Mathematics in Machine Learning

Bank Marketing Dataset Analysis

Hajali Bayramov
S288874

July 27, 2021

Politecnico
di Torino

Contents

Abstract

In this project I introduced possible approaches for preparing the dataset and building a classification model for bank marketing data obtained from [1]. Specifically, methodology consists of preprocessing the data by cleaning, encoding, scaling, and balancing, and comparing built classification models, namely Decision Tree, Random Forest, Logistic Regression, Support Vector Machines, and AdaBoost by predefined metrics, such as accuracy and f1-score. The approaches achieve overall adequate results.

1 Introduction

Bank sector is one of the demanding fields of machine learning applications. The dataset used in this project specifically gathered by a Portuguese banking institution to improve the marketing campaigns which at a time was undergone by the phone calls. At the end of each call customer's answer to accepting bank term deposit (yes or no) is registered. Purpose of the machine learning is to predict this outcome so that there is no need to waste time on old manners.

2 Overview of the data

The dataset is based on "Bank Marketing" UCI dataset [1] and composed of 41188 instances with 20 features and 1 output variable. Columns of dataset is as following:

1. age (numeric)
2. job : type of job (categorical: "admin.", "blue-collar", "entrepreneur", "housemaid", "management", "retired", "self-employed", "services", "student", "technician", "unemployed", "unknown")
3. marital : marital status (categorical: "divorced", "married", "single", "unknown"; note: "divorced" means divorced or widowed)
4. education (categorical: "basic.4y", "basic.6y", "basic.9y", "high.school", "illiterate", "professional.course", "university.degree", "unknown")
5. default: has credit in default? (categorical: "no", "yes", "unknown")
6. housing: has housing loan? (categorical: "no", "yes", "unknown")
7. loan: has personal loan? (categorical: "no", "yes", "unknown")
8. contact: contact communication type (categorical: "cellular", "telephone")
9. month: last contact month of year (categorical: "jan", "feb", "mar", ..., "nov", "dec")
10. day_of_week: last contact day of the week (categorical: "mon", "tue", "wed", "thu", "fri")
11. duration: last contact duration, in seconds (numeric). Important note: this attribute highly affects the output target (e.g., if duration=0 then y="no"). Yet, the duration is not known before a call is performed. Also, after the end of the call y is obviously known. Thus, this input should only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.
12. campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13. pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14. previous: number of contacts performed before this campaign and for this client (numeric)
15. poutcome: outcome of the previous marketing campaign (categorical: "failure", "nonexistent", "success")

16. emp.var.rate: employment variation rate - quarterly indicator (numeric)
17. cons.price.idx: consumer price index - monthly indicator (numeric)
18. cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19. euribor3m: euribor 3 month rate - daily indicator (numeric)
20. nr.employed: number of employees - quarterly indicator (numeric)
    Output variable (desired target):
21. y: has the client subscribed a term deposit? (binary: "yes", "no")

Basic statistics of the numeric variables of the dataset is shown in the figure 1.

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	93.575664	-40.502600	3.621291	5167.035911
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	0.578840	4.628198	1.734447	72.251528
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.634000	4963.600000
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.344000	5099.100000
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

Figure 1: Descriptive statistics of the numeric variables.

3 Data exploration

3.1 Missing Values and Data Types

First thing we can do is to see if the data has any missing values. However it has no null or NAN values, it has some instances that has 'unknown' in one or more of its features. Features of 'job', 'maritial', 'education', 'default', 'housing', 'loan' have 330, 80, 1731, 8597, 990, 990 'unknown's respectively.

Moreover, information about columns' types are extracted as follows:

- Binary columns: ['default', 'housing', 'loan', 'y']
- Categorical columns: ['job', 'marital', 'education', 'contact', 'month', 'day_of_week', 'poutcome']
- Numerical columns: ['age', 'duration', 'campaign', 'pdays', 'previous', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx', 'euribor3m', 'nr.employed']

3.2 Data distribution

Following graphs are showing the distributions of features (first non-numerical by count-plot then numerical by the help of simple histograms and box plot):

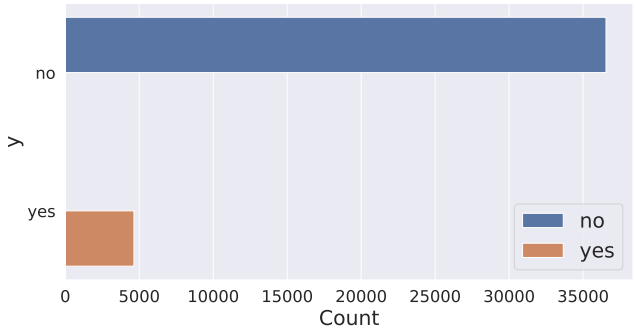

Figure 2: Distribution of y.

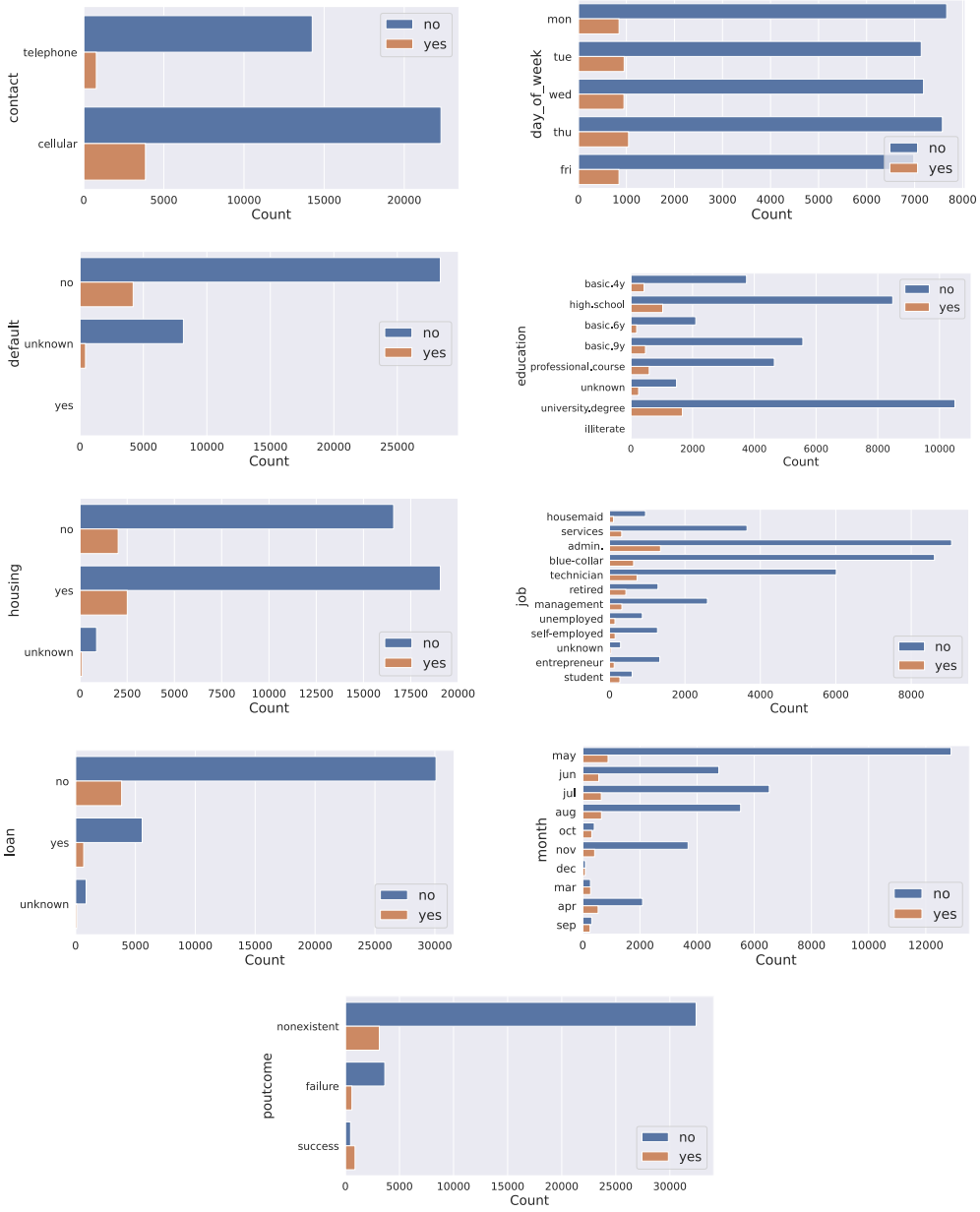

Figure 3: Distribution of non-numerical variables.

