Analysis of a Bank's Marketing Dataset
Springboard DSC Capstone Project I
By: Anshul Gupta
September 2017

Summary

In this capstone project, I analyze the data associated with direct marketing campaigns of a Portuguese banking institution. The Dataset is available from https://archive.ics.uci.edu/ml/datasets/bank+marketing . The marketing campaigns were collected through phone calls with the purpose of recognizing potential customers interested in buying products or services from the bank. Given the size of the bank's customer base, it is neither practical nor cost-efficient for the bank to contact every customer in order to separate good prospects from the rest. I have modeled this problem as a Machine Learning binary classification problem where I have a dependent variable **'y'** which can be categorized as 'yes' or 'no' based on certain attributes. An affirmative label signifies that a customer is ready to do business with the bank, and a negative label signifies the opposite. Since the proportion of positive-to-negative labels is strongly skewed towards the latter, one says this classification problem is imbalanced. In this capstone project, I first define a baseline using models that are oblivious to the imbalance in the classes, and then use models that address it explicitly. We observed that the models that address the imbalance explicitly perform considerable better than their counterparts, with respect to a performance metric that captures the business needs—namely, the true positive rate.

Introduction

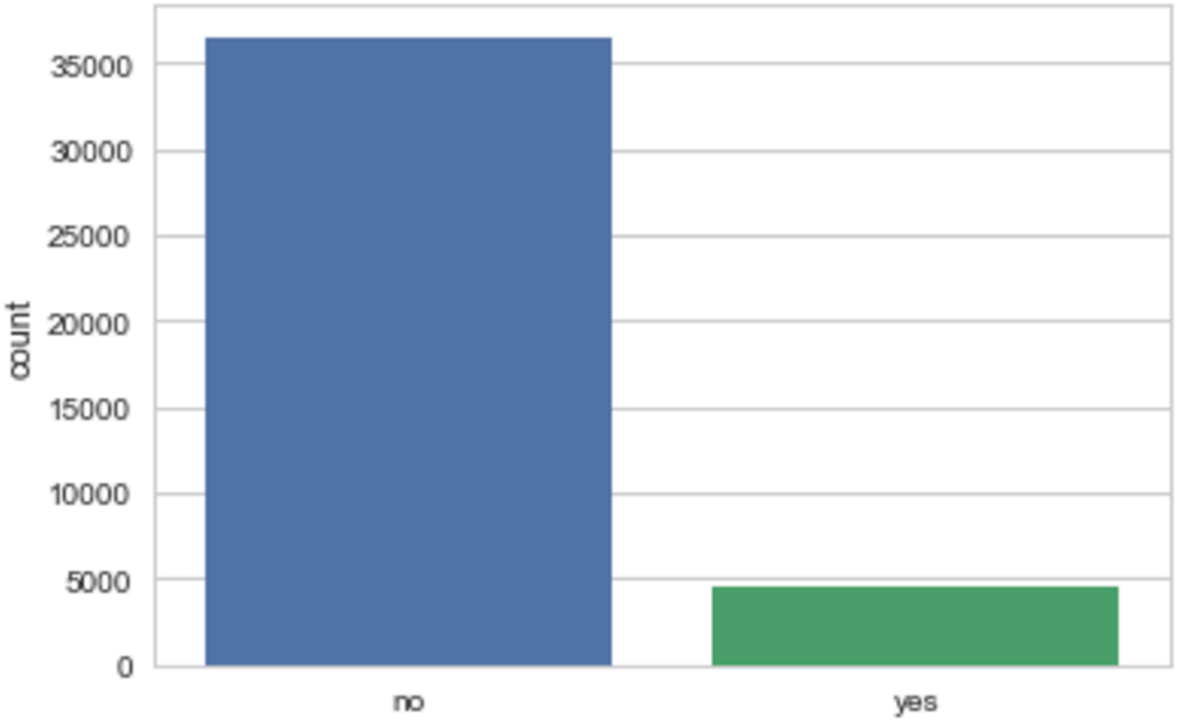
The bank's marketing dataset has about 40K+ instances, 20 features, and a binary response variable. Using this dataset, I have implemented various Machine Learning Binary Classification Models to predict whether a client will subscribe to the product being offered by the bank, or not. The table below summarizes the fields in the dataset, and their types.

Dataset Field	Type
Age	Numeric (Integer)
Job	Categorical
Marital	Categorical
Education	Categorical
Default (Indicate whether the client has credit in default)	Categorical
Housing	Categorical
Loan	Categorical
Contact	Categorical
Month	Categorical
Day of the week	Numeric (Integer)
Duration	Numeric (Integer)
Campaign	Numeric (Integer)
PDays (Number Of Days passed by since the client was last contacted in a previous campaign)	Numeric (Integer)
Previous (Number of contacts performed before this campaign for this client)	Numeric (Integer)
POutcome (Outcome of previous marketing campaign)	Categorical
Emp.var.price (Quarterly Employee variation rate)	Numeric (Decimal)
Cons.price.idx (Monthly consumer price index)	Numeric (Decimal)
Cons.conf.idx (Montly consumer confidence index)	Numeric (Decimal)

Euribor3m	Numeric (Decimal)
Nr.Employed	Numeric(Decimal)
Y	Categorical

The dataset contains 36548 negative labels and 4640 positive labels. From the perspective of the business problem, a positive label is vital as these are associated with customers who have decided to subscribe to offered bank products. Given the marked difference between positive and negative labels in the dataset, this type of classification problem is called “imbalanced”.

The chart below graphically compares the number of positive (“yes”) and negative (“no”) labels.



The evaluation metrics I used to evaluate the performance of the models I tried are True Positive Rate (a.k.a. Sensitivity or TPR) and True Negative Rate (a.k.a. Specificity or TNR), since the total accuracy score in itself does not give an accurate judgement of our model as the response variable is highly imbalanced.

From a business perspective, a higher TPR is preferred, even at the expense of a modest TNR. These metrics are defined as follows:

$$TPR = TP/(TP + FN)$$
$$TNR = TN/(TN + FP)$$

Where TP stands for True Positive, and TN stands for True Negative. Notice that (TP + FN) is the total number of positive labels, and (TN + FP) is the total number of negative labels. Hence, TPR is the fraction of positive labels that are correctly classified; while TNR is the fraction of negative labels that are correctly classified.

Modeling the Business Problem

The goal of the business problem is to predict if a customer is going to peruse the banking service or not. There are about 21 attributes out of which Age, Job, Marital Status, Education, Default, Housing and Loan are the customer features. Contact, Month, Day Of Week and Phone Call Duration are data recorded by telemarketing campaign based phone calls. Employment Variation Rate, Consumer Price Index, and Consumer Confidence index are the social economic factors included in the Dataset . The response variable has categorical values 'yes' or 'no'. This business problem is modeled as a machine learning classification problem. Therefore, the data science problem is to build a binary classifier that can be used to assist the clients in addressing the business problem.

