Bank Marketing Campaign Analysis

Omid Erfanmanesh

June 26, 2021

Abstract

Bank marketing campaign is a method for increasing the number of clients and in this paper, I am going to analyse the data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls.This current analysis consider the impact of different classifier models and pca component analysis on this dataset. Moreover, all these model will be evaluated with metric of *f1 score*. As the result, I obtained the score of *0.99* from *SVM* model.

Contents

1 Introduction

Marketing to potential clients has always been a crucial challenge in attaining success for banking institutions. It's not a surprise that banks usually deploy mediums such as social media, customer service, digital media, and strategic partnerships to reach out to customers. But how can banks market to a specific location, demographic, and society with increased accuracy? With the inception of machine learning - reaching out to specific groups of people has been revolutionized by using data and analytics to provide detailed strategies to inform banks which customers are more likely to subscribe to a financial product.

In this project on bank marketing with machine learning, I will explain how a particular Portuguese bank can use predictive analytics to help prioritize customers which would subscribe to a bank term deposit.

Marketing campaigns are characterized by focusing on the customer's needs and their overall satisfaction. Nevertheless, there are different variables that determine whether a marketing campaign will be successful or not. There are certain variables that we need to take into consideration when making a marketing campaign.

2 Dataset

In this analysis, we are going to use the data is related with direct marketing campaigns of a Portuguese banking institution in order to analyzing the markets campaign benefits. The marketing campaigns were based on phone calls. [36] Often, more than one contact to the same client was required, in order to access if the product (bank term deposit) would be ('yes') or not ('no') subscribed. [1]

2.1 About Dataset

Fortunately, there are *4521* samples with no missing values in this dataset. The Table 2 describes our dataset briefly. As what you can see in this table, the columns of *Age*, *Job*, *Marital*, *Education*, *Default*, *Balance*, *Housing*, *Loan* are bank clients information. Also, the columns of *Contact*, *Day*, *Month*, *Duration* are related to the last contact of the current campaign. In Addition, *Campaign*, *Pdays*, *Previous*, *Poutcome* are related with the last contact of the current campaign. Finally, *Y* is the target values. Moreover, to better understanding each column, I am going to introduce them in the 2.4.

	count	mean	std	min	25%	50%	75%	max
age	4521	41.17	10.57	19	33	39	49	87
balance	4521	1422.65	3009.63	-3313	69	444	1480	71188
day	4521	15.91	8.24	1	9	16	21	31
duration	4521	263.96	259.85	4	104	185	329	71188
campaign	4521	2.79	3.10	1	1	2	3	50
pdays	4521	39.76	100.12	-1	-1	-1	-1	871
previous	4521	0.54	1.69	0	0	0	0	25

Table 1: Numerical Features Overviews

2.2 Skewness and Kurtosis

Before starting to explore our dataset, it is necessary to know about *Skewness* and *Kurtosis*.

Skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. As what the Fig. 1 shows, Negative skew commonly indicates that the tail is on the left side of the distribution, and positive skew indicates that the tail is on the right.[21]

Kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. Like skewness, kurtosis describes the shape of a probability distribution and

Dataset Description			
Column	**Type**	**Description**	**Example**
bank client data:			
Age	NUMERIC	type of job	"admin.", "unknown", "unemployed", "management", "housemaid", "entrepreneur", "student", "blue-collar", "self-employed", "retired", "technician", "services"
Job	CATEGORICAL	marital status	"married", "divorced", "single"; note: "divorced" means divorced or widowed
Marital	CATEGORICAL	has credit in default?	"unknown", "secondary", "primary", "tertiary"
Education	CATEGORICAL	average yearly balance, in euros	"yes", "no"
Default	BINARY	has credit in default?	-
Balance	NUMERIC	average yearly balance, in euros	"yes", "no"
Housing	BINARY	has housing loan?	"yes", "no"
Loan	BINARY	has personal loan?	-
related with the last contact of the current campaign:			
Contact	CATEGORICAL	contact communication type	"unknown", "telephone", "cellular"
Day	NUMERIC	last contact day of the month	-
Month	CATEGORICAL	last contact month of year	"jan", "feb", "mar", ..., "nov", "dec"
Duration	NUMERIC	last contact duration, in seconds	-
related with the last contact of the current campaign:			
Campaign	NUMERIC	number of contacts performed during this campaign and for this client	includes last contact
Pdays	NUMERIC	number of days that passed by after the client was last contacted from a previous campaign	-1 means client was not previously contacted
Previous	NUMERIC	number of contacts performed before this campaign and for this client	-
Poutcome	CATEGORICAL	outcome of the previous marketing campaign	"unknown", "other", "failure", "success"
target			
Y	BINARY	subscribe or not ?	"yes", "no"