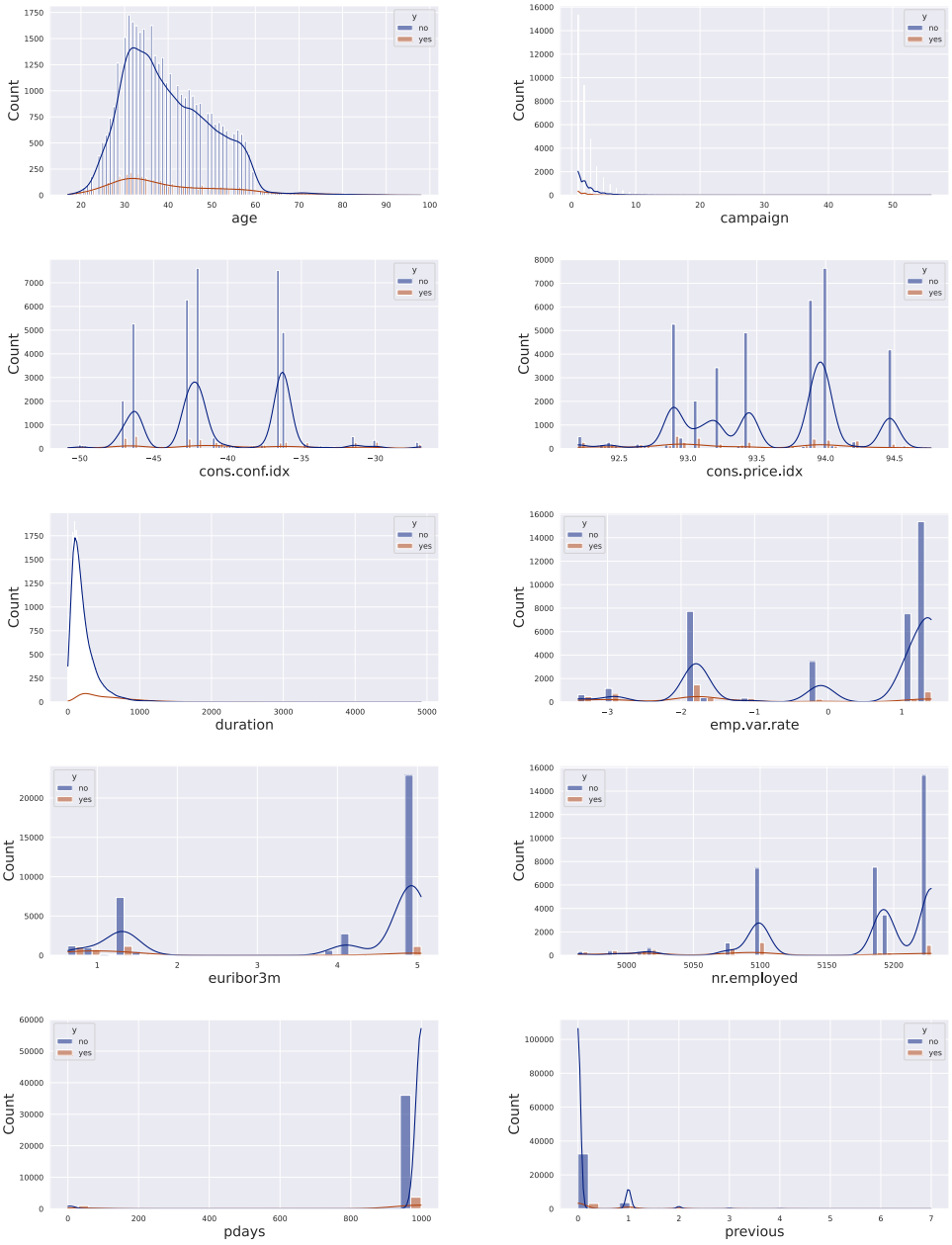

Figure 4: Histograms of numerical variables.

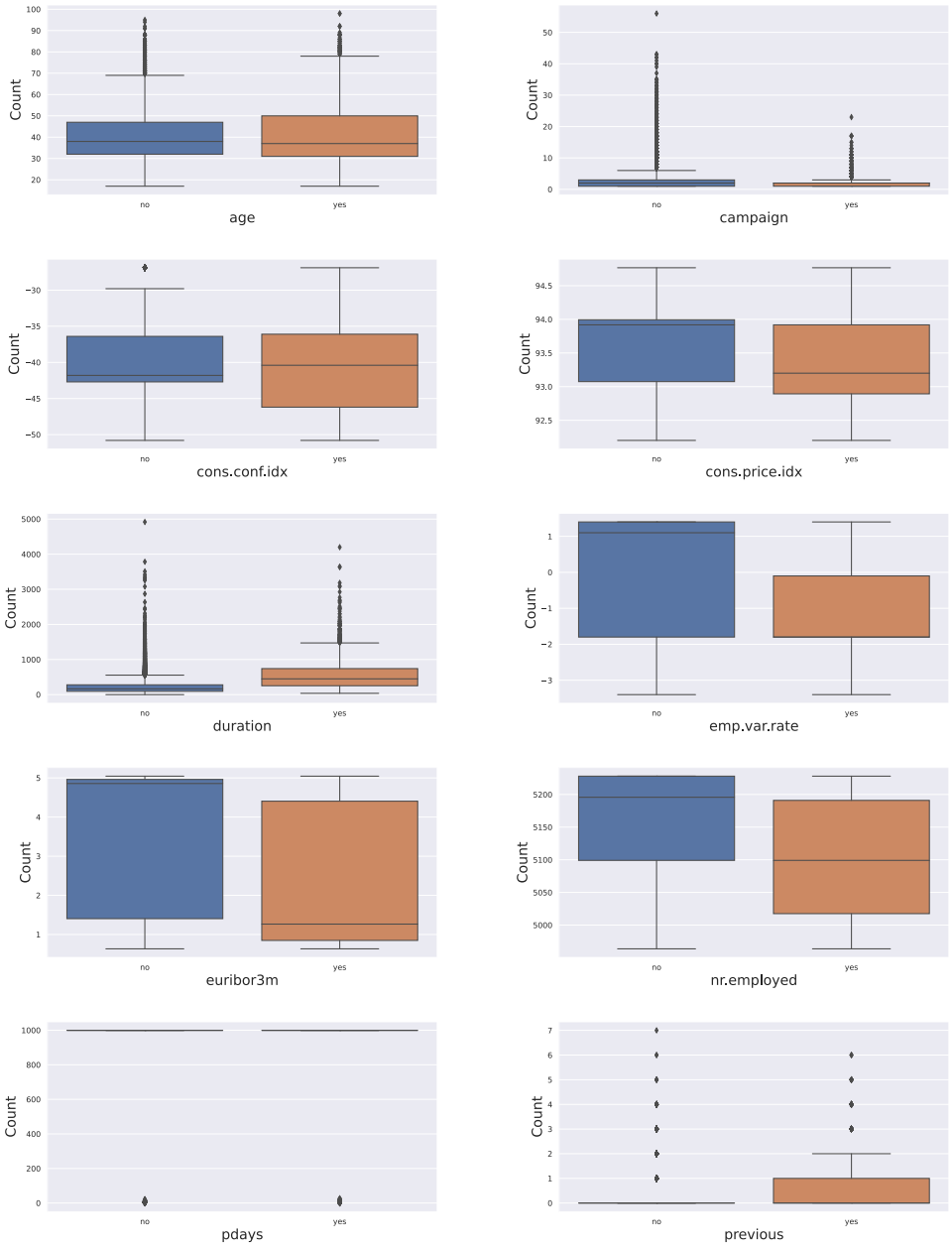

Figure 5: Box plots of numerical variables.

From the first glance, we can see that this dataset significantly suffers from imbalance. Most of the subjects are marked as non subscribed for bank term deposit. This means that we need to implement some artificial methods to balance the dataset.

Also, by looking at the box plots of figure 5, it becomes evident that there are many outliers in the dataset and we also need to deal with that.