Approach

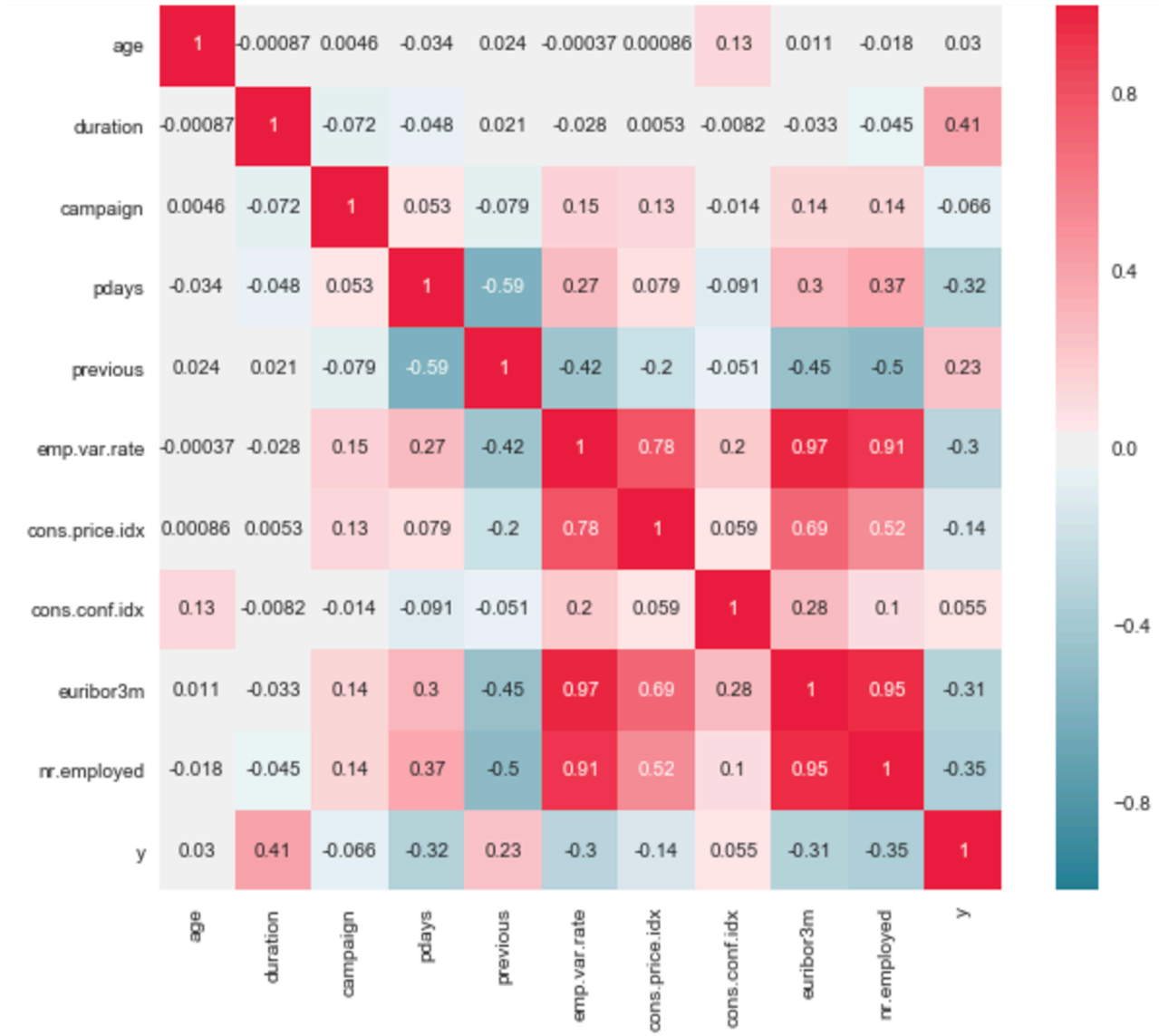
In this section I discuss how I implemented the wrangling, exploration, and analysis of the dataset associated with this project.

Data Wrangling and Exploration

There are about 10 features which have categorical values, which were "one-hot encoded" by introducing multiple "dummy variables" that take numeric values (either 0 or 1). For that, Pandas provide a function called pd.get_dummies to one-hot encode the categorical features to fit with various classification and regression problems. After performing this transformation, the number of attributes increased from 21 to 64.

I examined the potential correlation between pairs of attributes to determine any positive or negative correlation with respect to response variable ‘y’. Correlation describes the degree of linear relationship between pairs of variables.

Following correlation chart shows correlation of numerical columns with response variable and among themselves. The diagram illustrates no particularly strong correlation with respect to the response variable.

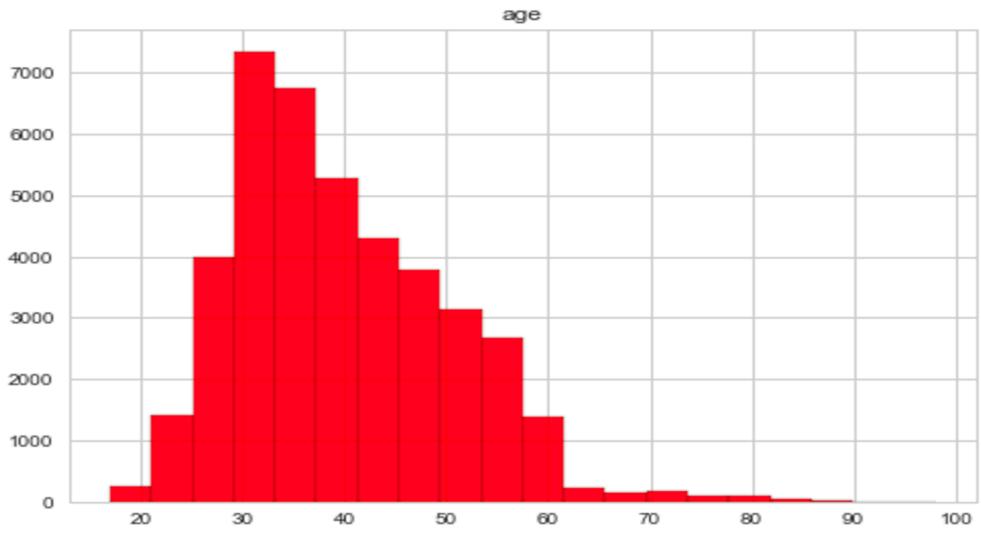


The Data contains no missing values hence there was no need to impute missing values. The following tables shows statistics for numerical columns.