Table 2: About data

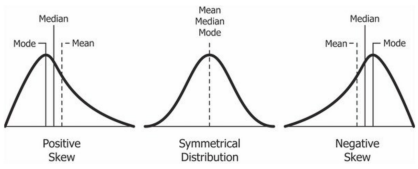


Figure 1: Skewness

there are different ways of quantifying it for a theoretical distribution and corresponding ways of estimating it from a sample from a population. [13]

Skewness is important because First, linear models work on the assumption that the distribution of the independent variable, and the target variable are similar. Therefore, knowing about the skewness of data helps us in creating better linear models. Second,Since our data is positively skewed here, it means that it has a higher number of data points having low values, i.e., cars with less horsepower. So when we train our model on this data, it will perform better at

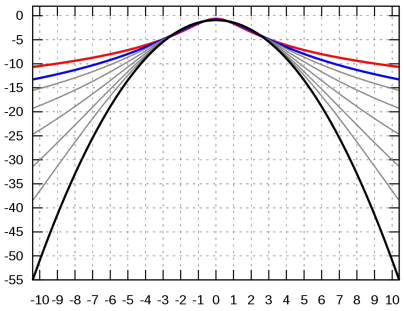


Figure 2: log-pdf for the Pearson type VII distribution with excess kurtosis of infinity (red); 2 (blue); 1, 1/2, 1/4, 1/8, and 1/16 (gray); and 0 (black)

predicting the mpg of cars with lower horsepower as compared to those with higher horsepower. Also, skewness tells us about the direction of outliers. You can see that our distribution is positively skewed and most of the outliers are present on the right side of the distribution. Note that the skewness does not tell us about the number of outliers. It only tells us the direction. [32]

	skw	skw log	skw sqrt	skw boxcox	krt	krt log	krt sqrt	krt boxcox
age	0.69	-0.35	0.83	-0.32	0.34	0.87	1.54	0.94
balance	6.59	1.88	2.56	3.45	88.39	4.50	9.50	7.10
day	0.09	-2.35	0.30	-1.27	-1.03	0.00	1.00	0.00
duration	2.77	1.01	1.66	1.39	12.53	2.67	3.81	3.48
campaign	4.74	1.55	2.17	2.54	37.16	3.66	6.26	5.29
pdays	2.71	0.99	1.64	1.36	7.95	2.30	3.16	2.88
previous	5.87	1.77	2.42	3.11	51.995	3.98	7.35	5.95

Table 3: Skewness and Kurt of numerical columns

Since we know how much the skewness is important, we should try to decrease the skewness of data. In order to solve this problem, we can use transformers function. In the following, I am going to introduce the transformers function that are used in this project to reduce the skewness.

1. Log Transformation

   A good method for normalizing data that is suffered from skewness is *Log transformation*. The log transformation is, arguably, the most popular among the different types of transformations used to transform skewed data to approximately conform to normality. If the original data follows a log-normal distribution or approximately so, then the log-transformed data follows a normal or near normal distribution. [34]

2. Sqrt Root Transformation

   The square root, x to $\sqrt{x} = sqrt(x)$, is a transformation with a moderate effect on distribution shape: it is weaker than the logarithm and the cube root. It is also used for reducing right skewness, and also has the advantage that it can be applied to zero values. Note that the square root of an area has the units of a length. It is commonly applied to counted data, especially if the values are mostly rather small.