Another good statistical parameter is Gini index which indicates the heterogeneity of each feature. In this project normalized Gini index is used:

$$G_n = \frac{m}{m-1} \cdot \left(1 - \sum_{i=1}^{m} f_i^2\right),$$

where m is the number of distinct categories, f_i is the relative frequency of each category i. This function gives results between 0 and 1, representing totally homogeneous (only one category is

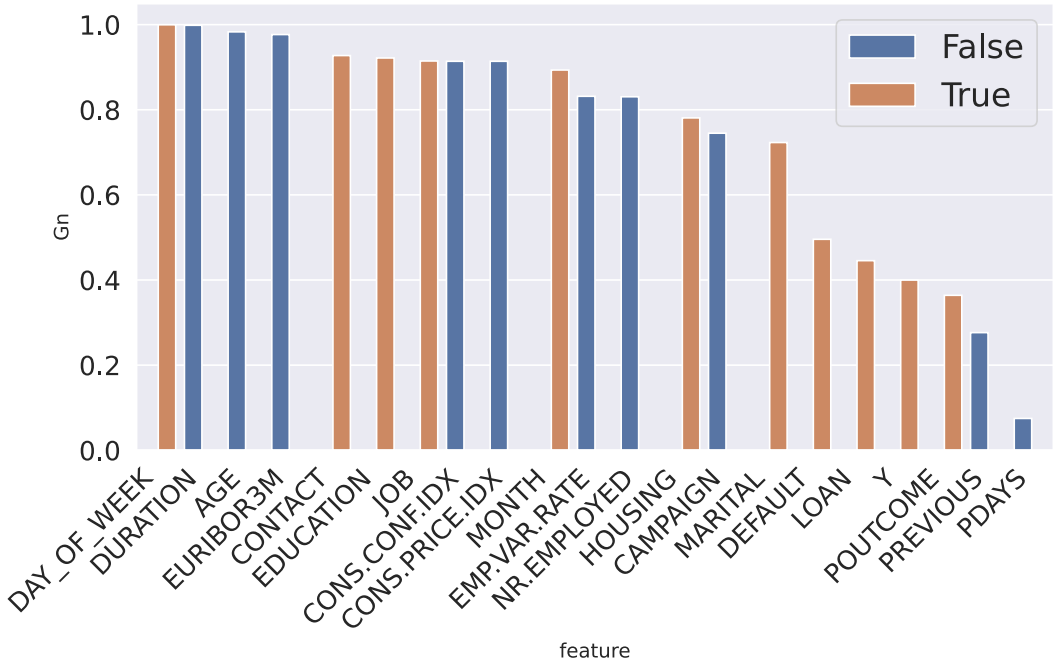

Figure 6: Gini indices of columns (legend shows if the feature is non-numerical).

dominant) and totally heterogeneous (all variables have equal probability to be in the column) respectively. Results are shown below in figure 6.

3.3 Data correlation

Correlation among input and between input and output can also be useful for data analysis before preprocessing. Each correlation measure is computed by the equation (pearson), where variables with bar represent the mean:

$$r(x,y) = \frac{\sum (x_i - \bar{x})(y_i - \bar{y})}{\sum (x_i - \bar{x})^2 (y_i - \bar{y})^2}.$$

Figure 7 shows the correlation between numerical features. It can be seen that 4 columns, namely, 'emp.var.rate', 'cons.price.idx', 'euribor3m', and 'nr.employed' have strong correlation among each other.

Moreover, it will also be useful to see the correlation between input features and output ('y'). This becomes possible with the help of point biserial correlation. In this kind of correlation we are able to find a correlation and p-value between binary(output) and continuous(input) variables. Like other correlation coefficients, this one varies between -1 and +1 with 0 implying no correlation. Correlations of -1 or +1 imply a determinative relationship [2].

$$r_{pb} = \frac{\bar{Y_1} - \bar{Y_0}}{s_y} \sqrt{\frac{N_1 N_0}{N(N-1)}},$$

where Y_0 and Y_1 are means of the metric observations coded 0 and 1 respectively; N_0 and N_1 are number of observations coded 0 and 1 respectively; N is the total number of observations and s_y is the standard deviation of all the metric observations. Also, for finding the correlation between output and non-numerical columns p-value obtained from χ^2 contingency is considered:

$$\chi^2 = \sum \frac{(O_i - E_i)^2}{E_i} \tag{1}$$

, where O_i is the observed frequency, E_i is the expected frequency. Where E_i is equal to $(row\_sum * column\_sum)/tota\_sum$ [3].

A p-value is the probability that the null hypothesis is true. In our case, it represents the probability that the correlation between x and y in the sample data occurred by chance. A p-value of 0.05 means that there is only 5% chance that results from your sample occurred due to chance[4]. In our case, variables that have correlation with 'y' has a chance of more that 0.05 are identified: $p_{housing}$ 0.06 and p_{loan} 0.58. These features have lower chances of having correlation with output, therefore should be manipulated in the preprocessing steps.

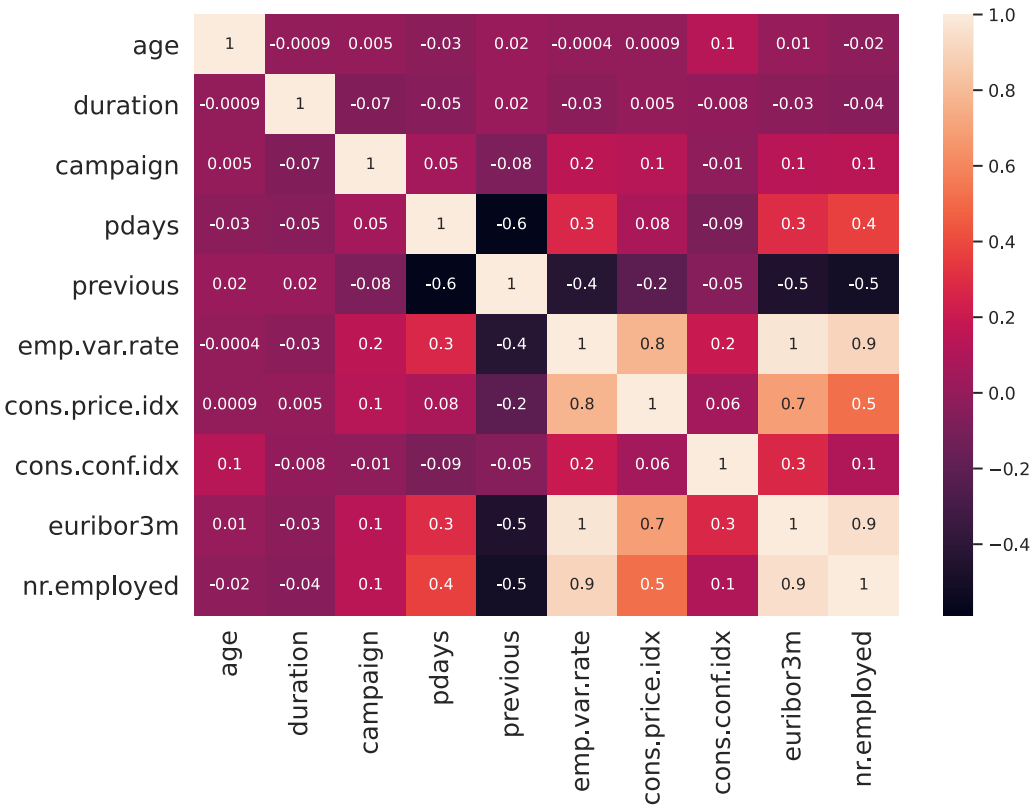

Figure 7: Correlation heatmap of numerical columns.

4 Data preprocessing

4.1 Data cleaning

In this stage previously discovered flaws are improved. First of all, there are 5 features with concerningly lower heterogeneity:

- default: there are only 3 rows with 'yes', it has too many rows for 'unknown', and even the heterogeneity inside the 'no' shows that using this feature for our model is not 'recommended'
- loan: we can see from p-value calculated (≈ 0.5) that the correlation between loan and target variable is not statistically significant, so we can neglect this feature too
- poutcome: even though this feature has very low gini index and imbalance towards 'success' is significant, we can see that it also has enough number of 'success' to later balance the dataset. For now we can keep this feature

- previous: we also keep this feature. It's skewed towards the end and it only has 8 distinct values, so we can improve this feature in balancing step
- pdays: in this feature 96% data is in one value, 999. We can remove this feature

Moreover, as suggested in the documentation of the dataset, we remove the 'duration' for the sake of realistic predictive model. Because duration of each call is not known before the call is actually performed. Also, 'housing' is removed since it has p-value greater than 0.05.

Having known all things mentioned above we can define the features we want to remove: ['default', 'loan', 'pdays', 'housing', 'duration'].

