Term Deposit Marketing Campaign

- Machine Learning Insights for Term Deposit Success -

Project Report

Naly RAZAFIARIFERA and Rayan SASSI

UNIVERSITÉ PARIS 1
PANTHÉON SORBONNE

Contents

Chapter 1

Introduction

A Bank's term deposit is a fixed-term investment that is quite similar to a savings account with some major differences. In a term deposit, the money is invested at an agreed interest rate for a fixed period of time called the term. It can't be withdrawn until the end of the term and in return, the client gets a guaranteed interest rate (usually higher than a typical savings account). This can be considered as a safe investment for a client. For a bank, term deposit subscriptions can represent a non negligible source of liquidity, in which optimizing and increasing the number of clients can be beneficial to its business.

1.1 Problem Definition and Dataset Choice

The Problematic will be the following : Can we use machine learning to improve the bank's marketing strategy and thus optimize the targeting in order to have more people subscribing to a Certificate of deposit (CD).

In order to achieve this goal the dataset that is going to be used is Kaggle's *Bank Term Deposit Predictions* provided by Andy Rasika from *Hugging face*. More than that, the dataset can also be used for customer segmentation analysis. By clustering customers based on their characteristics and behavior, banks can identify different segments of customers with varying propensities to subscribe to term deposits. This information can then be used to tailor marketing campaigns and strategies specific to each segment.

1.1.1 Presentation of the dataset

This dataset is a collection of data related to the direct marketing campaigns conducted by a Portuguese banking institution. It contains various features that provide insights into customer attributes and campaign outcomes. These features include:

- **Age:** The age of the customer.

- **Job:** The occupation of the customer.
- **Marital Status:** The marital status of the customer.
- **Education:** The education level of the customer.
- **Default:** Whether or not the customer has credit in default.
- **Balance:** The balance of the customer's account.
- **Housing Loan:** Whether or not the customer has a housing loan.
- **Contact Communication Type:** The method used to contact the customer (e.g., telephone, cellular).
- **Day:** The day of the month when the last contact with the customers was made.
- **Duration:** The duration (in seconds) of the last contact with customers during a campaign.
- **Campaign Contacts Count:** Number of contacts performed during this campaign for each customer.
- **pdays:** Number of days passed since previously contacted from the previous campaign.
- **poutcome:** Outcome from the previous marketing campaign.
- **y :** Indicates wether a customer subscribed to term deposit.

The dataset is divided into two files: a training set and a test set. The training set comprises approximately 45,000 rows, while the test set consists of around 5,000 rows. Here are the first 5 rows of the training set:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	y
0	58	management	married	tertiary	no	2143	yes	no	unknown	5	may	261	1	-1	0	unknown	no
1	44	technician	single	secondary	no	29	yes	no	unknown	5	may	151	1	-1	0	unknown	no
2	33	entrepreneur	married	secondary	no	2	yes	yes	unknown	5	may	76	1	-1	0	unknown	no
3	47	blue-collar	married	unknown	no	1506	yes	no	unknown	5	may	92	1	-1	0	unknown	no
4	33	unknown	single	unknown	no	1	no	no	unknown	5	may	198	1	-1	0	unknown	no

It's noteworthy that the dataset appears to be well-prepared, with no missing values (NaNs), suggesting a potentially clean dataset. Moreover, as the dataset contains many categorical features, it is important to understand their intrinsic contribution to the dataset to then perform the best preprocessing that enhances the performances of our models. With such dataset, we expect tree-based models such as XGBOOST or RandomForest to outperform other models due to their versatility and their ability to deal with categorical features and their performances on tabular data [1]. However before diving into the modelling, it is necessary to explore the dataset more in detail and address the specific challenges posed by categorical features.

1.2 Exploratory Data Analysis (EDA)

In this section, we embark on a comprehensive exploration of the dataset through Exploratory Data Analysis (EDA). Our goal is to uncover patterns, relationships, and insights that will inform subsequent modeling decisions. For example, looking at different groups of people and seeing the correlation with the outcome.

The first thing worth denoting is that the dataset is highly imbalanced. Out of the 45,000 examples in the training set, only around 5,000 of them have a positive answer.

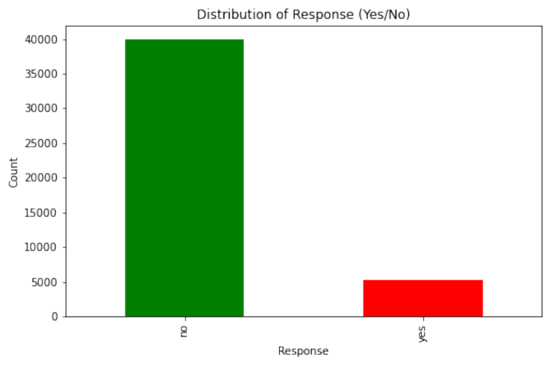


Figure 1.1: Yes/No proportion in the train dataset

As the dataset is highly imbalanced with a scale close to 1:10 it is necessary to use different metrics and/or techniques to overcome this problem.

Another thing we can look at is our quantitative features : age and balance. For example, we can look at the balance distribution and look at some correlation that might exist between the age and bank account balance that could give us problem when working with linear models such as Logistic Regression.

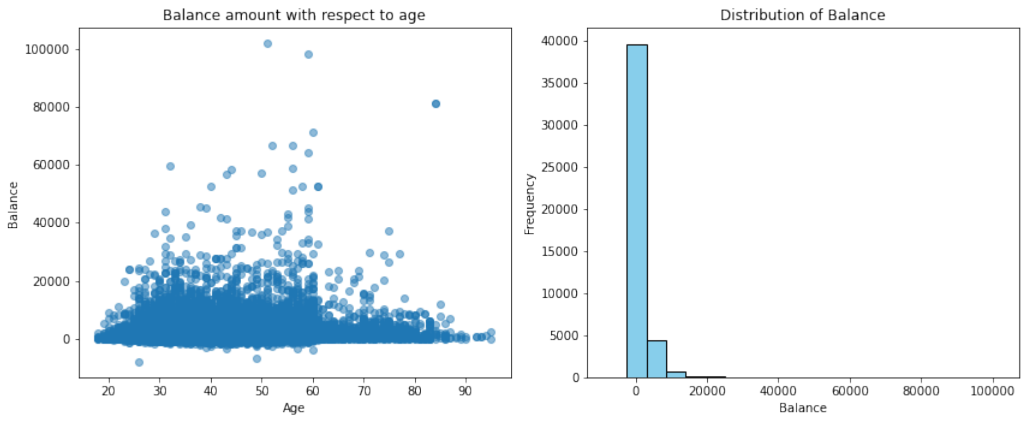


Figure 1.2: Age/Balance scatter and Balance distribution

As we can see above, the balance distribution is skewed on the right side with a mean around 0. Moreover, no direct correlation can be observed with the age.

Now that we have looked at our quantitative features we can look at the qualitative features and more especially how they influence the outcome. By doing that, we might get insights on feature importance and do an appropriate feature encoding. Let's look at a few plots.