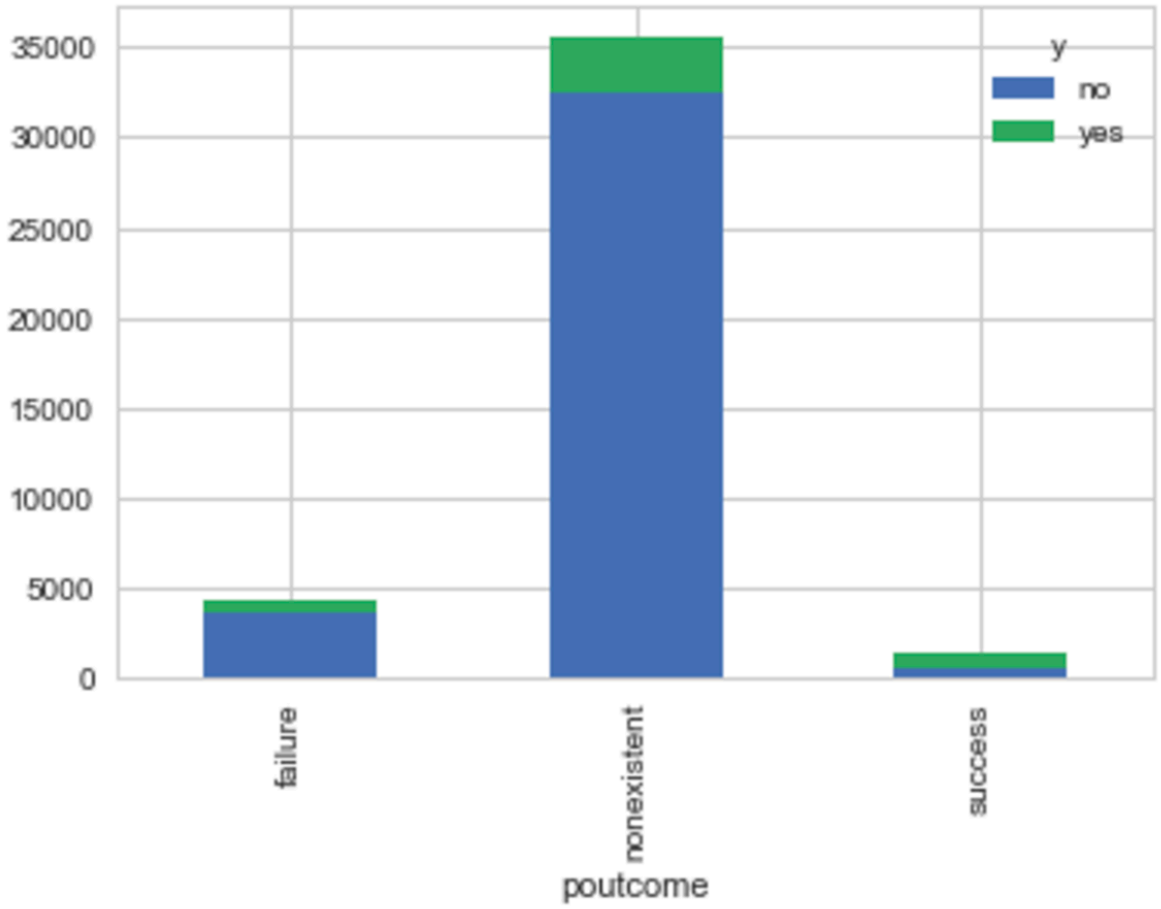
	Age	Duration	Campaign	Pdays	Previous
count	41188	41188	41188	41188	41188
mean	40.02	258.29	2.58	962.48	0.17
std	10.42	259.28	2.77	186.91	0.49
min	17	0	1	0	0
25%	32	102	1	999	0
50%	38	180	2	999	0
75%	47	319	3	999	0
max	98	4918	56	999	7

	Emp.var.rate	Cons.price.idx	Cons.conf.idx	Euribor3m	Nr.Employed
count	41188	41188	41188	41188	41188
mean	0.08	93.57	-40.50	3.62	5167.03
std	1.57	0.57	4.63	1.73	72.25
min	-3.4	92.20	-50.8	0.634	4963
25%	-1.8	93.07	-42.70	1.34	5099.1
50%	1.10	93.749	-41.8	4.857	5191
75%	1.4	93.99	-36.4	4.96	5228.1
max	1.4	94.77	-26.9	5.04	5228.1

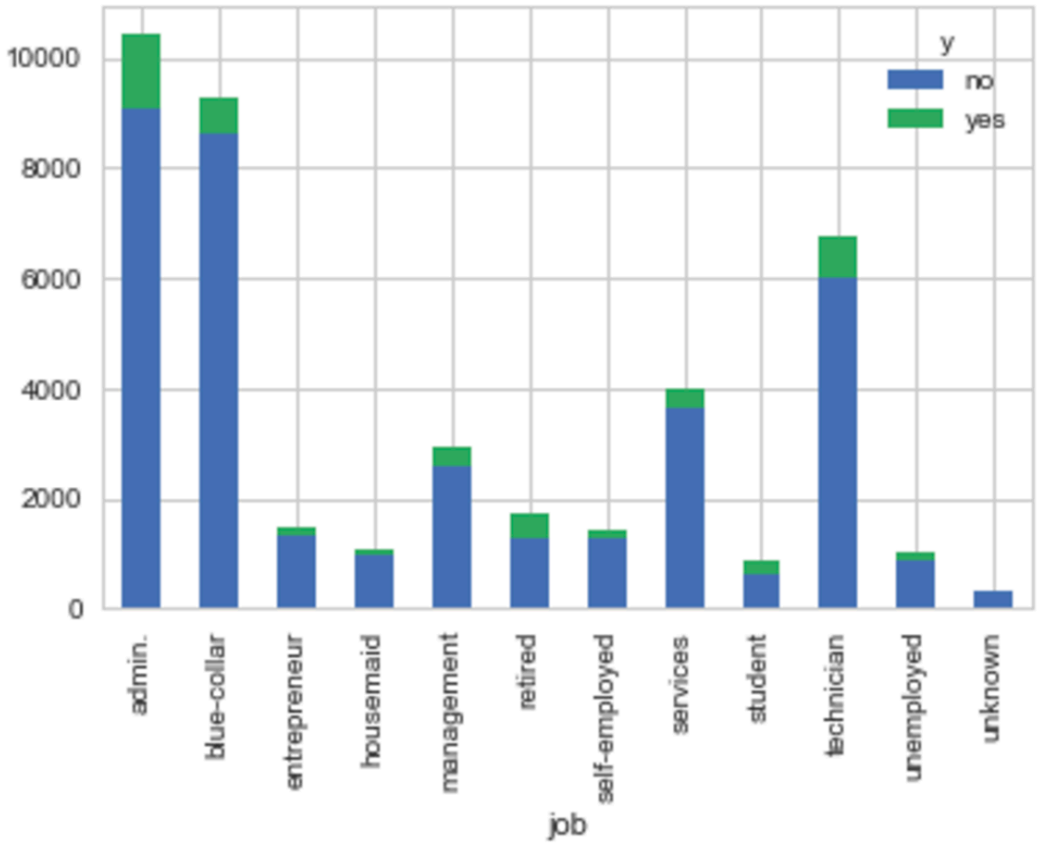
The following histogram shows that age group 30-40 shows maximum number of customers in the Dataset.