After eliminating some of the columns number of 'unknown's in the dataset drop significantly since most of them are accumulated inside 'default' and 'housing'. Now only 'job'(330), 'marital'(80), and 'education'(1731) have 'unknown's. There are three main ways to deal with the missing, or in this case 'unknown' instances:

- Possible class label (yes, no, unknown)
- Deletion
- Imputation (mean, median, random)

In this project, deletion method is used since instances are not very much. After dropping rows with 'unknown's, size of our dataframe dropped to 39191.

From the box plots of figure 5 we can see that some variables have a lot of outliers in columns 'age', 'previous', and 'campaign'. Outliers affect the mean, as well as encoding because it uses mean. Thus, huge proportion of outliers are removed from dataset, after which there are only 35158 rows left.

4.2 Feature encoding

All the non-numerical features should be encoded into numerical, either float or integer. Columns that are composed only by 'yes' and 'no' encoded into 1 and 0 respectively. For others one hot encoding technique is used.

4.3 Input-output split, Feature scaling, and Train-test split

After splitting input(X) and output(y) inside dataframe we are able to standardize X. This procedure is not applied on binary columns (all non-numerical columns are represented by binary - 1 and 0 after encoding) because it is not necessary. At last, we can split our X and y into train and test in order to put them into our model. Since our dataset is relatively small test size is 10% of dataset.

4.4 Balance dataset

As can be seen before the dataset is quite imbalanced towards the output 'no'. In order to overcome the problems imbalanced dataset creates we have few methods to try: gather more data, resampling by generate synthetic samples, etc. In our case it is not possible to get more data. That is why we need to focus on other techniques, like resampling. Oversampling is one of the resampling methods which increases the number of samples in the smallest class up to the size of biggest class. It is useful when the dataset is not so big in size. Undersampling however is more useful when the dataset is relatively bigger. It decreases the number of samples in biggest class to some extent.

In this project random over sampling, SMOTE, random over sampling, and SMOTE together with random under sampling techniques have been used [5]. Basic statistics derived after these methods are shown in the table 1, where target variable and X_test should not change.

SMOTE means Syntetic Minority Over-sampling Technique and it generates synthetic samples from minority class by filling the 'empty' spots between points, described better with visual in figure 8.

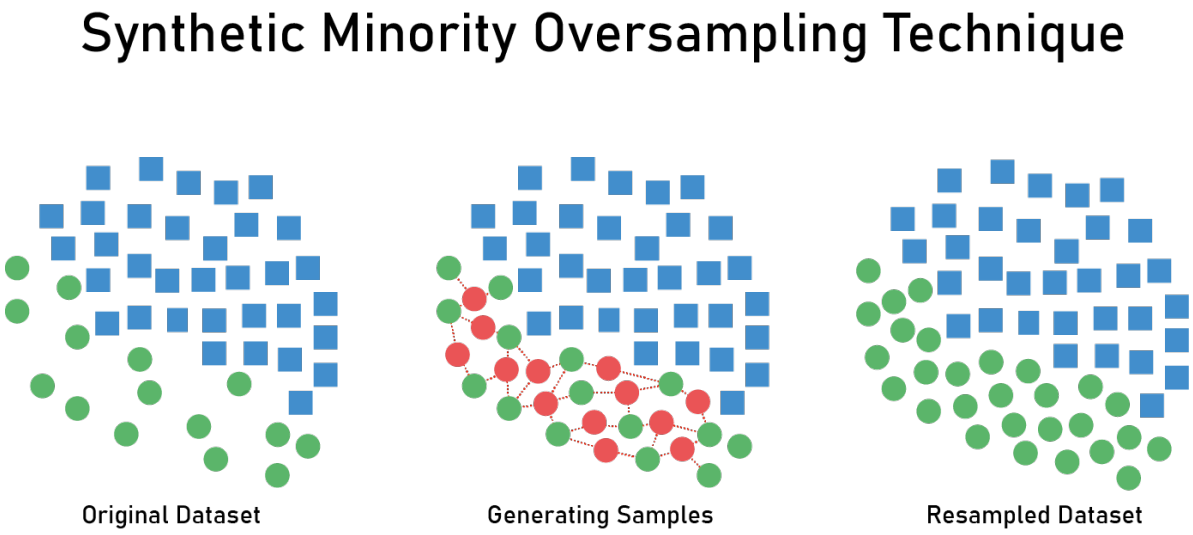

Figure 8: SMOTE explanation[6]

Table 1: Distribution of samples before and after resampling

X_train	Original	SMOTE	ROS	RUS	RUS+SMOTE
0	28421	28421	28421	3578	16105
1	3221	28421	28421	3221	16105

4.5 PCA

PCA is a principal component analysis and is one way to perform dimensionality reduction in dataset. It transforms the dataframe into new system so that the first principal component coincides with the biggest variance, second PC for second biggest and so on. Principal component analysis can be done in the our dataset, however, since it does not have many features or instances, time gained from applying PCA may not compensate what we lose in accuracy which is more important in our case. Nevertheless, PCA can show the difference of distribution between the training datasets before and after balancing. Figure 10 shows scatter plot of first two principle components of imbalanced data. As can be seen, instances with 'yes' are almost invisible among too many 'no's. Figure 11 shows results after balancing applied.

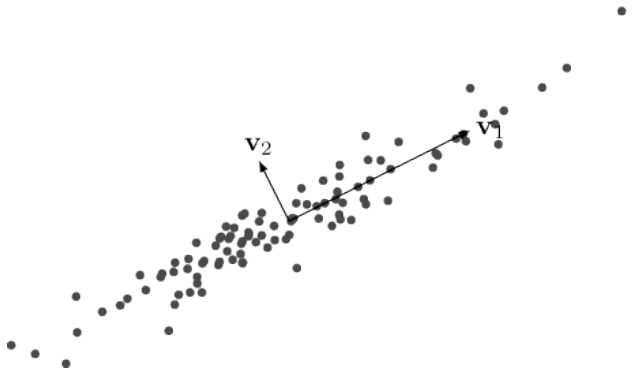

Figure 9: Principal components example[7]

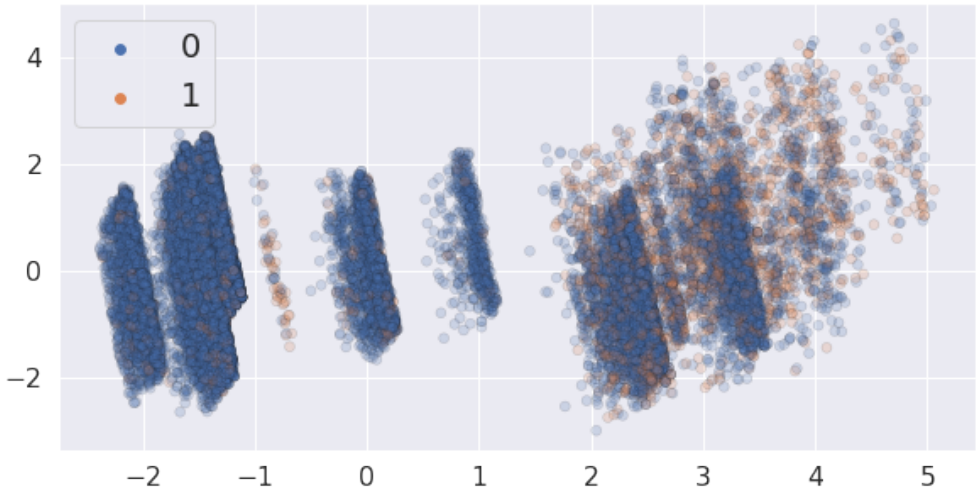

Figure 10: First 2 PCs of X_train.