   The square, x to x^2, has a moderate effect on distribution shape and it could be used to reduce left skewness. In practice, the main reason for using it is to fit a response by a quadratic function $y = a + bx + c\,x^2$. Quadratics have a turning point, either a maximum or a minimum, although the turning point in a function fitted to data might be far beyond the limits of the observations. The distance of a body from an origin is a quadratic if that body is moving under constant acceleration, which gives a very clear physical justification for using a quadratic. Otherwise quadratics are typically used solely because they can

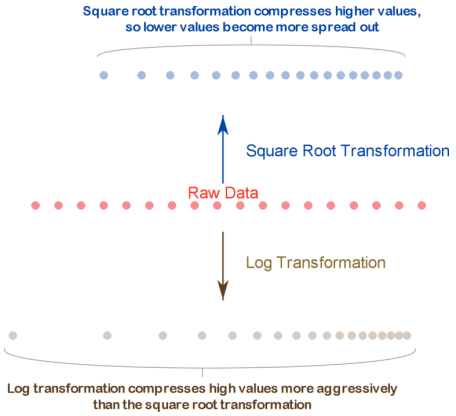


Figure 3: Square root vs Log transformation

mimic a relationship within the data region. Outside that region they may behave very poorly, because they take on arbitrarily large values for extreme values of x, and unless the intercept a is constrained to be 0, they may behave unrealistically close to the origin.

Squaring usually makes sense only if the variable concerned is zero or positive, given that $(-x)^2$ and x^2 are identical. [22] The square root transformation will not fix all skewed variables. Variables with a left skew, for instance, will become worst after a square root transformation. As discussed above, this is a consequence of compressing high values and stretching out the ones on the lower end. [23]

3. BOX-COX Transformation

   Let $y = (y_1, y_2, \ldots, y_n)'$ be the data on which the Box–Cox transformation [33] is to be applied. Box and Cox defined their transformation as

$$y_i^\lambda = \begin{cases} \lambda^{-1}(y_i^\lambda - 1) & \text{if } \lambda \neq 0 \\ log(y_i) & \text{if } \lambda = 0 \end{cases} \tag{1}$$

   Such that, for unknown λ:

$$y^{(\lambda)} = X\beta + \epsilon \tag{2}$$

   where $y^{(\lambda)}$ is the λ-transformed data, X is the design matrix (possible covariates of interest), β is the set of parameters associated with the λ-transformed data, and $\epsilon = (\epsilon_1, \epsilon_2, \ldots, \epsilon_n)'$ is the error term. Since the aim of Equation 1 is that

$$y^{(\lambda)} \sim N(X\beta, \sigma^2 I_n) \tag{3}$$

   then $\epsilon \sim N(0, \sigma^2)$. Note that the transformation in Equation 1 is only valid for $y_i > 0, i = 1, 2, \ldots, n$, and modifications have to be made when negative observations are present. [3]

Table 3 claims that the column of *balance* has the highest and *day* has the lowest skewness value among others. Moreover, it shows that *Log transformation* is an effective transformation for the columns that have skewness.

2.3 Outliers

An outlier is an observation that lies an abnormal distance from other values in a random sample from a population. In a sense, this definition leaves it up to the analyst (or a consensus process) to decide what will be considered abnormal. Before abnormal observations can be singled out, it is necessary to characterize normal observations.

Two activities are essential for characterizing a set of data:

1. Examination of the overall shape of the graphed data for important features, including symmetry and departures from assumptions.

2. Examination of the data for unusual observations that are far removed from the mass of data. These points are often referred to as outliers. Two graphical techniques for identifying outliers, scatter plots and box plots, along with an analytic procedure for detecting outliers when the distribution is normal.

The box plot is a useful graphical display for describing the behavior of the data in the middle as well as at the ends of the distributions. The box plot uses the median and the lower and upper quartiles (defined as the 25th and 75th percentiles). If the lower quartile is Q1 and the upper quartile is Q3, then the difference (Q3 - Q1) is called the interquartile range or IQ. A box plot is constructed by drawing a box between the upper and lower quartiles with a solid line drawn across the box to locate the median. The following quantities (called fences) are needed for identifying extreme values in the tails of the distribution:

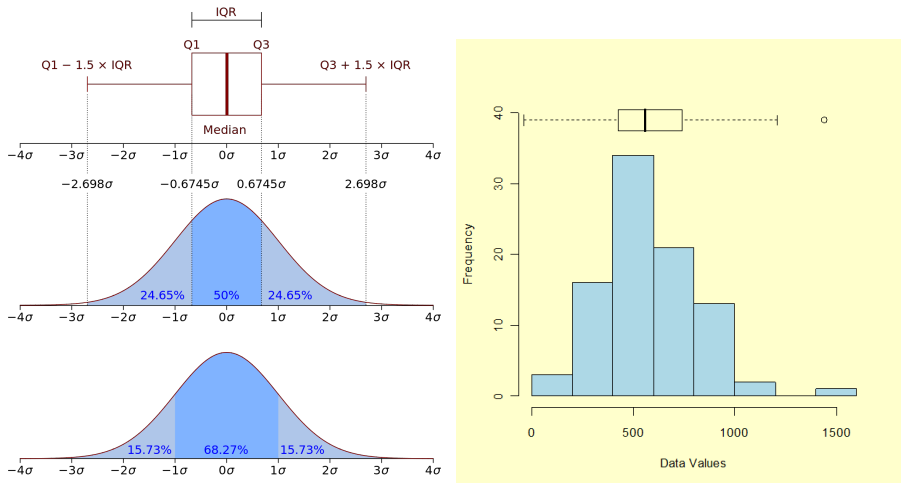
1. lower inner fence: Q1 - 1.5*IQ

2. upper inner fence: Q3 + 1.5*IQ

3. lower outer fence: Q1 - 3*IQ

4. upper outer fence: Q3 + 3*IQ

A point beyond an inner fence on either side is considered a mild outlier. A point beyond an outer fence is considered an extreme outlier. A histogram with an overlaid box plot are shown in Fig. 4b



(a) (b)

Figure 4: The outlier is identified as the largest value in the data set, 1441, and appears as the circle to the right of the box plot.

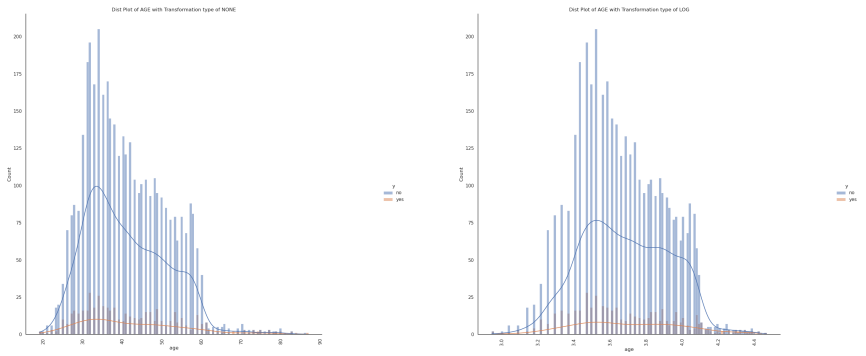
Outliers should be investigated carefully. Often they contain valuable information about the process under investigation or the data gathering and recording process. Before considering the possible elimination of these points from the data, one should try to understand why they appeared and whether it is likely similar values will continue to appear. Of course, outliers are often bad data points. [31]

2.4 Data Exploration

In this section, I am going to consider all feature columns in the dataset.

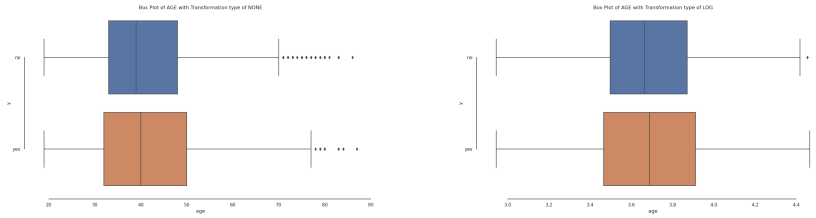
1. Age

   According to the Table 1, The *age* of customers are in the range between [19, 87]. The mean age of customers is 40 and the Fig. 5 shows that the values are not distributed normaly and *log* transformation help to handle and decrease the skewness of data. Furthermore, most of the customers lie between age of 20 and 60



(a) Skewness of *Age* is approximately *0.69* (b) Reduce skewness of *Age* with *log* transformation

Figure 5: Distribuation plots of *Age*



(a) Before applying log transformation (b) The values after log transformation

Figure 6: Box plot of Age

2. Job

Value	Count
management	969
blue-collar	946
technician	768
admin	478
services	417
retired	230
self-employed	183
entrepreneur	168
unemployed	128
housemaid	112
student	84
unknown	38

Table 4: Count values of Job

Fig 7 shows the distribution of categories in the column of *Job*. These values are not ordinal and According to the Table 4, the *unknown* category has a low number of values

and we can consider it as an outlier. In addition, most of *unknown* values are in majority class so we can remove them from dataset.