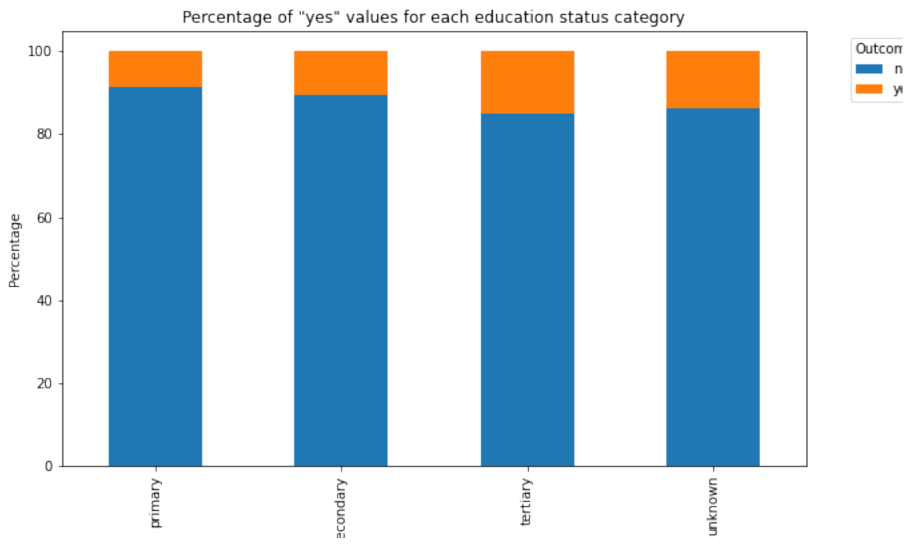


Figure 1.3: Percentage of Yes/No answers for each Education category

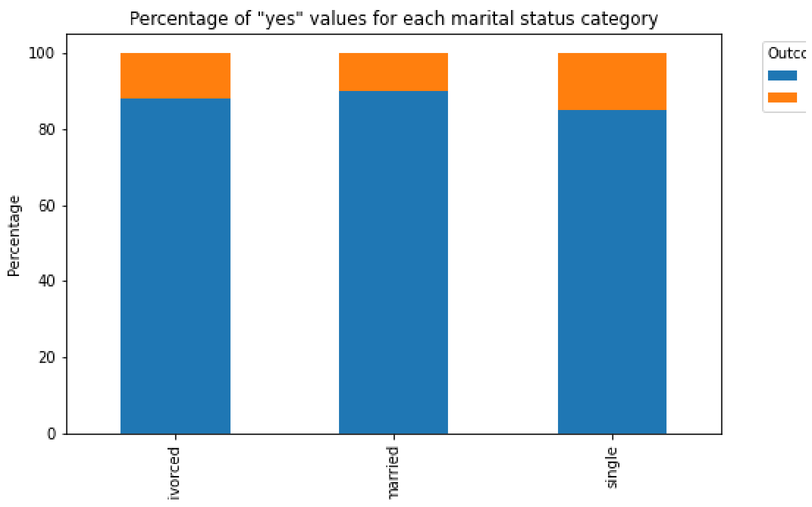


Figure 1.4: Percentage of Yes/No answers for each Marital Status category

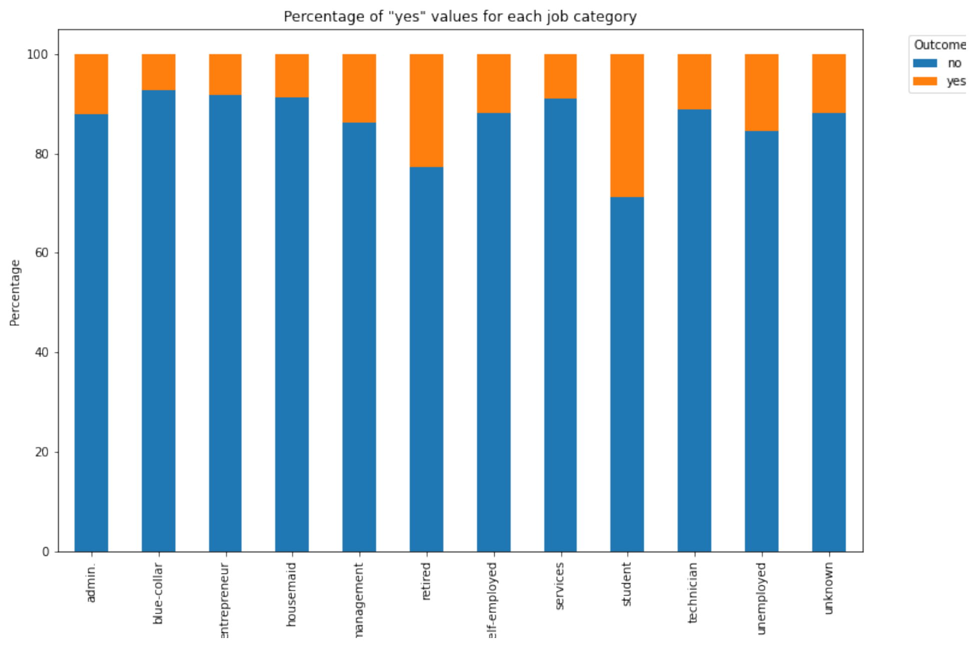


Figure 1.5: Percentage of Yes/No answers for each job category

For the "Education" category, we observe a discernible impact on the outcome. Specifically, within the "tertiary" and "unknown" education levels, the percentage of positive answers is approximately 15%, surpassing the corresponding proportion of 10% observed for the "primary" and "secondary" education categories. Therefore, using an Ordinal encoding on the "Education" category makes sense as an order within that feature seems to exist. For this purpose, we decided to give a weight of 2 for Tertiary, 1.5 for unknown, 1 for secondary and 0 for primary.

For marital status, there is no clear relationship with the outcome so we decided to use OneHotEncoding. Despite increasing the dimensionality of the dataset, it is important to remain true to the feature. While this step might not be important nor necessary for Tree-based methods, it is still needed for linear models and neural network to let them handle the data.

As for the job category, despite some categories having more positive answers than the average, establishing an ranking in this category might be more difficult. Therefore, let's look at some more plots regarding the job category.

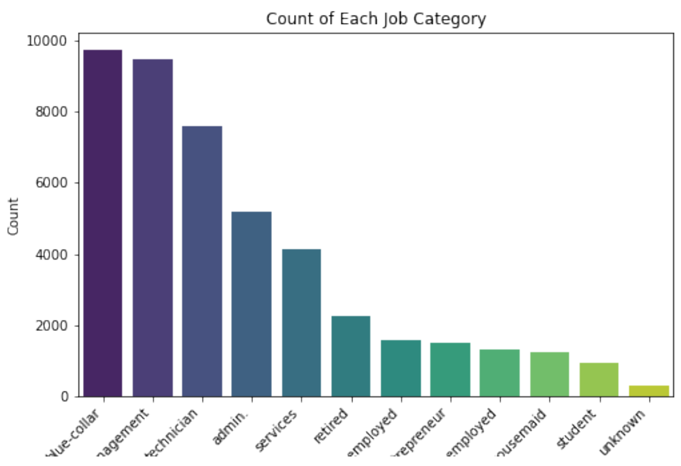


Figure 1.6: Job repartition

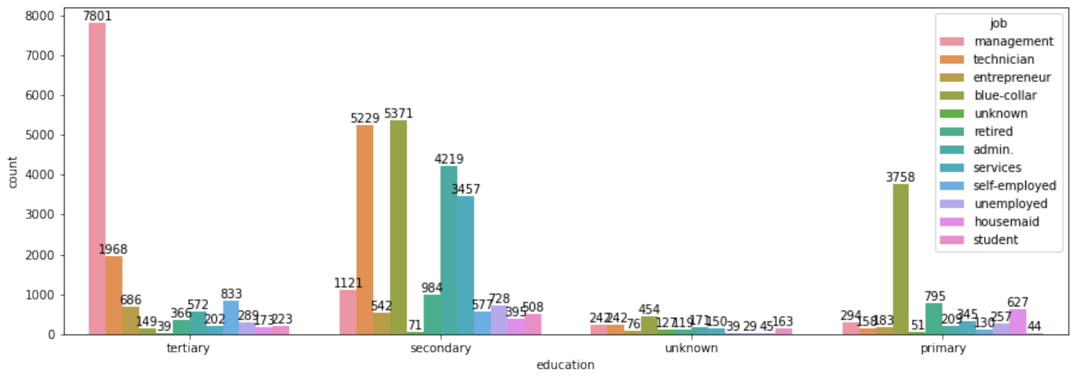


Figure 1.7: Job repartition for every education category

As depicted in figure 1.6, around 50% of our sample has either a blue-collar, management or technician position. And if we look more in detail, in figure 1.7 we can see that management positions represent the vast majority of the tertiary population whereas on the other side, blue-collar positions represent the majority of the primary population. Nonetheless, for secondary and unknown populations, diversity is notably higher. Thus, for this category, we decided to use also OneHotEncoding. Even though some kind of ordinality might exist in this category, it seems rather difficult to establish it so it's a better idea to let the algorithm learn it on its own. Let's look at day and duration plots:

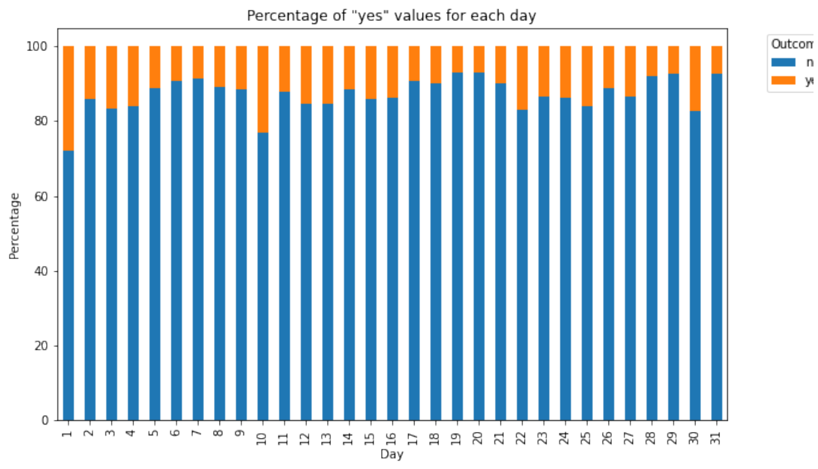


Figure 1.8: Positive answer per day called

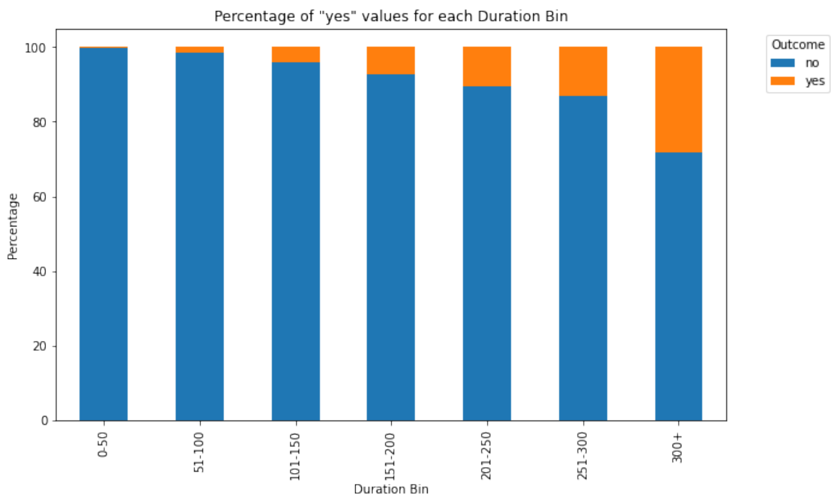


Figure 1.9: Positive answer per call duration

Interestingly, positive answers were significantly more frequent when a called occured on the 1st and 10th of the month. Regarding the duration, this is as expected. When a call is longer, the customer is much more likely to be interested in the product hence the high positive answer percentage.

For the remaining features we observed the same results so we decided to also apply OneHotEncoding to them. More details and graphs are available on the notebook. Following that, we decided to use StandardScaler to the dataset in order to perform PCA. after the preprocessing, we end up with 46 columns.

1.2.1 Principal Component Analysis (PCA)

After preprocessing, when faced with a dataset containing numerous columns, employing Principal Component Analysis (PCA) becomes advantageous for dimensionality reduction and data exploration. It facilitates the visualization of high-dimensional data, allowing for a better understanding of which components contribute the most to the dataset's variance. Therefore, it becomes a valuable tool for detecting outliers as well.

Explained variance is a statistical measure of how much variation in a dataset can be attributed to each of the principal components (eigenvectors) generated by the principal component analysis (PCA) method. In other words, it tells us how much of the total variance is explained by each component. Let's see the results on our dataset.

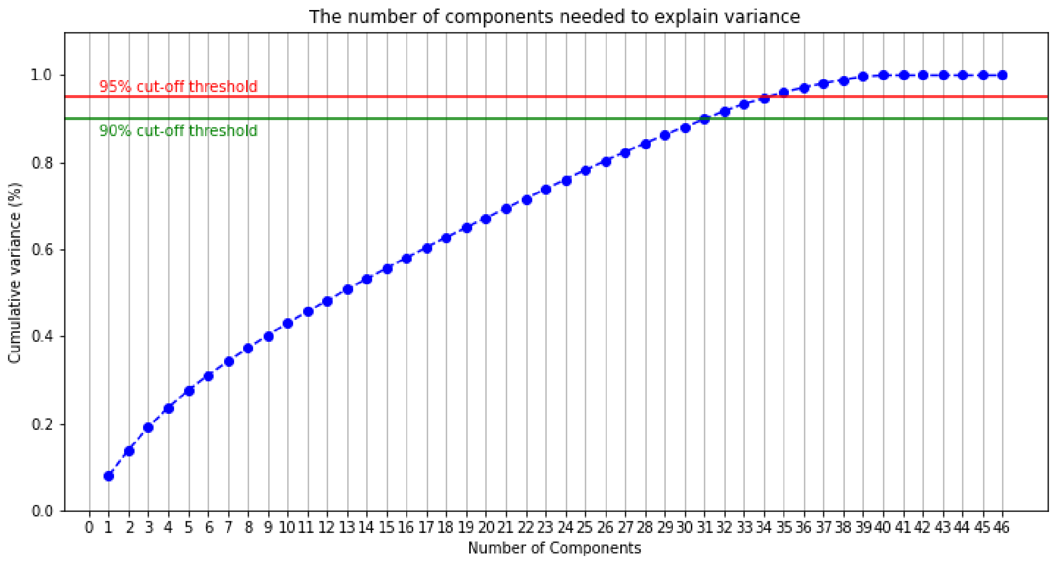


Figure 1.10: PCA Explained Variance

As we can see, we need 31 components to explain 90% of the variance and 35 to explain 95%. The first 3 components only account for 20% of the total variance which could indicate that the information is spread out and not concentrated in a small subset of features. Moreover, if we plot the first two PC we obtain :

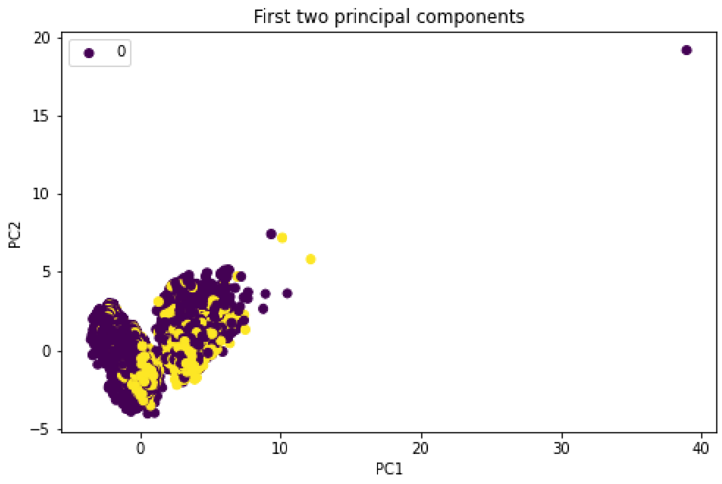


Figure 1.11: PC1 against PC2 scatter plot

As we can see, there is an outlier at the top right corner of the plot. Clusters or trends are not clearly visible which might be explained by the fact that the first two components only represent 20% of the variance.

Chapter 2

Modelling

Now that we explored and analyzed the dataset, it's time to do some modelling to solve our problem. But first, I find important to talk about the imbalanced dataset and how we tried to overcome this hurdle.

Measures we used Usually in a classification task the goal is to predict the class to which the sample is belonging, so we need a classification measure. The first one coming to mind is accuracy but as we all know, accuracy in an imbalanced dataset might not be the best choice. In our problem, we value the positive (non-dominant) class as we are likely to get the more insight from it so we want a relatively high recall. On the other side, we don't want too much false positive as it can skew the insights we can take from the model. To overcome this, we decided to use the ROC curve, PR curve and F1-score. These measures will help us choose the best threshold for classification. While ROC curve is more appropriate for balanced dataset as it can give pretty good results on imbalanced dataset, PR curve is more interesting as it tailored for the detection of rare events and more useful in these scenarios (such as here) and will be our main evaluation measure.