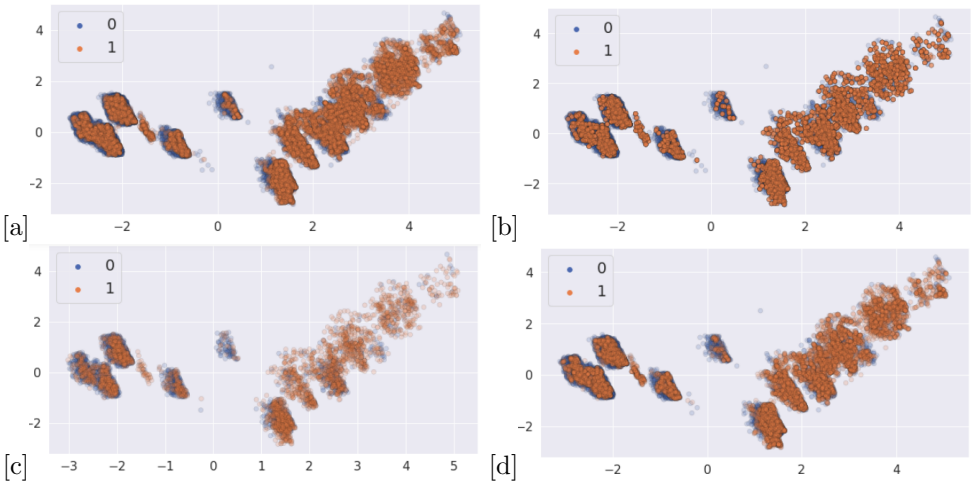

Figure 11: (a) First 2 PCs of X_train after SMOTE (b)First 2 PCs of X_train after ROS (c) First 2 PCs of X_train after RUS (d) First 2 PCs of X_train after RUS + SMOTE

5 Model building

5.1 Metrics

In this project mainly accuracy and f1-score metrics are used to measure the results of the tests. Accuracy is the number of correct predictions over total number of prediction [8], or:

$$Accuracy = \frac{\#ofcorrectpredictions}{Total\#ofpredictions} = \frac{TP + TN}{TP + TN + FP + FN},$$

where TP = True Positive, TN = True Negative, FP = False Positive, and FN = False Negative. Just accuracy does not provide us with good evaluation of the model if the set is highly imbalanced. For example, if there is 90% of 'no' and 10% of 'yes' in the dataset, and if machine puts 'no' for all the labels we get 90% accuracy. But is it really what we want? This result is highly biased towards 'no'. In this situation recall, precision, and most importantly f1-score. Recall helps us to

understand what proportion of actual positives was identified correctly [9].

$$Recall = \frac{TP}{TP + FN}.$$

Precision on the other hand, answers what proportion of positive identifications was actually correct [9].

$$Precision = \frac{TP}{TP + FP}.$$

F1 score is the harmonic mean of precision and recall. It is useful especially in our case since we have uneven class distribution.

In binary classification there is threshold which determines the border between the decisions, 0 or 1, yes or no. By default it is initialized as 0.5. Having known that fact, it is a good idea to show ROC curve. It shows us the performance of a classification model at different thresholds [10]. It plots two parameters:

- $TruePositiveRate = \frac{TP}{TP+FN}$
- $FalsePositiveRate = \frac{FP}{FP+TN}$

Figure 12 gives us a brief explanation. Lowering threshold means all the output will be classified as one class and it results in both higher TPR and FPR. Increasing it results in decrease in both rates.

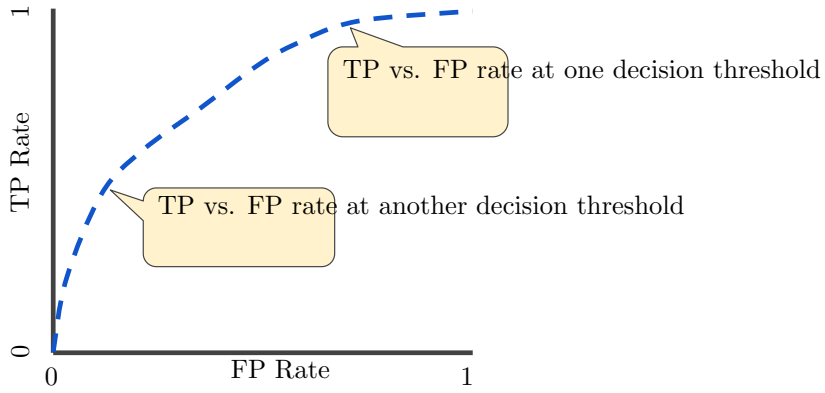

Figure 12: ROC.

Another good measure comes from ROC (figure 13) curve which is AUC (Area Under Curve) and it provides us with measure that is how well the predictions are ranked. It also measures quality of predictions without depending on threshold(usually 0.5, higher than that results in 1, otherwise 0).

Table 2: Evaluation results of train sets

	Original	SMOTE	ROS	RUS	RUS+SMOTE
Precision	0.64	0.50	0.30	0.34	0.45
Recall	0.17	0.34	0.60	0.59	0.45
F1-score	0.28	0.40	0.40	0.43	0.45
Accuracy	0.90	0.89	0.81	0.83	0.88

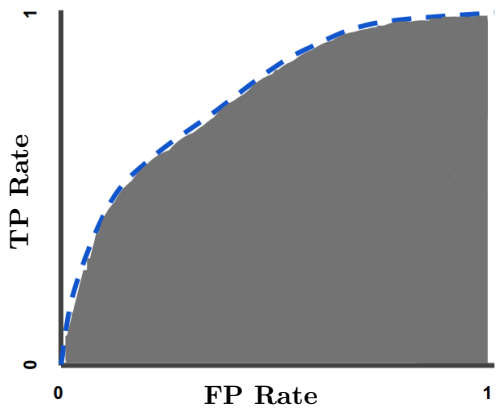

Figure 13: AUC.

5.2 Choosing train set

Previously, 5 train sets have been compiled to test the results in example model. For simplicity logistic regression model is chosen for comparison. All the metrics, including ROC and AUC (figure 14) is generated. Table 2 shows the results, from which it is evident that balancing is needed for the sake of better recall and precision. Thus, the decision is to choose RUS+SMOTE for the model.

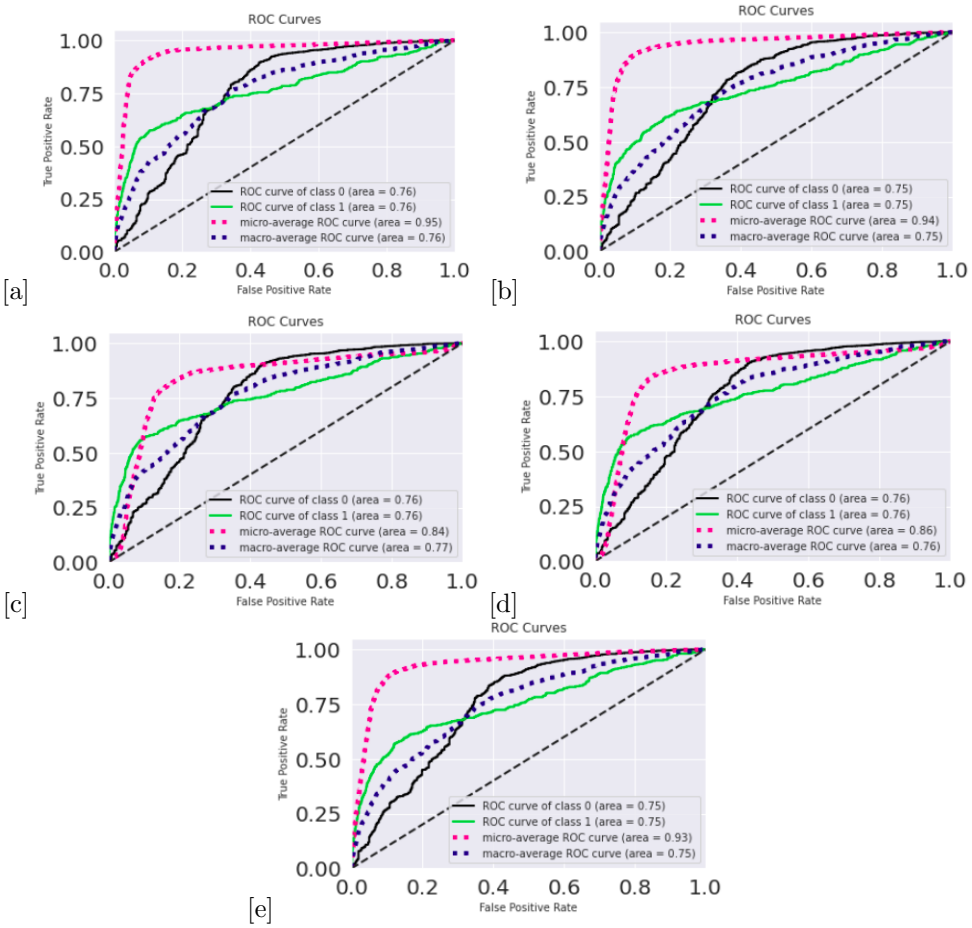

Figure 14: (a) First 2 PCs of original X_train (b)First 2 PCs of X_train after SMOTE (c) First 2 PCs of X_train after ROS (d) First 2 PCs of X_train after RUS (e) First 2 PCs of X_train after RUS+SMOTE

5.3 Algorithms

5.3.1 Decision tree

Decision tree is one of the most common classification techniques used in machine learning. It is built by a series of test questions and structured like a tree. After building a model, algorithm asks a followup question until it reaches a conclusion. Figure 15 and 16 taken from [11] can better represent the training and test phases respectively. In figure 15 decision tree is built based on questions asked in training data. When the new instance is asked from the model algorithm starts to ask questions and assign the corresponding output in figure 16.