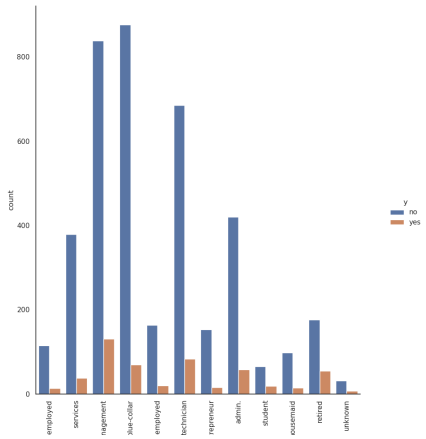


Figure 7: Job Categories

Regarding the Fig. 7 in our dataset:

(a) Customers with 'blue-collar' and 'services' jobs are less likely to subscribe for term deposit.

(b) Less number of students and more number of management and technician customers would like to subscribe.

3. Marital

The column of *Marital* status is categorical and has non-ordinal values of *divorced*(*divorced* means divorced or widowed), *married*, *single*.

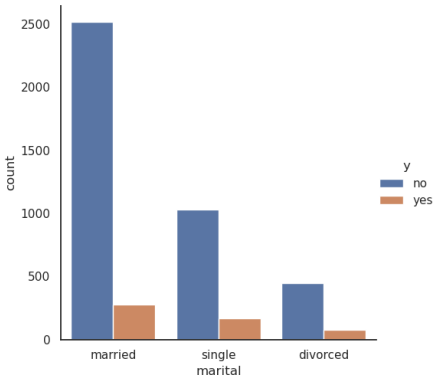
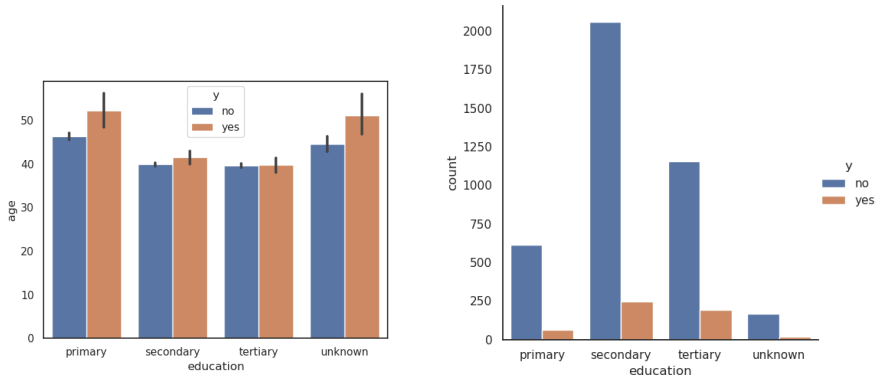


Figure 8: marital Categories

4. Education

At this column, we have four ordinal values such as: *primary*, *secondary*, *tertiary*, *unknown*. According to Fig. 9b, the education level of most customers is secondary.

The value of *unknown* can be considered as outlier and removed from the dataset. Although due to low number of samples for this category, deleting them can be useful for our model. Regarding to the Fig. 9a, the representation of *unknown* column with respect to the age is very similar to the *primary*. So, it is an another alrternative to assume *unknown* category as *primary* value.



(a) Age VS Education (b) Categories in this dataset

Figure 9: Education Categories

5. Default

   This is a categorical feature with binary values of *yes* and *no*. The *yes* means that customers had credits in their account previously otherwise *no* means that they don't have any credit in their accounts. Based on Fig. 10, most of the customers didn't have any credits in their accounts.

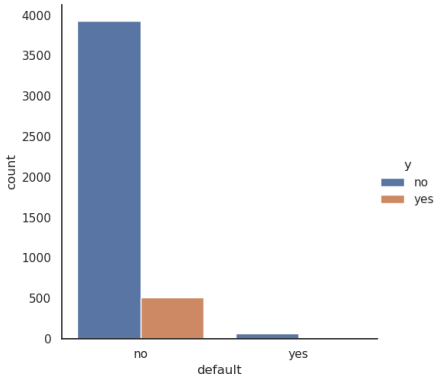


Figure 10: Defaults Categories

6. Balance

   This is a numerical feature and it holds the average yearly balance of customers in euros. This is one of our high skew features. I try to use *Log Transformaation* to decrease the skewness of data. Fig. 11 shows that although after applying *Log Transformaation* the skewness is decreased, still we have skewness in our data.