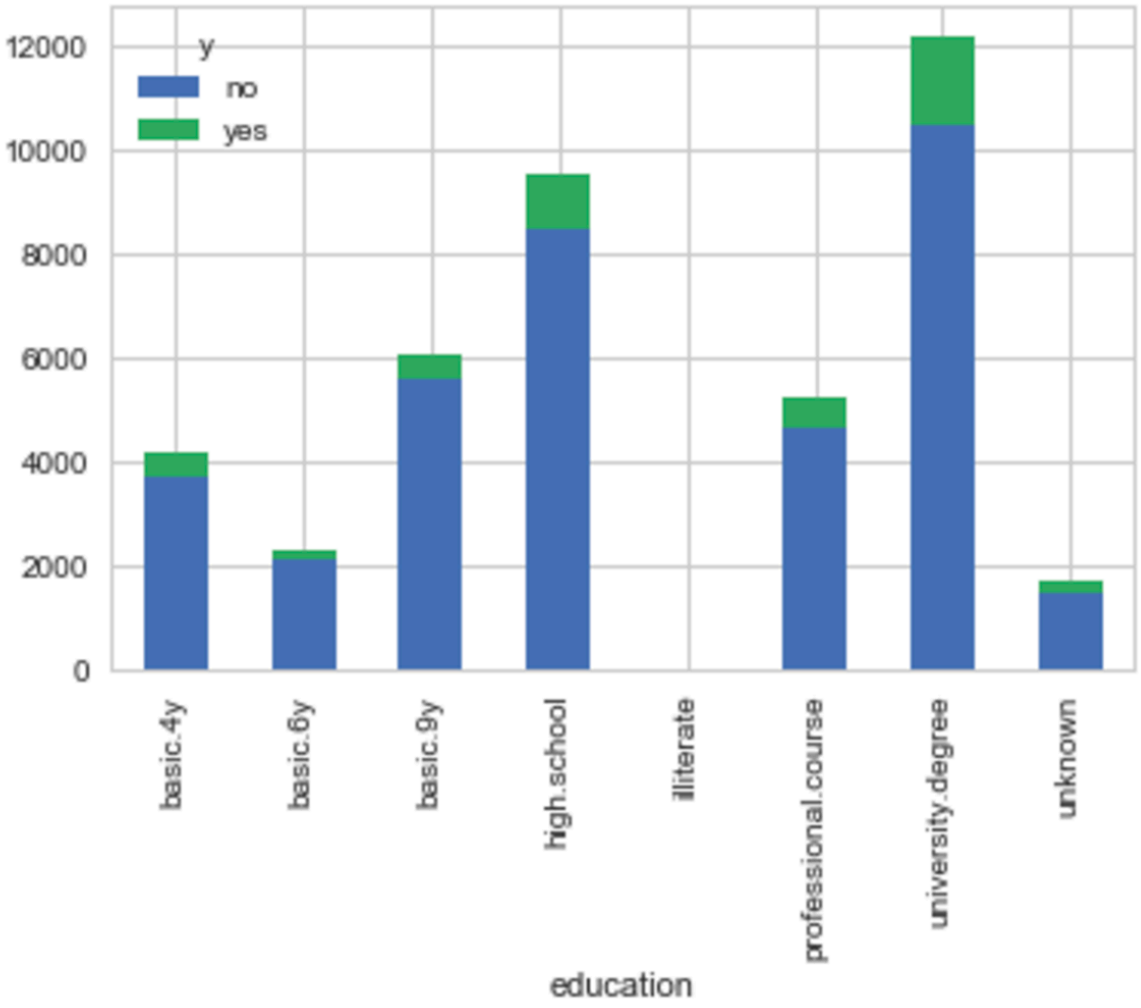
The following plot shows the distribution of values for the variable associated with previous outcome.



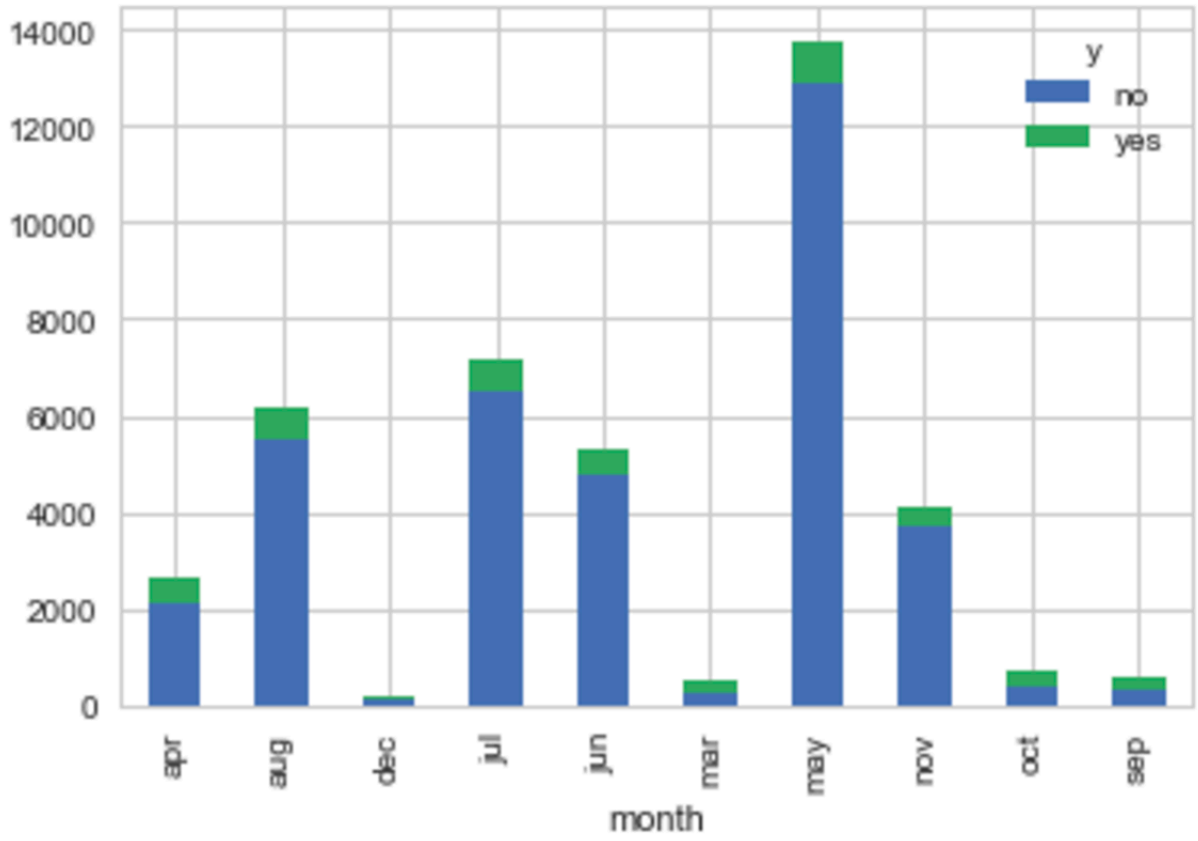
The three top jobs people were contacted about were Admin, Blue-Collar and technician which comprise more than half of the total people contacted by the Telemarketing company.