Sampling Another way we tried to overcome this problem is to use sampling. We used and compare three methods : OverSampling, ADASYN and SMOTE. While Oversampling rebalance the training set using multiple times samples from the minority class, ADASYN and SMOTE are synthetic methods as they create new data points that are similar to the one from the minority class. Knowing that, we can start our analysis of the models.

2.1 Champion Model: XGBOOST

XGBOOST is a famous and powerful machine learning algorithm that belongs to the class of gradient boosting methods and first used around 2015. Its popularity comes from its performances and speed. We used XGBOOST as a champion model because of its known efficiency on tabular data, its reputation in machine learning competitions and the feature importance analysis.

To optimize the hyperparameters we ran a GridSearch cross-validation on the following parameters : max depth and numbers of tree. between 6 and 9 for the former and between 300 and 1000 for the latter using F1-score as a metric. We obtained that the best model was a GBM with max_depth = 6 and n_estimators = 500. Note that due to CPU limitation we didn't include parameters such as gamma (minimum loss reduction 0 default), lambda (L2 regularization 1 default) or the eta (learning rate 0.3 default). We obtained the following graphs:

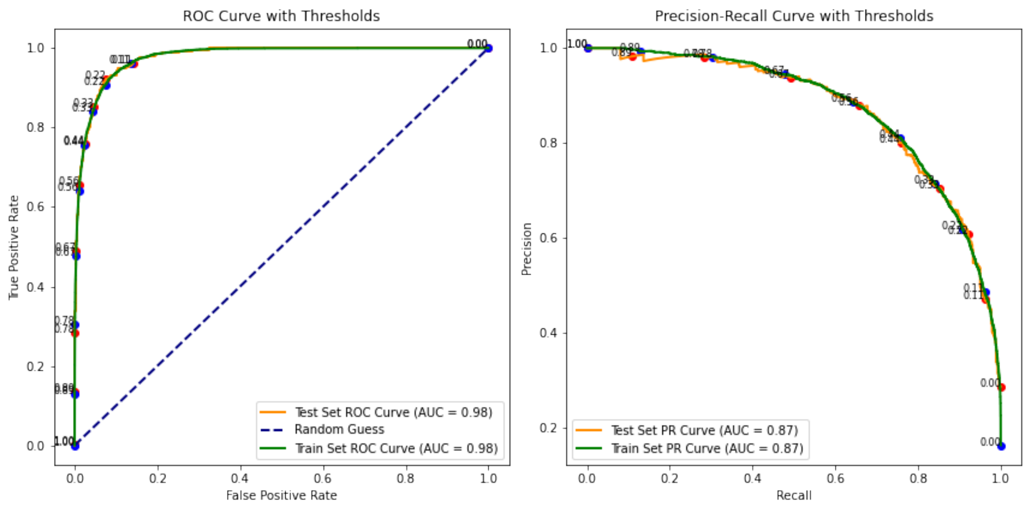


Figure 2.1: ROC curve and PR Curve on train test and test set

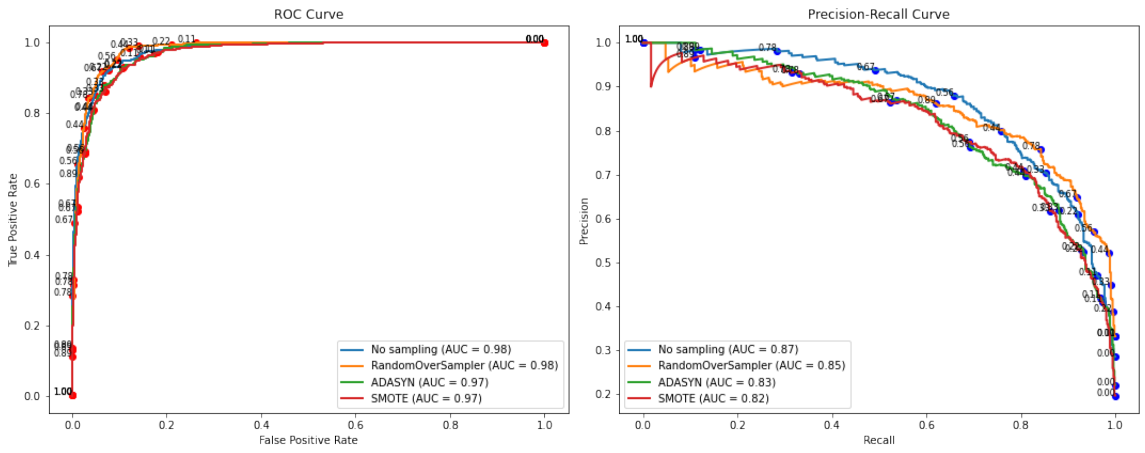


Figure 2.2: ROC curve and PR Curve using different sampling methods

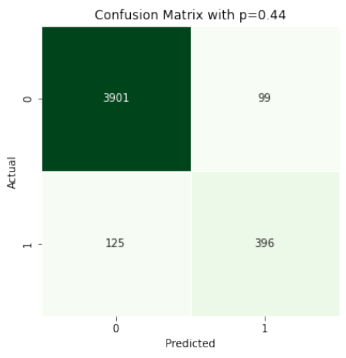
we observe that the performances on the train set and test set seem to be similar and that oversampling methods don't seem to impact much the performances of our model when purely looking at PR AUC. However, it seems that the model that has been trained using the RandomOverSampler (ROS) method has an interesting threshold p. Let's compare the reports and the confusion matrices of base model with $p = 0.44$ and the ROS model with $p = 0.78$

	precision	recall	f1-score	support
0	0.968952	0.975250	0.972091	4000.000000
1	0.800000	0.760077	0.779528	521.000000
accuracy			0.950453	0.950453
macro avg	0.884476	0.867663	0.875809	4521.000000
weighted avg	0.949482	0.950453	0.949900	4521.000000

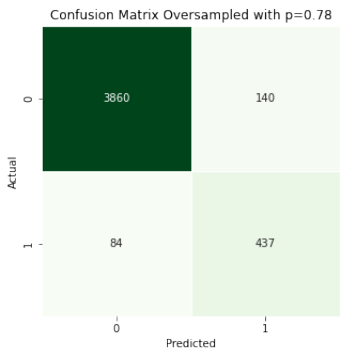
Table 2.1: Classification Report of XGBOOST without sampling method and $p = 0.44$

	precision	recall	f1-score	support
0	0.978702	0.965000	0.971803	4000.000000
1	0.757366	0.838772	0.795993	521.000000
accuracy			0.950453	0.950453
macro avg	0.868034	0.901886	0.883898	4521.000000
weighted avg	0.953195	0.950453	0.951542	4521.000000

Table 2.2: Classification Report of XGBOOST with ROS sampling and $p = 0.78$



(a) Confusion Matrix without sampling



(b) Confusion Matrix with ROS sampling

Figure 2.3: Confusion Matrices of XGBOOST

Looking at the classification report and the matrices, we observe that despite having a PR AUC lower than the one without sampling, using a threshold of 0.78 seems to bring a higher f1-score. Moreover, we can see that we traded some precision (from 0.8 to 0.76) for some recall (0.76 to 0.84) and if we look closer at the confusion matrices, it confirms that we traded some false negative for some false positive. So, which one is better ? Before answering this question let's take a look at the challenger models.

2.2 Challenger Model: Logistic Regression

Logistic Regression is a linear model used in classification tasks that is one if not the most famous one. As it is simple, interpretable and quick to implement, it is also used as a benchmark. Let's see how it performs.