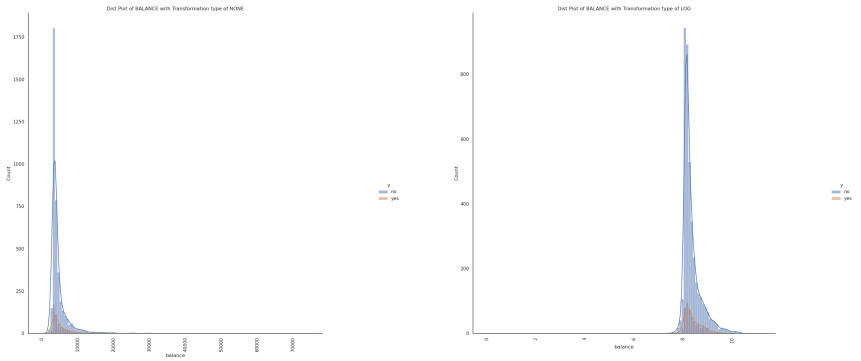
   By looking at Fig. 12a, we can consider some of values as outlier but after transforming these data, we will have less outliers than before.

7. Housing

   This is another binary value feature that represents whether a customer had house loan or not. In this regard, based on Fig. 13 there are more than 50% of customers have taken housing loan.

8. Loan

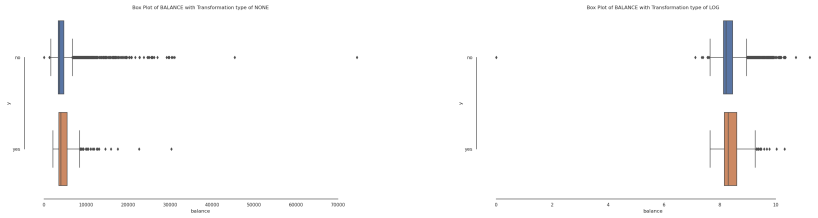
   The last bank client personal feature is *loan*. *yes* means that a customer has personal loans. As what is presented in Fig. 14, nearly 85% of customers have not taken personal loan.



(a) The skewness is 6.59

(b) After applying log transformation, the skewness is 1.88

Figure 11: Balance Distribution plots



(a) we can see some outlier

(b) we can remove outliers after transformation

Figure 12: Box plot representation of Balance

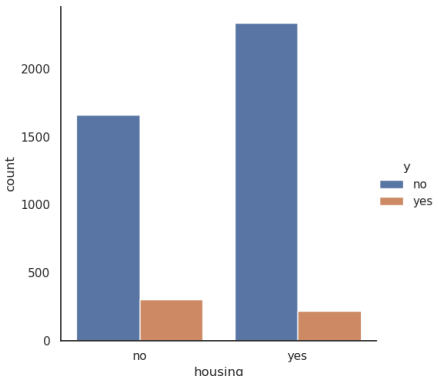


Figure 13: Has housing loan?

9. Contact

   The first feature that is related with the last contact of the current campaign is *contact*. This is a categorical feature with values of *cellular*, *telephone*. Another category is *unknown* which means the type of communication is not provided.

10. Day

    day is the second numerical feature that represents last contact day of the month to a customer. Fig. 16 shows the distribution of data and also we can conclude that in the middle of a month, customers were contacted mostly.

11. Month

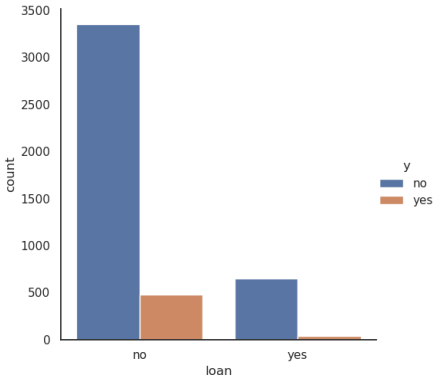


Figure 14: Has personal loan?