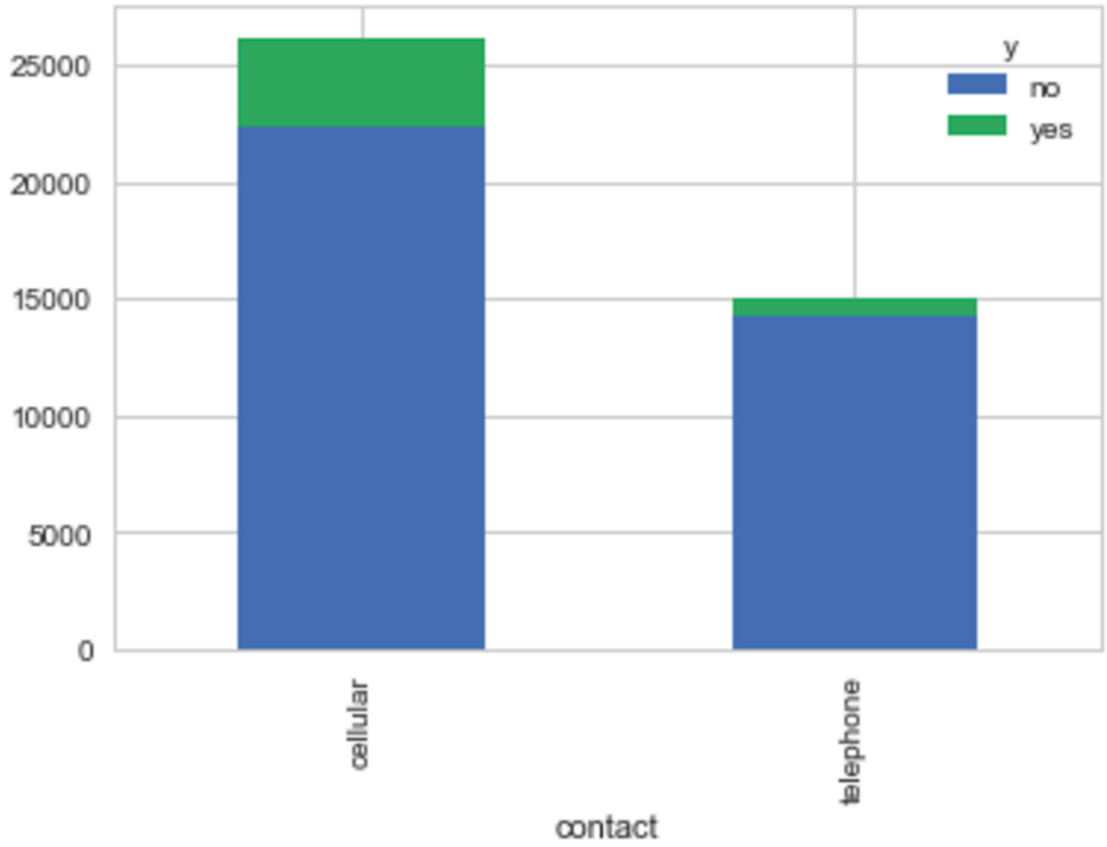
The following plot shows that people with university degrees were contacted the most.



The following plot shows that the month of May was when maximum calls were made which is almost double than second highest month that is July.