For this model we tried two different approaches in order to train it. On both of them we tried to optimize the "class_weight" parameter. One with a grid search cross validation and the other using the preset "balanced" of sklearn API, all of them using L2 regularization. Here are the results we get :

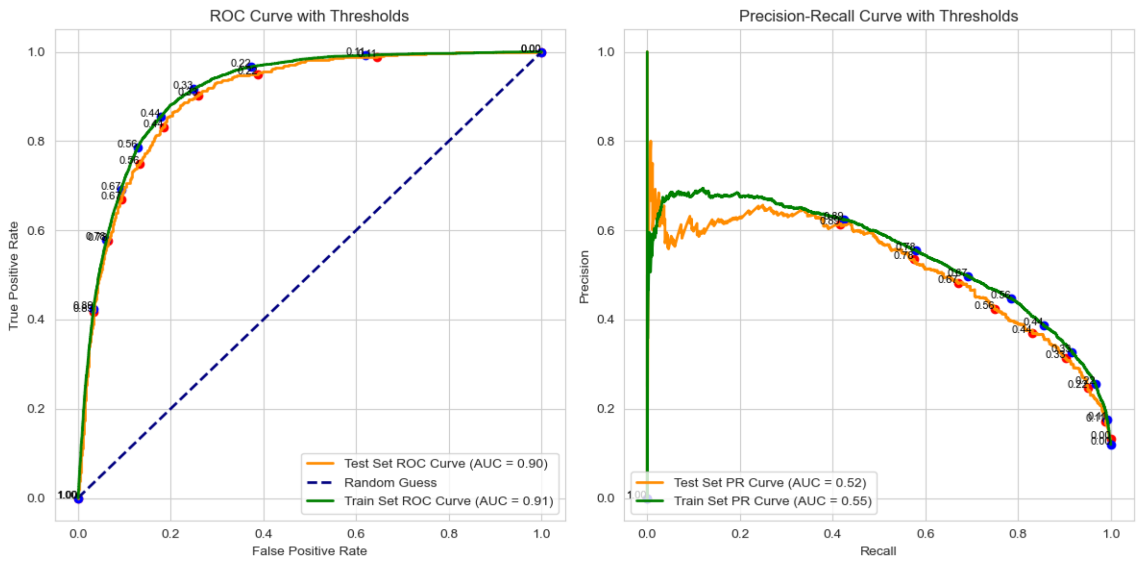


Figure 2.4: ROC curve and PR Curve on train test and test set with class_weight='balanced'

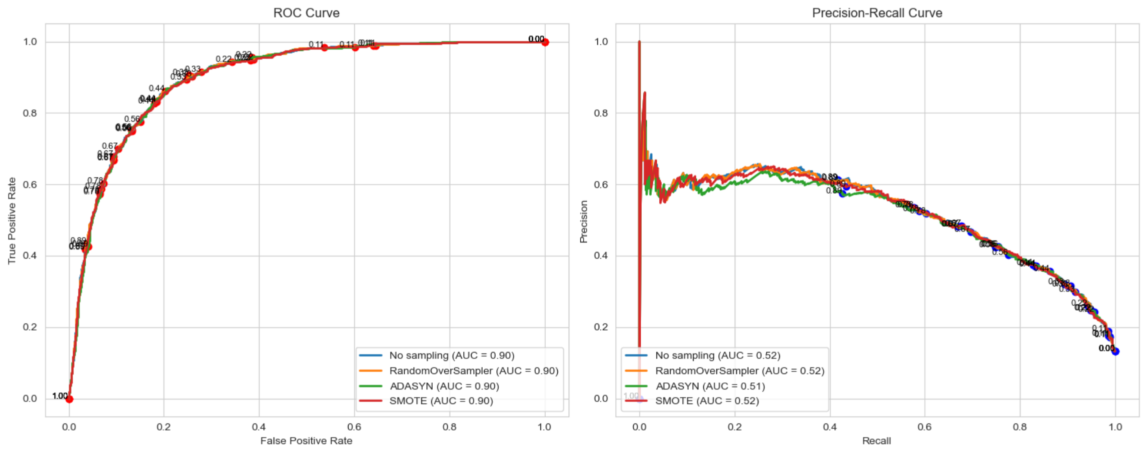


Figure 2.5: ROC curve and PR Curve with different sampling methods with class_weight='balanced' using oversampling

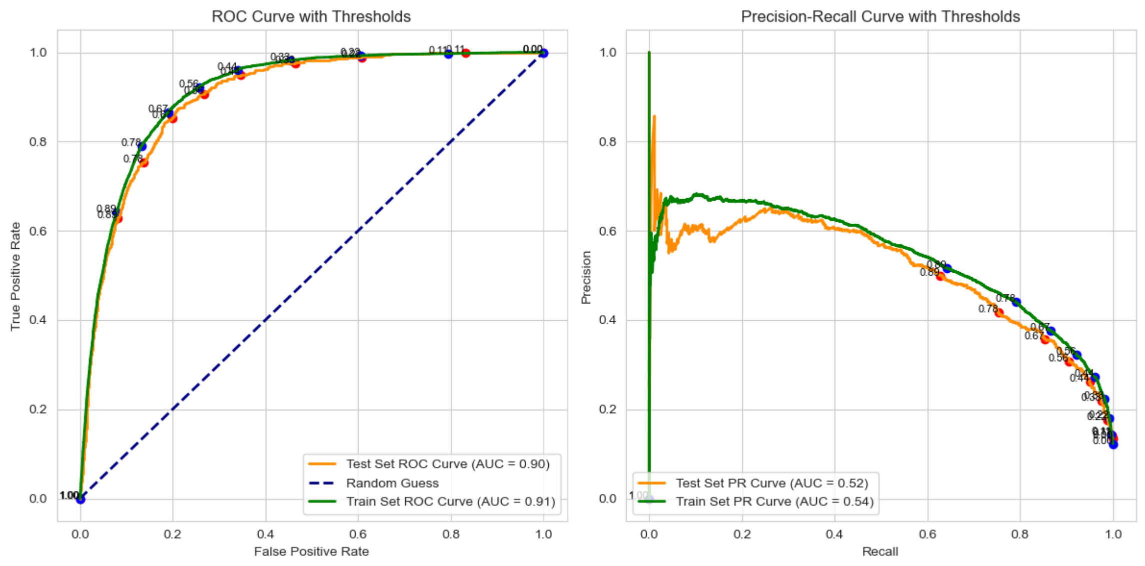


Figure 2.6: ROC curve and PR Curve on train test and test set with grid search CV on class_weight

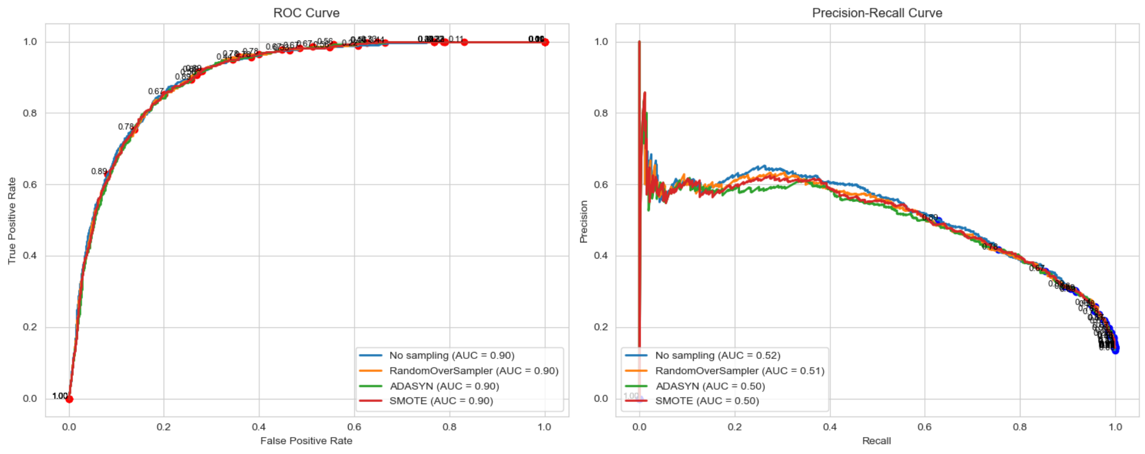


Figure 2.7: ROC curve and PR Curve with different sampling methods with grid search CV on class_weight

As we can see, the results are quite similar. The PR AUC score is smaller than our champion model but considering that a random 1/2 model would have a PR AUC converging to the ratio of the minority class, we can consider it as an interesting model.