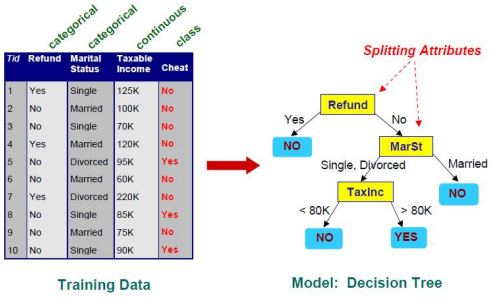

Figure 15: Training example of decision tree

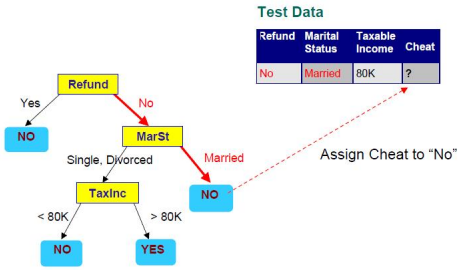

Figure 16: Test example of decision tree

The function measures the quality of the split is criterion which is either 'gini' for Gini impurity or 'entropy' for information gain.

$$Gini = \sum_{k=1}^{K} \hat{p}_{mk}(1 - \hat{p}_{mk}),$$

Entropy = $-\sum_{k=1}^{K} \hat{p}_{mk} log \hat{p}_{mk}$(2)Also 'best' and 'random' decision is made to choose split at each node. 'max_depth' is used to decide the maximum depth of the tree. If we do not assign a number to 'max_depth', then nodes are expanded until all leaves are pure. Finally, our grid parameters become:

- 'criterion':['gini', 'entropy']
- 'splitter':['best', 'random']
- 'max_depth':[10,100,None]

Figure 17 shows that the best result emerged in *config[6]* with 0.88 accuracy and 0.45 f1-score, which is {'criterion': 'entropy', 'max_depth': 10, 'splitter': 'best'}.

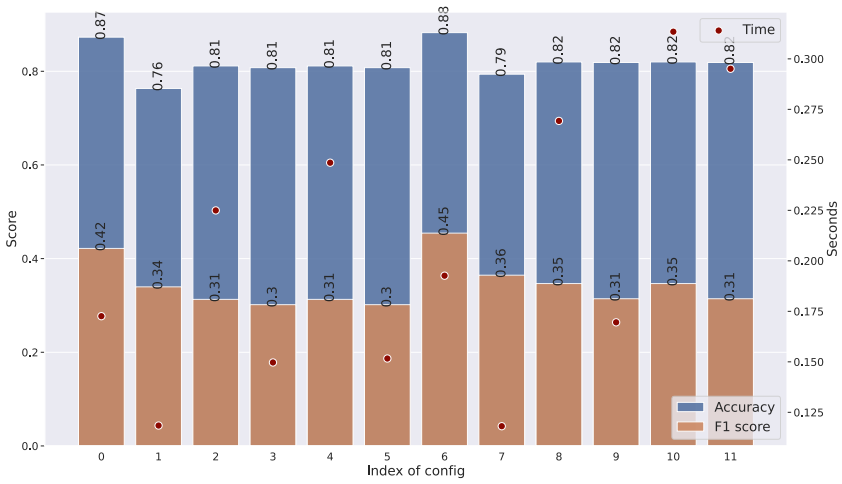

Figure 17: Test results of decision tree

5.3.2 Random forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting[12].

Random forest is one of the methods that uses ensemble learning where outputs of multiple models are put together to emerge the result. In this type of ensemble learning a decision tree is used for that purpose (figure 18).

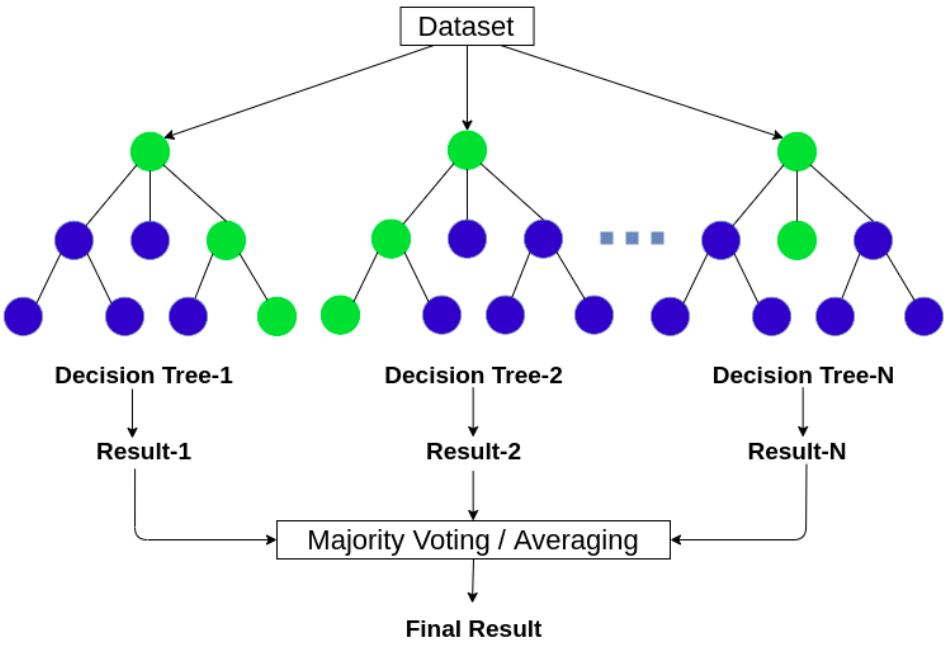

Figure 18: Random forest example [13]

Main advantage of random forest over decision trees is that it easily deal with overfitting. Since it uses the decision trees most of the parameters of the model are the same. Although this time 'max_features' and 'n_estimators' are added to grid. Former helps us to calculate the max features by sqrt(#features) or log2(#features), while latter represents the number of trees in forest. Furthermore, 'class_weight' is set to 'balanced' in order to achieve better results with imbalanced dataset we have. Grid parameters for random forest are as follows:

- 'criterion':['gini', 'entropy']
- 'n_estimators':[10, 50, 100]
- 'max_features':['auto', 'sqrt','log2']
- 'class_weight':['balanced'] (always 'balanced')

After running the tests we can see that (figure 19) almost all the results are identical. We can choose the best set of parameters based on time and it turns out to be 'config' number 7, which is {'class_weight': 'balanced', 'criterion': 'gini', 'max_features': 'log2', 'n_estimators': 50}

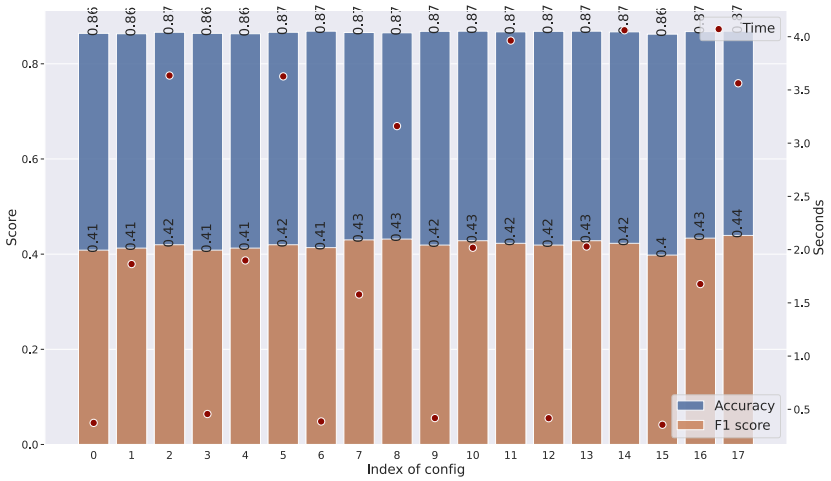

Figure 19: Test results of random forest

5.3.3 Logistic regression

Logistic regression is the most common classification algorithm when the desired output is binary. Possible outcomes of single trial are modeled using logistic function. The most common used function is sigmoid (figure 20):

