska-Marańda, A., Chomątek, Ł.: Towards the named entity recognition methods in biomedical field. In: Chatzigeorgiou, A., et al. (eds.) SOFSEM 2020. LNCS, vol. 12011, pp. 375–387. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-38919-2_31