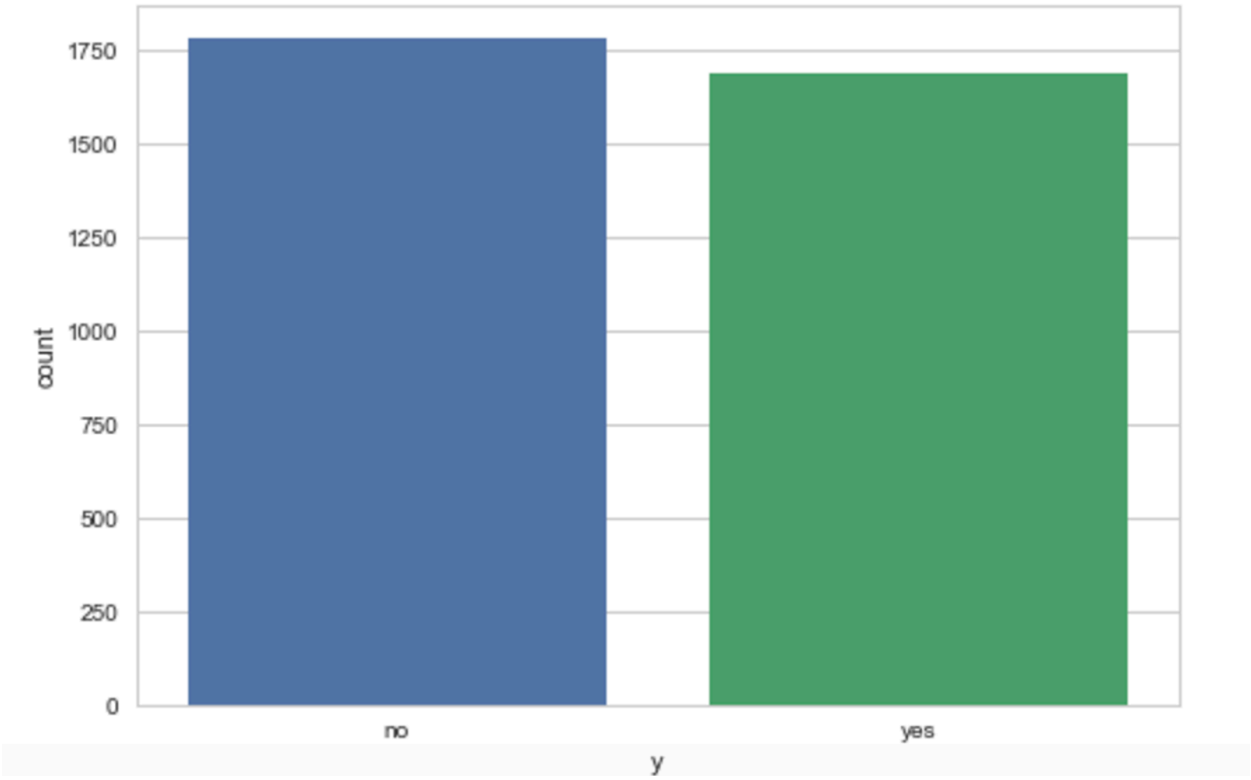


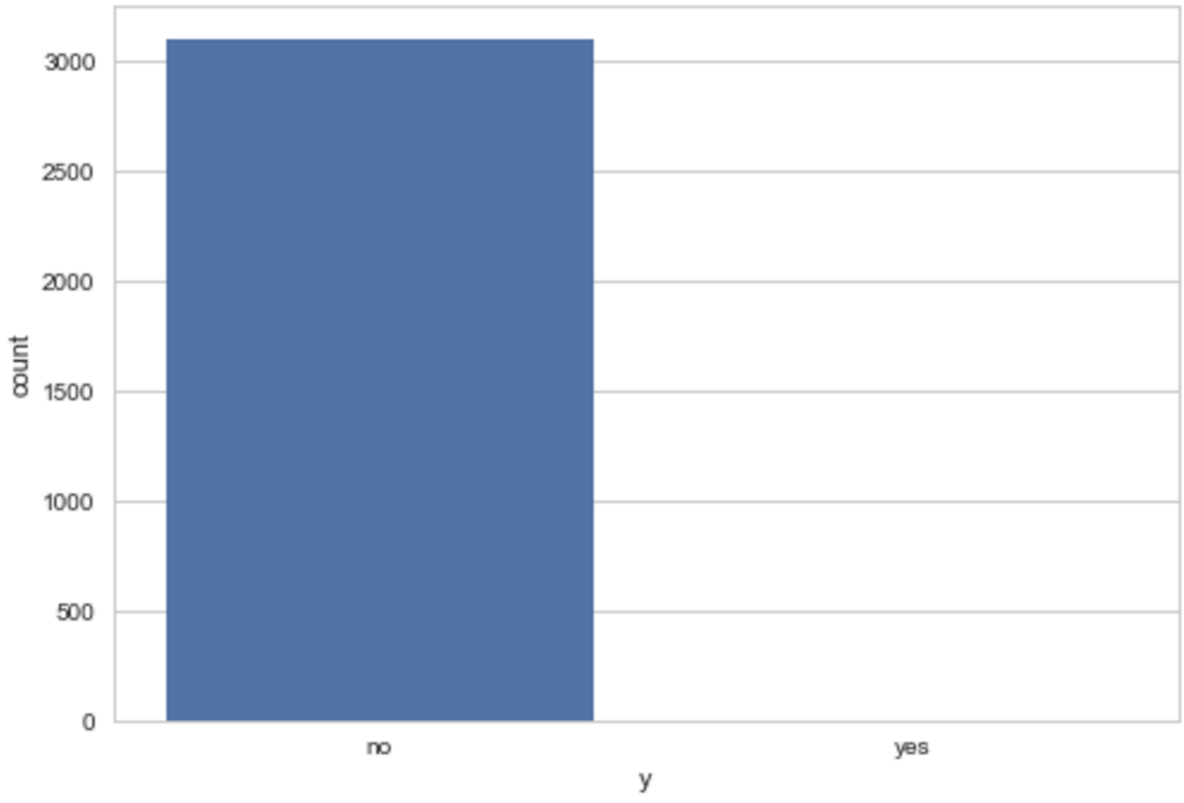
The following chart shows that there were two types of method of contact: one was via a mobile phone and other was a regular phone. The bars show yes or no for mobile and regular phones. It shows that the former was preferred over the latter.



The call duration attribute could highly influence the response variable. Longer the call duration the more chances are for a yes. When the duration of call is more than 600s, the response variable has value ‘no’ with frequency 1780, and has value ‘yes’ with frequency 1684. When the call duration is less than 50 seconds, the response variable is ‘no’ 3101 times, and ‘yes’ only in one case.

So, we can infer that call duration plays an important role in getting a positive response for the bank. But as soon as the call is over we get to know the outcome, and hence this can be used as benchmark that longer call duration is a sign that customer is going to avail a bank product.


1750
1500
1250
1000
750
500
250
0
count
no
yes
y


3000
2500
2000
1500
1000
500
0
count
no
yes
y

Models

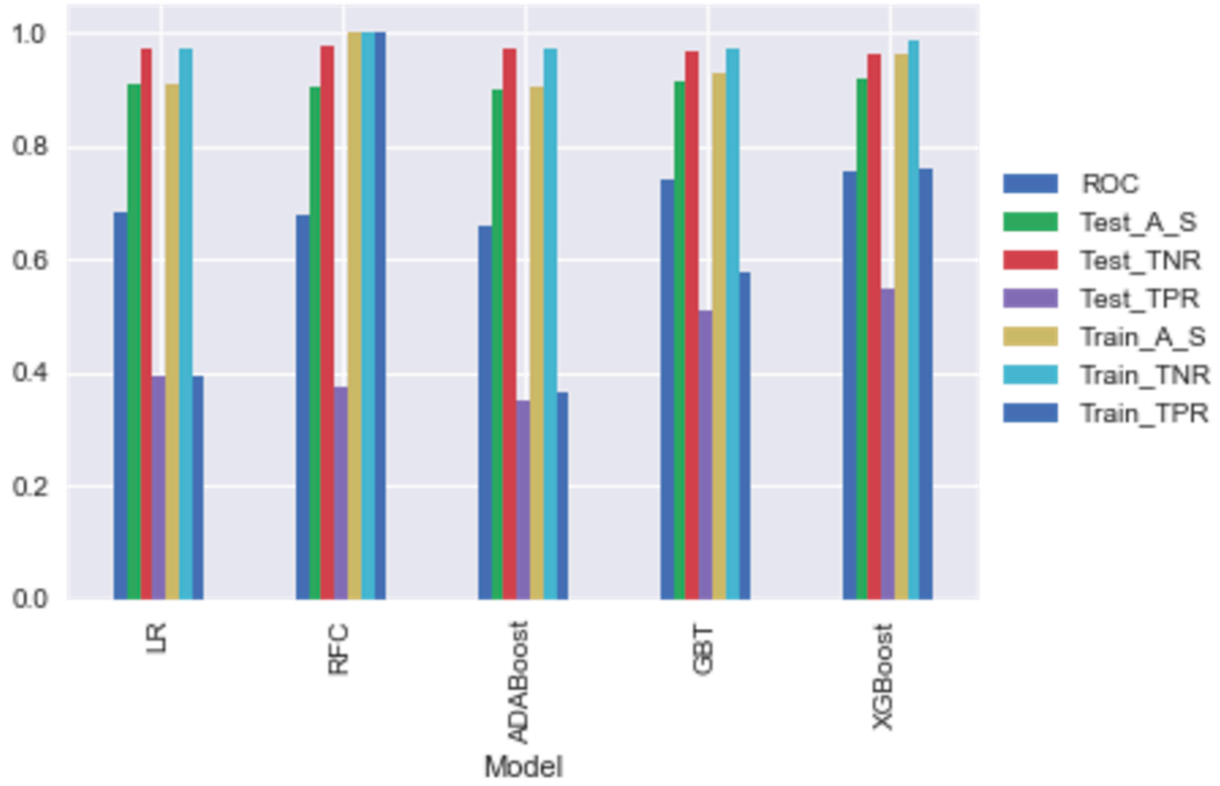
As this is classification problem with an imbalanced dataset, I first used several algorithms without explicitly dealing with the imbalance, which defines baseline.

Specifically I used the following classification algorithms: Logistic Regression (LR), Random Forest Classifier (RFC), ADABoost, Gradient Boosting (GBT), and XGBoost. To tune the hyperparameters, I used grid search and cross validation to optimize the parameters such as regularization strength (C) for LR, with L2 penalization.

The below figure indicates that model is predicting a poor True Positive Rate (TPR) which is the most important evaluation metric in our scenario as it indicates customers are willing to subscribe to bank products.

In the table, I use the following abbreviations: Test_X, and Train_X indicate the value for the performance metric X with the test and training sets, respectively.

The abbreviations for the performance metrics I used are: A_S (Accuracy Score), TNR (True Positive Rate), and TNR (True Negative Rate). The training and testing sets were created from the original dataset.