$$sig(t) = \frac{1}{1 + e^{(} - t)}$$

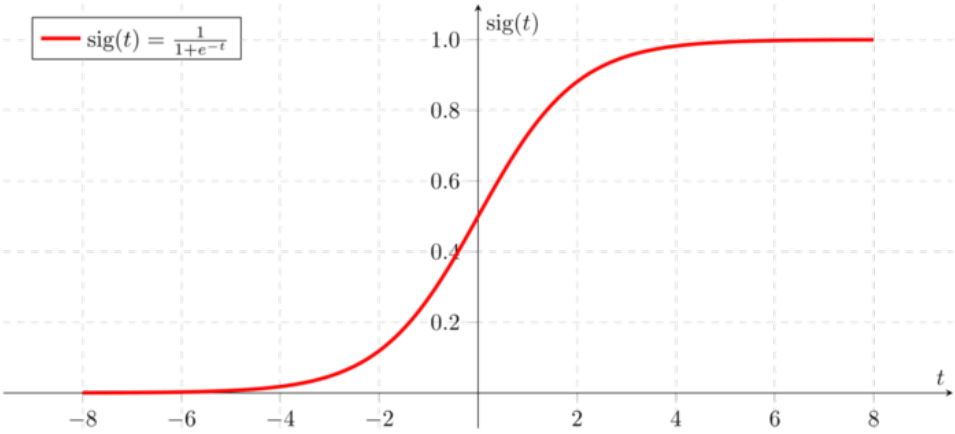

Figure 20: Sigmoid function [14]

Sigmoid function is really helpful to map results to either 0 or 1, depending on the threshold by the help of the function:

$$P(Y = 1|X) = \frac{e^{\beta_0 + \beta_1 X}}{1 + e^{\beta_0 + \beta_1 X}} \tag{3}$$

. Parameters of β can be estimated by using Maximum Likelihood Estimation

$$l(\beta_0, \beta_1) = \prod_{i:y_i=1} p(x_i) \prod_{i:y_i=0} (1 - p(x_i)) \tag{4}$$

There is also so-called solver, which is the algorithm used in the optimization problem to minimise the cost function. This could be 'newton-cg' which implements a quadratic function minimisation, 'lbfgs' (Limited-memory Broyden-Fletcher-Goldfarb–Shanno) which is alternative to the Newton's Method but Hessian matrix in this case is approximated, 'liblinear' which is a linear classification library that uses coordinate descent, 'sag' which stands for Stochastic Average Gradient optimizes the sum of a finite number of smooth convex functions, 'saga' which is the a variant of 'sag' that also supports the non-smooth penalty. 'fit_intercept' specifies if a constant (a.k.a. bias or intercept) should be added to the decision function [15]. At the end, tested parameters become

- 'solver':['newton-cg', 'lbfgs', 'liblinear', 'sag', 'saga']
- 'fit_intercept':[True,False]

After running the tests we can see that (figure 21) almost all the results are identical. We can choose the best set of parameters based on time and it turns out to be 'config' number 1, which is {'solver': 'lbfgs', 'fit_intercept': True}, which makes total sense because 'lbfgs' is recommended for small datasets like ours.

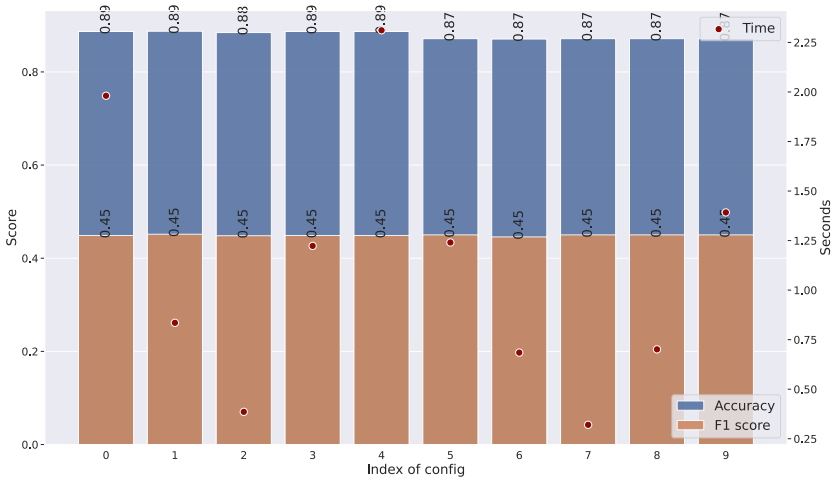

Figure 21: Test results of logistic regression

5.3.4 SVM

SVM (Support Vector Machines) is a supervised machine learning technique that is useful in classification, regression and outliers detection. Its main principle is to use a function (so-called *kernel*) to transfer dataset into another domain in which it is much easier to separate the classes with a borderline. In classification it is even more straightforward. Figure 22 better describes the separation visually and compare the different kernel functions.

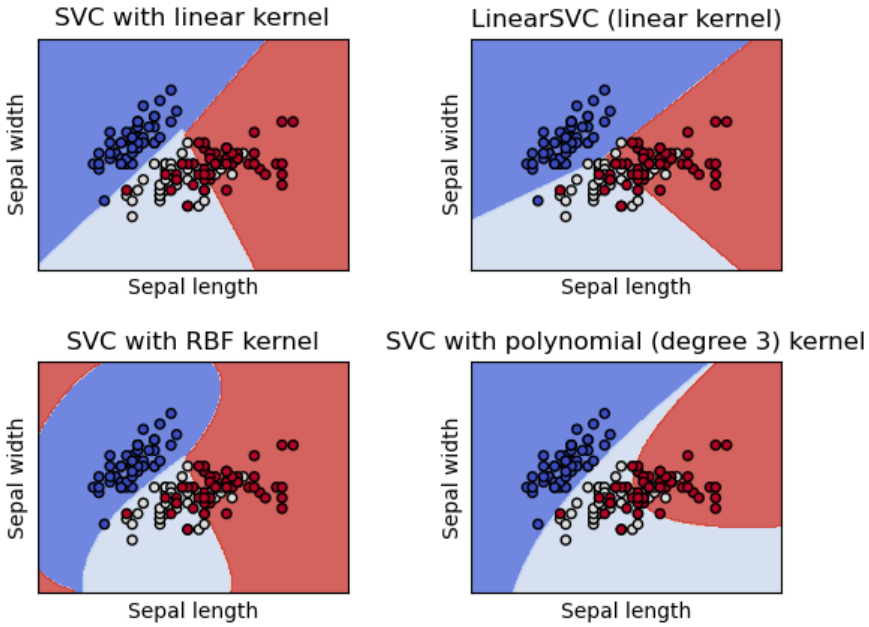

Figure 22: SVM (SVC stands for C-Support Vector Classification) [16].

In this project three types of kernels are used:

- 'kernel':['linear', 'rbf', 'sigmoid']

$$k_{linear}(x, y) = x^T y + c, c = interceptconstant$$

$$k_{rbf}(x, y) = exp(-\frac{||x - y|\,|^2}{2\sigma^2}), \sigma = adjustableparameter$$

$$k_{sigmoid}(x, y) = tanh(\alpha x^T y + c), \alpha = slope, c = interceptconstant$$

After running the tests we can see that (figure 23) 'linear' and 'rbf' have generated pretty decent results while 'sigmoid' left far behind. We can choose 'rbf' as a kernel since it takes relatively less time.

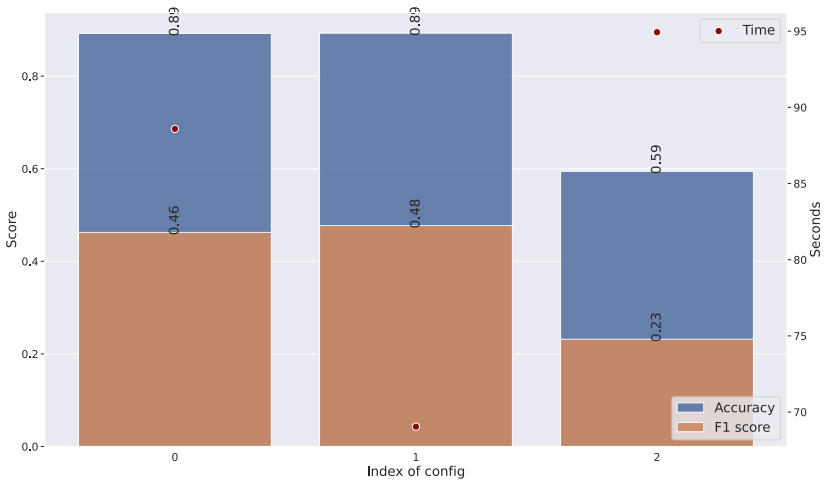

Figure 23: Test results of SVM

5.3.5 AdaBoost

> *An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases[17]*

AdaBoost is another method in ensemble learning (figure 24). Here first an original dataset is classified. Then second time it uses the same dataset but weights of incorrectly classified instances are adjusted to focus on difficult cases. By defaults it uses decision tree as a base estimator. For simplicity, only 'n_estimators' parameter is tuned for the current project.