Now, lets look at both of the classification report an confusion matrices. As there is no significant improvement between models trained on sampled and original dataset, only the latter will be shown. Regarding the threshold, the PR AUC curve do not give us interesting threshold so I decided to set it at 0.5.

	precision	recall	f1-score	support
0	0.987606	0.697250	0.817409	4000.000000
1	0.286388	0.932821	0.438233	521.000000
accuracy			0.724397	0.724397
macro avg	0.636997	0.815036	0.627821	4521.000000
weighted avg	0.906798	0.724397	0.773713	4521.000000

Table 2.3: Classification Report for the manually tuned Logistic Regression

	precision	recall	f1-score	support
0	0.968885	0.840750	0.900281	4000.000000
1	0.393333	0.792706	0.525780	521.000000
accuracy			0.835213	0.835213
macro avg	0.681109	0.816728	0.713030	4521.000000
weighted avg	0.902558	0.835213	0.857124	4521.000000

Table 2.5: Classification Report for the automatically tuned Logistic Regression

	precision	recall	f1-score	support
0	0.918391	0.976250	0.946437	4000.000000
1	0.646840	0.333973	0.440506	521.000000
accuracy			0.902234	0.902234
macro avg	0.782616	0.655112	0.693472	4521.000000
weighted avg	0.887098	0.902234	0.888134	4521.000000

Table 2.4: Classification Report for the base Logistic Regression model

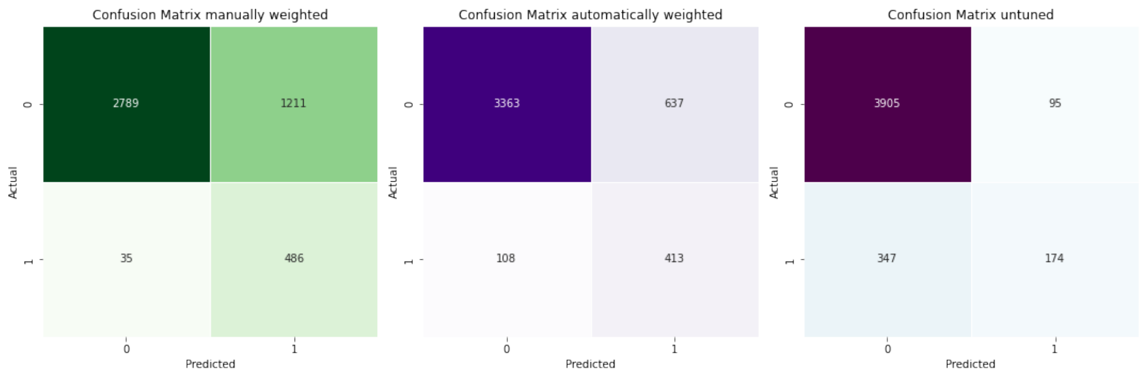


Figure 2.8: Confusion Matrices of Logistic Regression Models

As we can see, for the manually tuned model, even though we have a high recall, the precision seems low to be considered a challenger. Despite having misclassed only 35 of the true positives, the model seems to be overly optimistic. The optimistic view can be detrimental to the business, as the call center is likely to spend much more time contacting customers who are not interested rather than the other way around. Same thing can be said about the second one, there are still too much false positives. The base model, on the other way around, has way too much false negative which leads to a ridiculously low recall. However, if I were to compare the three, I would choose this one considering what is at stake at a business level.

2.3 Challenger Model: Random Forest

Random Forest is a powerful and versatile machine learning algorithm known for its robust performance. It belongs to the ensemble learning category, aggregating decision trees for accurate predictions. Widely used for handling diverse data types, Random Forest excels in mitigating overfitting and capturing complex relationships within datasets. Its effectiveness with tabular data and feature importance analysis makes it a preferred choice in various applications.

To optimize the hyperparameters we ran a GridSearch cross-validation on 3 folds on the following parameters : max depth,numbers of tree, min_samples_split and min_samples_leaf. Here is the table used for the grid search.

Parameter	**Values**
n_estimators	[1500, 1750, 2000, 2250]
max_depth	[10, 20]
min_samples_split	[6, 7, 8]
min_samples_leaf	[2, 4, 6, 7]

After running the search, we found that the best model had the following parameters: {max_depth: 20, min_samples_leaf: 2, min_samples_split: 7, n_estimators: 2000}. Now let's take a look at the ROC curve and PR curve for a Random Forest trained on the original dataset and one trained on a dataset with sampling methods.

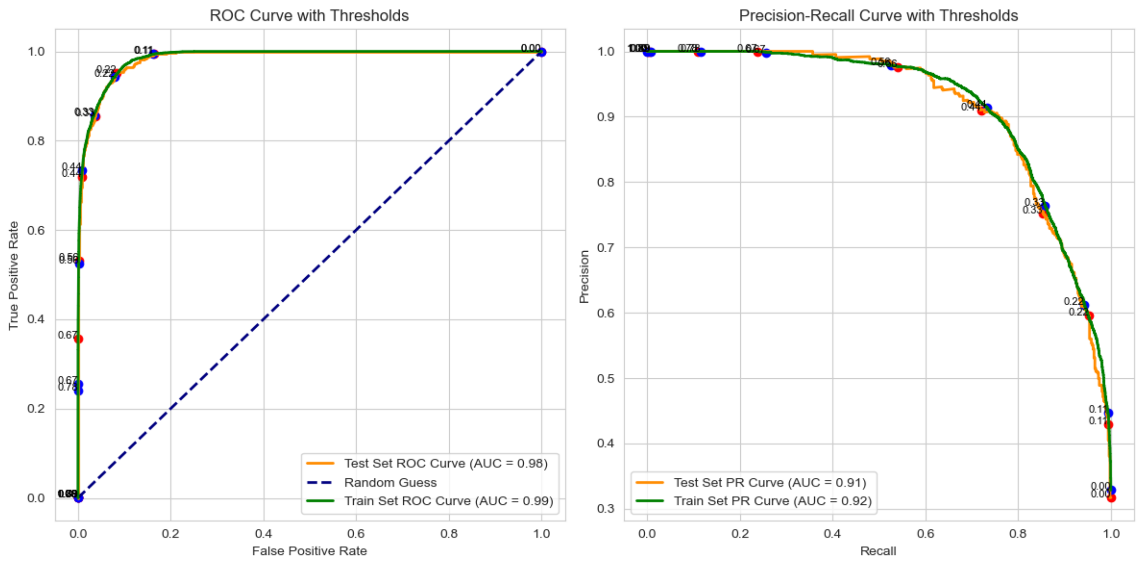


Figure 2.9: ROC curve and PR Curve on train test and test set

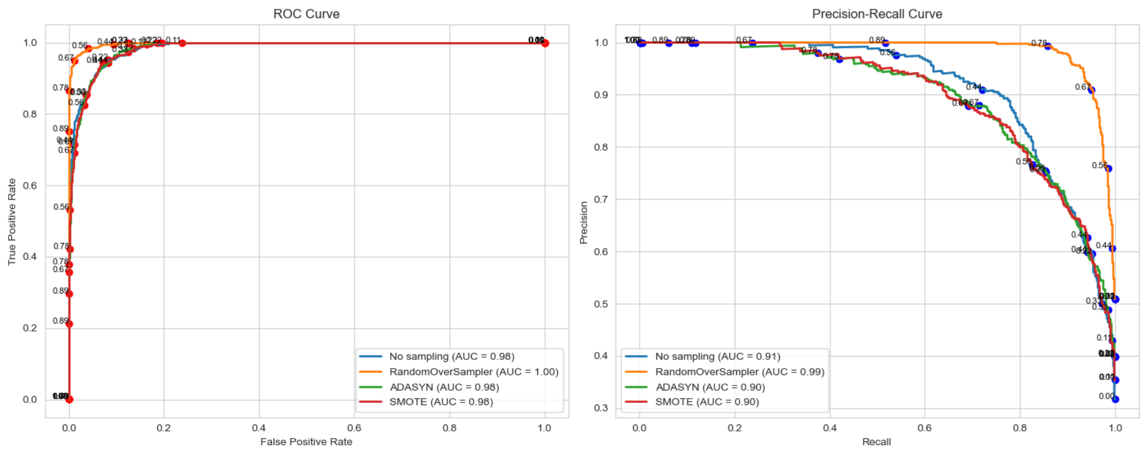


Figure 2.10: ROC curve and PR Curve using different sampling methods

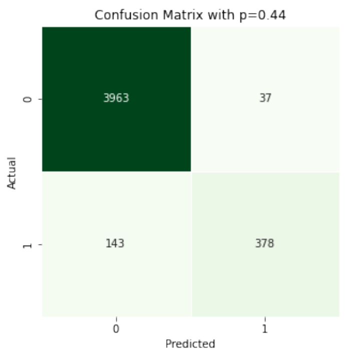
Here we have very interesting results. Compared to the two models above, on this one there is a significant difference when using a sampling methode, especially the RandomOverSampler. in terms of PR AUC there is a jump from 0.90 to 0.99 which is almost a 10% increase. Moreover, using this plot choosing a threshold at 0.67. Let's compare the reports and confusion matrices of both model, the base one using a threshold of 0.44 and the sampled one using a threshold of 0.67.