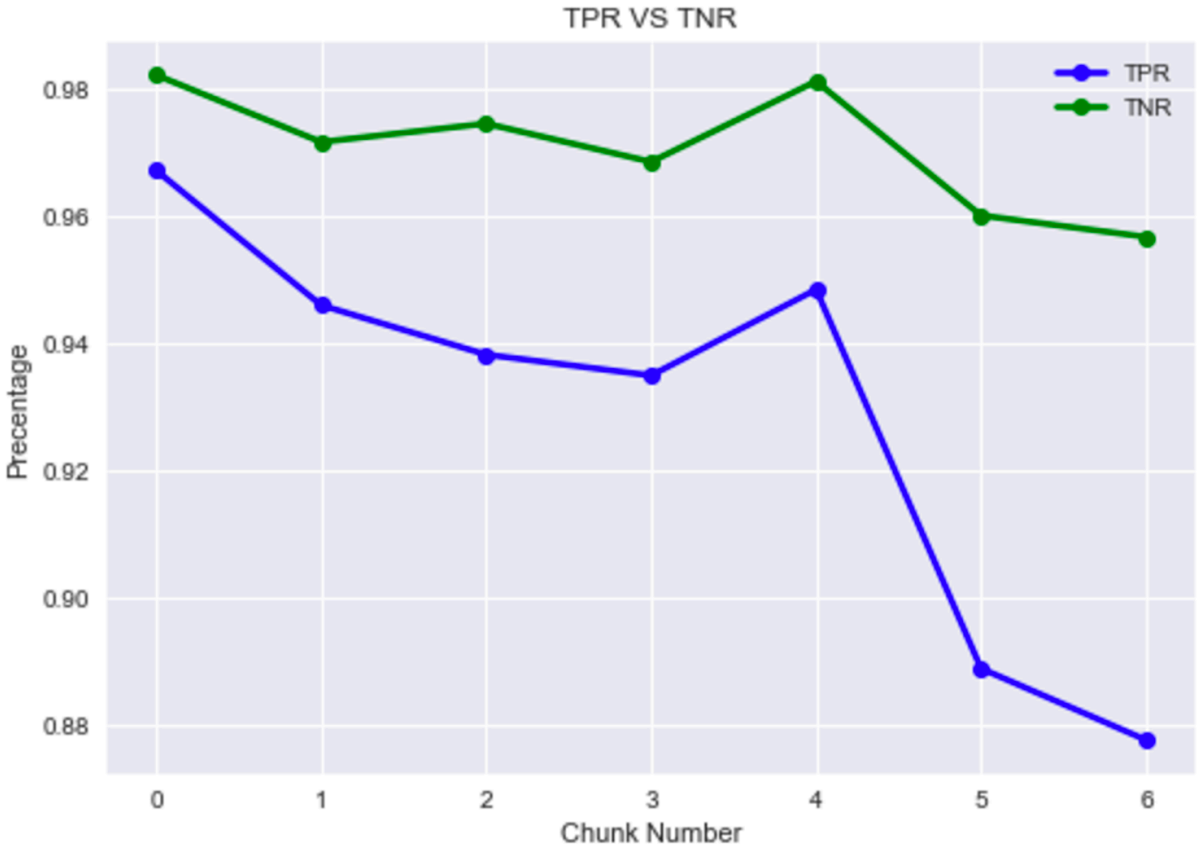
	Train_AS	Test_AS	Train_TNR	Test_TNR	Train_TPR	Test_TPR
Model						
LR	0.909	0.908	0.975	0.974	0.394	0.391
RFC	1.0	0.907	1.0	0.976	1.0	0.376
ADABoost	0.906	0.901	0.974	0.973	0.365	0.348
GBT	0.928	0.915	0.972	0.967	0.575	0.510
XGBoost	0.962	0.918	0.987	0.964	0.761	0.550

From the above table and chart one can conclude that XGBoost has the best performance, but with onlt 55% TPR.

Dealing with Class Imbalance

I used resampling techniques to explicitly deal with class imbalance. Broadly speaking there are two types of resampling methods: undersampling, and

oversampling. In the former, a sample from larger class (called the “majority class”) is taken so its size is equal to the size of the smaller class (called the “minority class”).The graph below shows the performance of tuned-regularized Logistic Regression, with respect to TPR and TNR, when it is applied to seven different samples taken from the majority class. The test/train split of the dataset is done in a stratified manner-which means that proportion of positive to negative classes is preserved in the split.



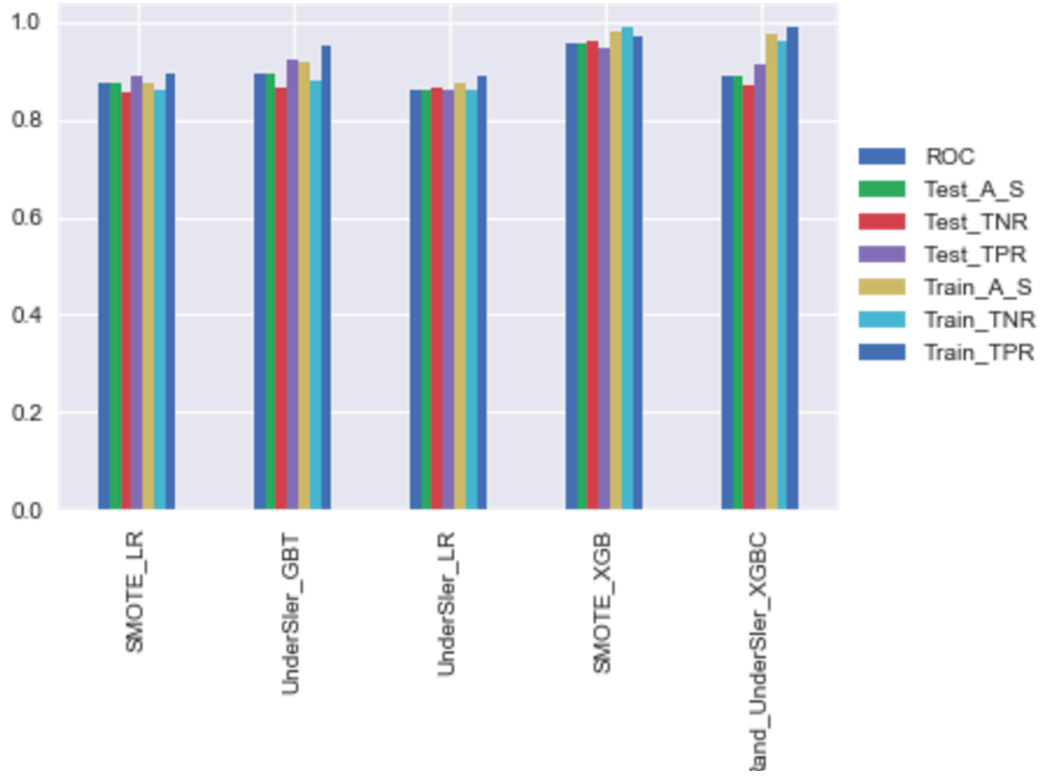
So on using Logistic Regression the TPR ranges from about 88% to 97% and TNR Ranges from 96% to 98%. This shows that undersampling significantly improves the performance of this model.