- 'n_estimators':[10,50,100]

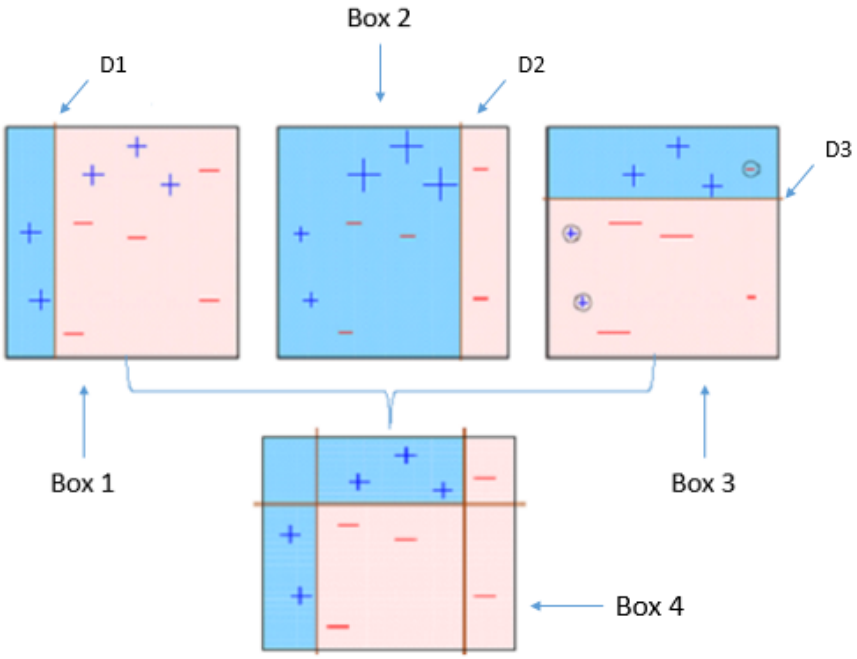

Figure 24: AdaBoost example [18].

After running the tests we can see that (figure 25) almost all the results are identical. We can choose the best set of parameters based on time and it turns out to be 'config' number 2, which is {'n_estimators': 100}.

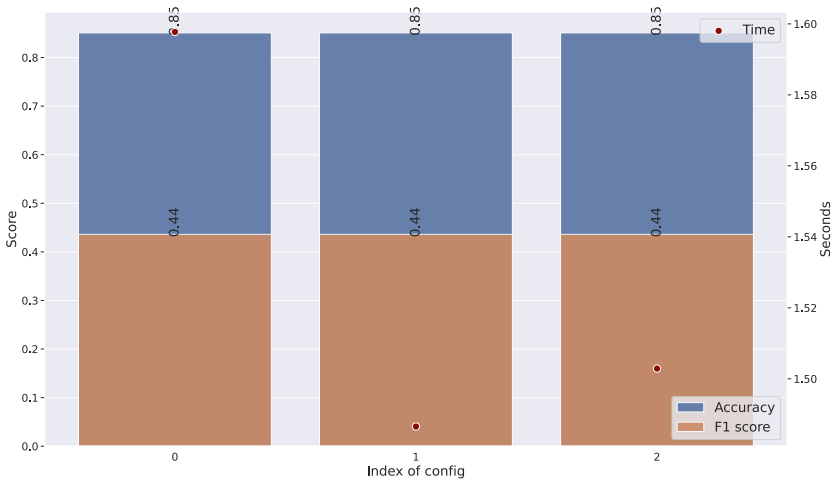

Figure 25: Test results of AdaBoost

6 Conclusion

To sum up, the models are great success in terms of the accuracy; all of the models show accuracy about as high as 90%. This is mostly because of the preprocessing step where unnecessary features

that deflect the output are removed from the training. Nevertheless, the biggest flaw of the model is its quite low precision and recall because of the unbalance existing inside the dataset.

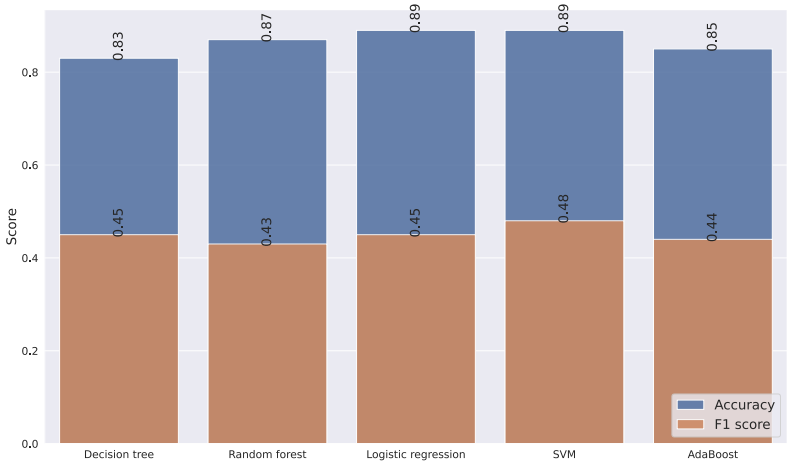

Figure 26: Overall results

The best methods to overcome this issue are used in the project, yet the results are unsatisfactory — f1-score below 50%. This indicates that the results are quite biased towards the customer saying 'no' to the call. Which may lead to the possible loss of the potential client. In the current case-study it might not seem significant, but if the model is applied on more serious cases like hiring a potential candidate or having malignant cancer (false negative can lead to failure for early diagnosis which is life-threatening), it becomes really consequential. Last issue is an ethical problem demanding discussions over using the model or not.

References

[1] S. Moro, P. Cortez, and P. Rita, "A data-driven approach to predict the success of bank telemarketing," *Decision Support Systems*, vol. 62, pp. 22–31, 2014.

[2] Pointbiserialr. [Online]. Available: https://docs.scipy.org/doc/scipy/reference/generated/scipy.stats.pointbiserialr.html

[3] How to calculate expected frequency. [Online]. Available: https://www.statology.org/expected-frequency/

[4] Z. Jaadi. Everything you need to know about interpreting correlations. [Online]. Available: https://towardsdatascience.com/eveything-you-need-to-know-about-interpreting-correlations-2c485841c0b8

[5] A. L. Duca. How to balance a dataset in python. [Online]. Available: https://towardsdatascience.com/how-to-balance-a-dataset-in-python-36dff9d12704

[6] Z. Jefferson. Bank data: Smote. [Online]. Available: https://medium.com/analytics-vidhya/bank-data-smote-b5cb01a5e0a3

[7] Pca. [Online]. Available: http://www.ce.unipr.it/people/medici/geometry/node50.html

[8] Classification: Accuracy. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/accuracy

[9] Classification: Precision and recall. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall?hl=en

[10] Classification: Roc curve and auc. [Online]. Available: https://developers.google.com/machine-learning/crash-course/classification/roc-and-auc?hl=en

[11] Decision tree classifier. [Online]. Available: http://mines.humanoriented.com/classes/2010/fall/csci568/portfolio_exports/lguo/decisionTree.html

[12] A random forest classifier. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html?highlight=randomforestclassifier#sklearn.ensemble.RandomForestClassifier

[13] A. SHARMA. Decision tree vs. random forest – which algorithm should you use? [Online]. Available: https://www.analyticsvidhya.com/blog/2020/05/decision-tree-vs-random-forest-algorithm/

[14] S. Swaminathan. Logistic regression — detailed overview. [Online]. Available: https://towardsdatascience.com/logistic-regression-detailed-overview-46c4da4303bc

[15] Logistic regression. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[16] Svm. [Online]. Available: https://scikit-learn.org/stable/modules/svm.html#svm-classification

[17] Adaboostclassifier. [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.AdaBoostClassifier.html?highlight=adaboost#sklearn.ensemble.AdaBoostClassifier

[18] A. Desarda. Understanding adaboost. [Online]. Available: https://towardsdatascience.com/understanding-adaboost-2f94f22d5bfe