	precision	recall	f1-score	support
0	0.965173	0.990750	0.977794	4000.000000
1	0.910843	0.725528	0.807692	521.000000
accuracy			0.960186	0.960186
macro avg	0.938008	0.858139	0.892743	4521.000000
weighted avg	0.958912	0.960186	0.958192	4521.000000

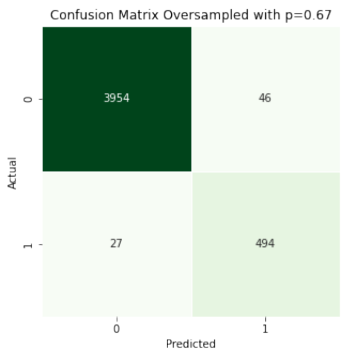
Table 2.6: Classification Report of Random Forest without using sample and p=0.44

	precision	recall	f1-score	support
0	0.993218	0.988500	0.990853	4000.000000
1	0.914815	0.948177	0.931197	521.000000
accuracy			0.983853	0.983853
macro avg	0.954016	0.968338	0.961025	4521.000000
weighted avg	0.984183	0.983853	0.983978	4521.000000

Table 2.7: Classification Report of Random Forest with RandomOverSampler and p=0.67



(a) Confusion Matrix without sampling p=0.44



(b) Confusion Matrix with ROS sampling p=0.67

Figure 2.11: Confusion Matrices of RandomForest

As we can see on the report and the matrices, there is a clear improvement in the performances of the model. The recall for the minority class goes from 0.72 to 0.94 whereas its precision only decreases by 0.05. As reflected in the confusion matrix, the number of false negative has been reduced by almost 80% and the number of true positive jumped from 378 to 494, which is a huge improvement. On this model, we can see how the choice of a sampler can impact the results but also how choosing p is important. Had I let p=0.5 (as it is the case in sklearn when using the *predict* function, it would have resulted in a significantly more underperforming model compared to the one we currently have.

2.4 Challenger Model: Multi-Layer Perceptron

A fully connected multi-layer neural network is called a Multilayer Perceptron. It has input and output layers, and one or more hidden layers with many neurons stacked together. And while in the Perceptron the neuron must have an activation function that imposes a threshold, like ReLU or sigmoid, neurons in a Multilayer Perceptron can use any arbitrary activation function. One can find here more information regarding the history about neural networks.

To optimize the hyperparameters we ran a GridSearch cross-validation on 5 folds on the following parameters : hidden_layer_sizes and activation. Here is the table used for the grid search.

Parameter	**Values**
hidden_layer_sizes	[(50,), (100,), (50,50,), (25,25,)]
activation	[tanh, relu]

For the neural network, we tried different sizes of network with 1 and 2 layers. Interestingly, on this dataset, the tanh activation function performed significantly better compared to the relu activation function. It may have to do with the non linearity of the dataset and the small number of layers.

After running the search, we found that the best model had the following parameters: {hidden_layer_sizes: (25,25,), activation: tanh}. Now let's take a look at the ROC curve and PR curve for a Random Forest trained on the original dataset and one trained on a dataset with sampling methods.

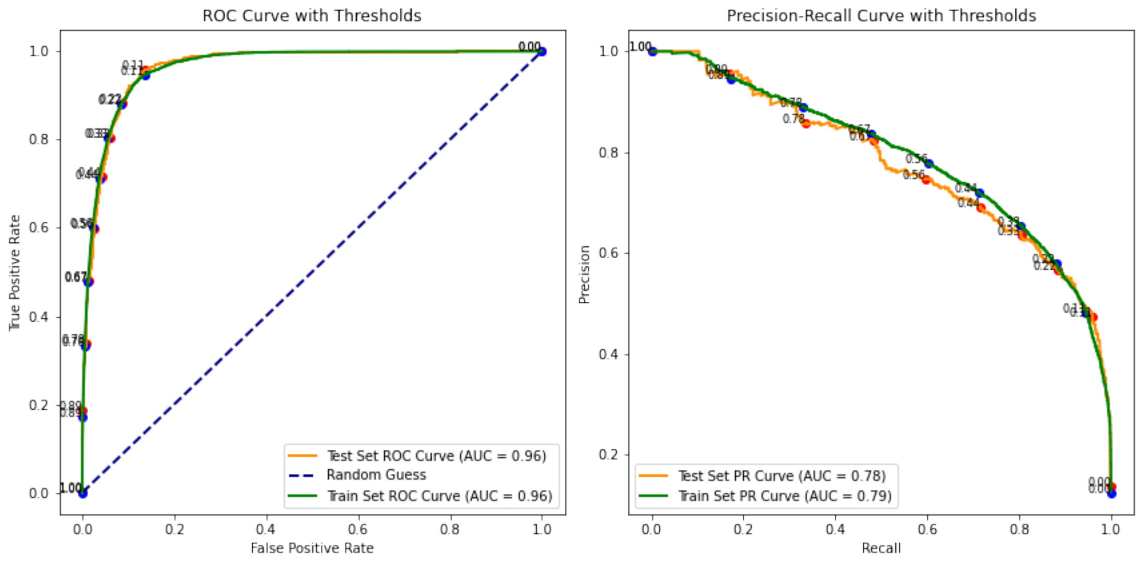


Figure 2.12: ROC curve and PR Curve on train test and test set

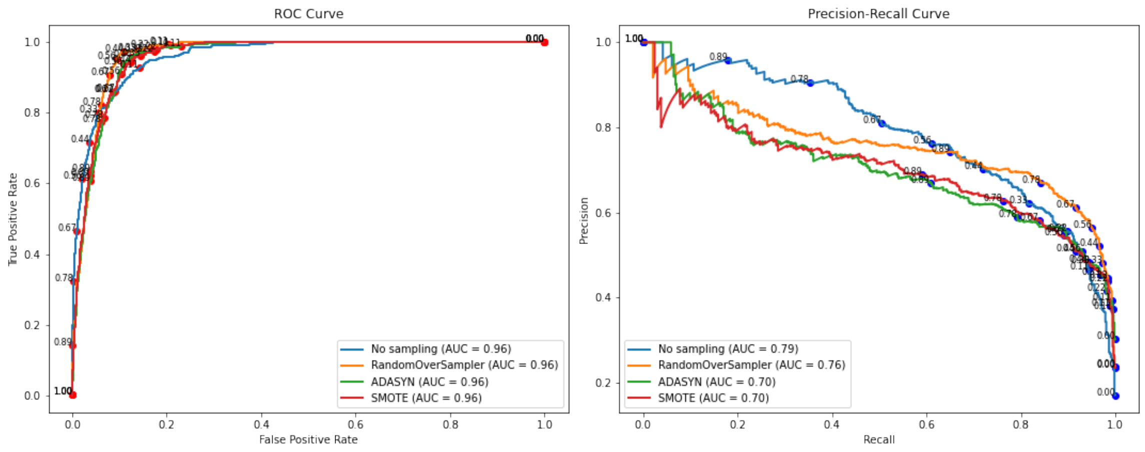


Figure 2.13: ROC curve and PR Curve using different sampling methods

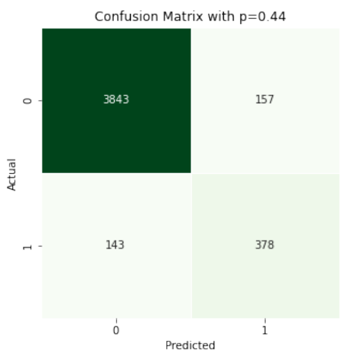
On this model and these parameters, we observe a similar behavior with the XGBOOST model but somewhat less performant. Despite the Over Sampling methods not showing convincing results overall, we can observe that as in the boosting model if we take p=0.78. Let's look again at both the reports and the confusion matrices.