When it comes to oversampling, this method works with the minority class. In this we take samples (randomly and with replacement) of the minority class. So, we have 36548 samples from the majority class, and 36548 samples from the minority class. For this we use a synthetic data generation, which overcomes the class imbalances by generating data for the minority class. More specifically, I

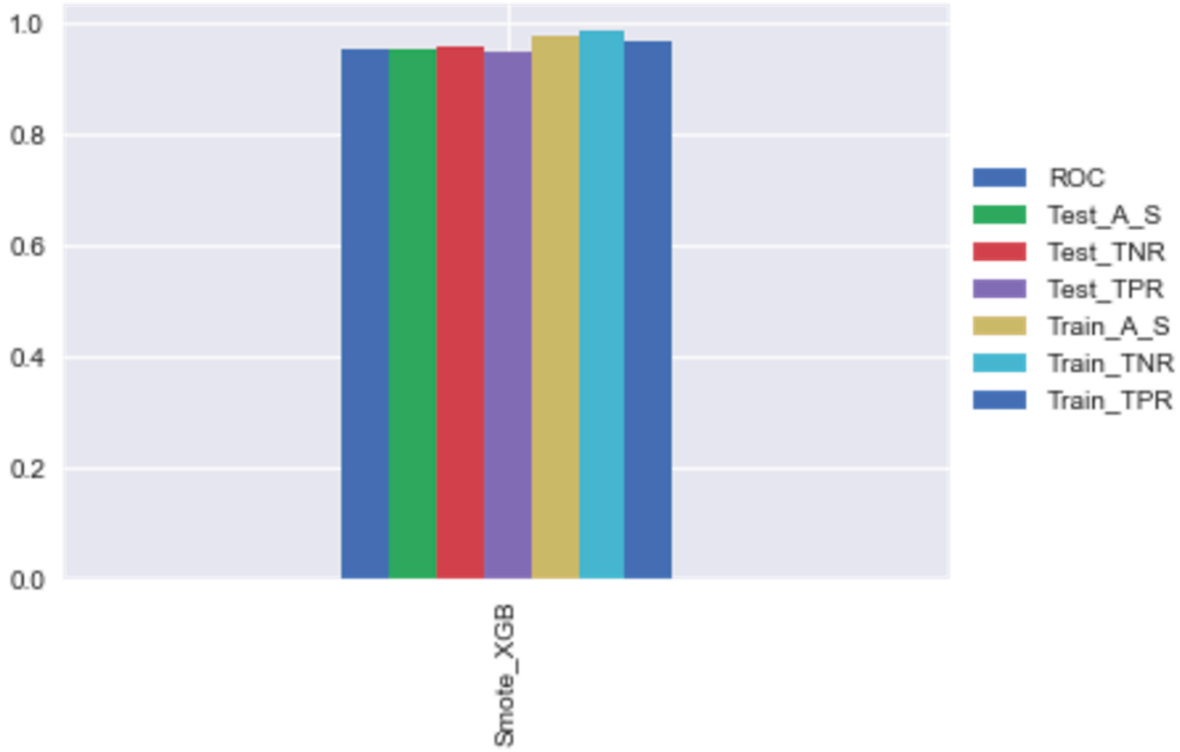
used the so called Synthetic Minority Oversampling Technique (SMOTE) discussed in http://contrib.scikit-learn.org/imbalanced-learn/stable/generated/imblearn.over_sampling.SMOTE.html .

In the table and chart below I show the results after applying various classification algorithms with resampling techniques such as SMOTE Logistic Regression (Smote_LR), UnderSampling Gradient Boosting (UnderSampler_GBT), UnderSampling Logistic Regression (UnderSampler_LR), Smote XGBoost (Smote_XGB) and UnderSampling XGBoost (UnderSampler_XGB).

	ROC	Test_A_S	Test_TNR	Test_TPR
Model				
SMOTE_LR	0.873	0.873	0.856	0.889
UnderSampler _GBT	0.891	0.891	0.863	0.920
UnderSampler _LR	0.861	0.861	0.863	0.859
SMOTE_XGB	0.953	0.953	0.958	0.947
UnderSampler_XGB	0.889	0.889	0.868	0.911



As we can see the results with resampling techniques are much better than algorithms applied on normal dataset. From the last figure we can conclude that Smote_XGB produces the best results for prediction of the response variable, which is summarized in the following chart.



Related Work

In this section I am comparing my work with the results published by Prusty Sagarika and Hany A. Elsalamony (see the web references at the end of this report.) Some of the algorithms they used are different in comparison with the ones I have used, but I can compare their results against mine with respect to the Sensitivity metric.

The table below shows the results reported by Sagarika.

Model	Accuracy	Sensitivity	F1
Naïve bayes(Modified Training Set)	78.15%	80%	78.6%
Decision Tree	93.7%	48%	70.1%
Decision Tree(Modified Training Set)	89.9%	88%	90.3%

Most efficient algorithm in the table is Decision Tree (Modified Training Set) with 88% Sensitivity.

The results reported by Elsalomony are shown in the following table.

Model	Accuracy	Sensitivity	Specificity
MLPNN	90.49%	62.20%	93.12%
TAN	88.45%	52.19%	91.73%
LR	90.43%	65.53%	92.16%
C5.0	90.09%	59.06%	93.23%

Most efficient algorithm in the table is Logistic Regression with 65.53% Sensitivity.

Finally, the table below summarizes the results I obtained.

Model	Accuracy	Sensitivity	Specificity
SMOTE_LR	87.3%	88.9%	85.6%
UnderSampler_GBT	89.1%	92%	86.3%

UnderSampler_LR	86.1%	85.9%	86.3%
SMOTE_XGB	95.3%	94.7%	95.3%
UnderSampler_XGBC	88.9%	91.1%	86.8%

Most efficient algorithm here is SMOTE_XGB with 94.7% Sensitivity.

I can conclude that all my results outperform the results reported by these two authors with respect to Sensitivity.

Recommendations to the Client

XGBoost model is the best algorithm to predict the output variable. Using XGBoost with SMOTE gives the best TPR which is about 94.7%, compared to 55% which was the result obtained without using any resampling technique. Even the Specificity is 95.8% which is far better than any other undersampling technique. In terms of the business problem, these results indicate that the client can be confident that positive and negative cases will be predicted with a high level of confidence.

Future Work

To improve the results, we can use a Neural Networks. We can add multiple hidden layers and make it a deep learning model and train it on a GPU to get high processing power and optimize it by using different optimizers.

Alternatively we can use higher n_estimators in XGBoost with a PC with higher processing power to further improve the TPR.

References

1. Sagarika's results: **https://pdfs.semanticscholar.org/a911/cbe221347b400d1376330591973bb561ff3a.pdf**
2. Elsalamony's results:

https://s3.amazonaws.com/recpass-production-project_files/project_docs/media/000/001/060/original/PrustySagarikaCourseProject_.pdf?1383838750