	precision	recall	f1-score	support
0	0.965621	0.962000	0.963807	4000.000000
1	0.716418	0.737044	0.726585	521.000000
accuracy	0.936076	0.936076	0.936076	0.936076
macro avg	0.841019	0.849522	0.845196	4521.000000
weighted avg	0.936903	0.936076	0.936470	4521.000000

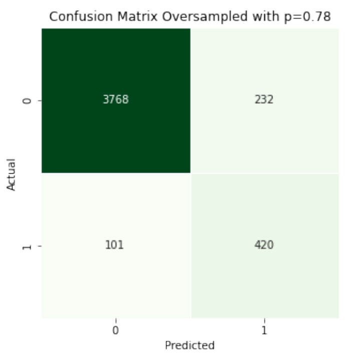
Table 2.8: Classification Report of MLP without using sample and p=0.44

	precision	recall	f1-score	support
0	0.973895	0.942000	0.957682	4000.000000
1	0.644172	0.806142	0.716113	521.000000
accuracy	0.926344	0.926344	0.926344	0.926344
macro avg	0.809033	0.874071	0.836897	4521.000000
weighted avg	0.935898	0.926344	0.929844	4521.000000

Table 2.9: Classification Report of MLP with RandomOverSampler and p=0.78



(a) Confusion Matrix without sampling p=0.44



(b) Confusion Matrix with ROS sampling p=0.78

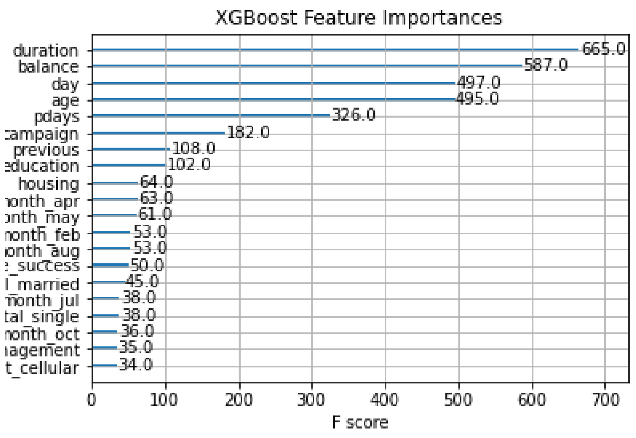
Figure 2.14: Confusion Matrices of RandomForest

We have here two models with similar performances, however, this similarity might not translate in similar performances business-wise. Here we trade some precision for recall but the gain doesn't seem to be significant to justify it. Is it worth detecting correctly

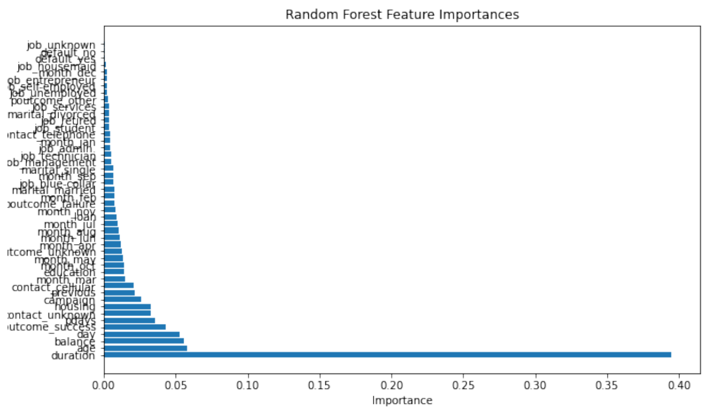
50 more customers but having 100 more false positive ? This issue will be discussed in chapter 3.

2.5 Feature Importances

As seen in the boosting model and Random Forest model, they seem to be quite convincing. So if one wants to go a step further, a feature importance analysis can be done. Find below the results:



(a) Feature Importances of XGBOOST



(b) Feature Importances of Random Forest

Figure 2.15: Feature Importances

Chapter 3

Results Discussion

Now that we have presented all of the results for the champion model and the three challengers, discussing and choosing the right one for our problem can be considered the most important part of the decision-making process. Now, we need to carefully assess performance metrics, consider our problem's specific needs, and find the right balance between accuracy and generalization for practical success.

If we look back at Chapter 2 we had one outstanding model and two adequate ones which are the Random Forest OverSampler and the XGBOOST base and OverSampler. When it comes to binary classification as it is the case here, one is always going to face tradeoff regarding various metrics. This is why in these cases understanding the dataset and the business goal is important. The base XGBOOST model had a nice PR AUC score of .87 and when observing the classification report ($p = 0.44$) we could see a f1-score of .78 for the minority class. At first glance, it looks nice but when investigation further we notice that the recall score for the minority class is only 0.76 which means that only 76% of the relevant items are retrieved. On the other way, with the XGBOOST ROS model ($p = 0.78$) we traded off 5% of precision for 8% of recall totalling a f1-score of .80 for the minority class. Here, despite having a smaller precision, here, almost 84% of the relevant items are retrieved, which in our case is a huge improvement. When we look at both Logistic Regression and MLP models we made, one can think that the results we have gotten are subpar. There can be multiple reasons why this was the case. Here, we made relatively simple models, hence, due to the complexity and high dimensionality of the dataset (40 features) despite regularization it might not be able to capture intricate relationships and patterns effectively, leading to sub-optimal model performance and can be also due to multicollinearity present in the dataset. Also, we should not exclude the possibility that the dataset has been overfitted which can happen when trying to optimize for the hyperparameters, which is why we used cross-validation to avoid this.

However if we take a step back, the fact that the dataset was imbalanced is a hurdle that can be quite difficult to overcome. Thankfully, the dataset being as it is, it is highly unlikely that it has been poisoned as the outcome for each sample shouldn't have reasons to be falsified.

However, one should take in account the stationarity of the dataset. Here, we have been given 45,000 rows worth of information but one should not exclude the possibility that the most meaningful information to assess the subscription rate to a CD isn't present in the dataset. For example, one of the main argument of a CD is the yield. If the yield is 10%, a customer is how much more likely to subscribe compared to a 1% yield? In other words, what if all of the subscribing clients were given high yields ? As the yield of a CD is dependent of the state of the economy (which changes all the time), is this information reflected in the dataset ? would it make all of our information irrelevant ? I don't know for sure, but in case there is a significant gap between the first and the last call this could change the dynamic.

If we discuss the results from a business point of view, the research we made gave us interesting insights. Let's say we choose to use the ROS Random Forest model. Looking at the feature importances plot in 2.15 duration seems to be the most important feature by some distance. However, this is a data point not known at the time of the call. Then we would have to focus our search on customers fulfilling the next conditions. When implementing this model to production, one should look at what improvement it can bring to the current workflow and efficiency. Here, even if the model reaches 0.95 recall score on the minority class using back-testing, it is way more important to forward-test our model to really assess the behavior of our solution. In a real-time scenario, the trade-off between losing potential clients due to a delayed response and the relatively minor inconvenience of spending an extra 2 minutes with an unsuccessful outcome needs to be carefully considered. This trade-off could significantly enhance the number of subscription by improving the targeting as what we wanted at the beginning. And by increasing the conversion rates will eventually contribute to higher revenue overall, hence the focus on improving the recall in our model. This model can bring higher efficiency and success rate while spending less time on call with customers. Moreover, this solution can be also used in all departments of the retail bank, for example the digital marketing department, which could use this tool on the internet targeting even more specifically the customers.

To conclude, machine learning helped us finding new insights about clients more likely to subscribe to a term deposit and despite having some convincing results, further analysis regarding the training of the models without incorporating features known post-hoc and also, improving the data processing might also improve the results.

thanks for reading and happy new year.

Bibliography

[1] Léo Grinsztajn, Edouard Oyallon, and Gaël Varoquaux. *Why do tree-based models still outperform deep learning on tabular data?* 2022. arXiv: `2207.08815 [cs.LG]`.