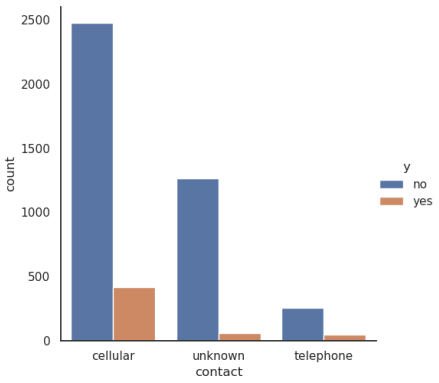
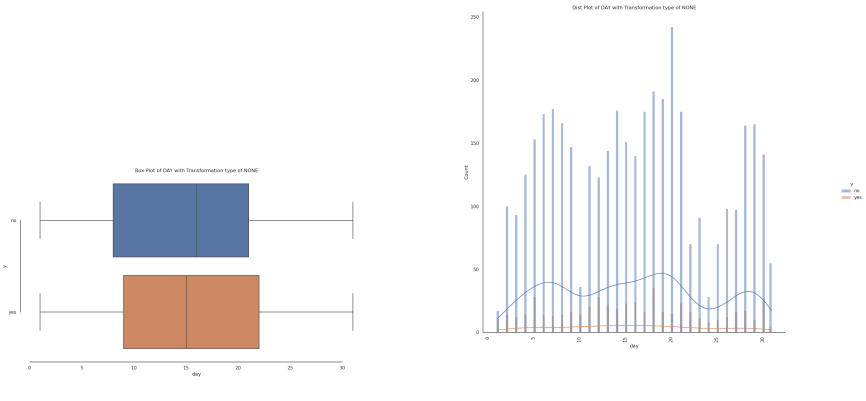


Figure 15: Contact Categories



(a) (b)

Figure 16: Day

month belongs to the features of the last contact to a customer with respect to month of the year. Obviously, by looking at Fig. 17, we can assume that most of the customers were last contacted in the month of May and at first six months of the year, we have more subscribers.

12. Duration

duration is a numerical feature and it refers to last contact duration (in second). This attribute highly affects the output target as what we can see in Fig. 18a, many target values of *y='no'* has *duration=0*. Yet, the duration is not known before a call is performed.

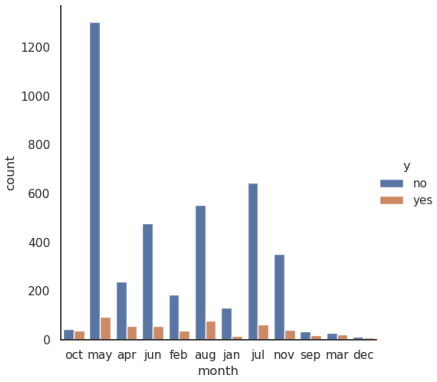
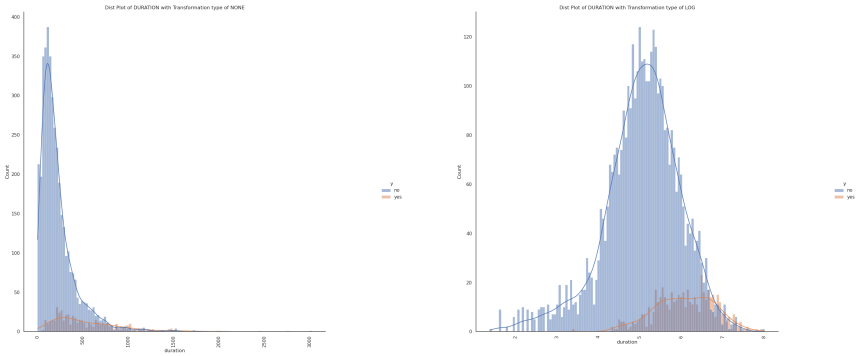


Figure 17: Last contact month of year

Also, after the end of the call y is obviously known. Thus, this input may only be included for benchmark purposes and should be discarded if the intention is to have a realistic predictive model.

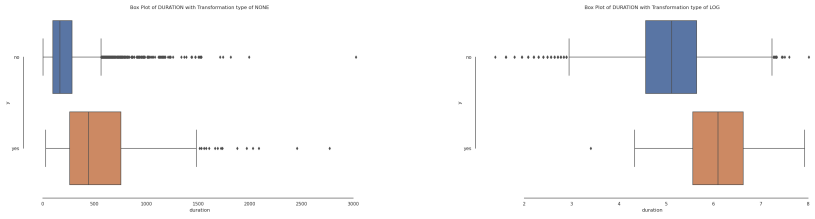


(a) Duration has about 2.77 skewness

(b) By applying log transformation, the skewness is decreased to 1.01

Figure 18: Duration distribution plots

Fig. 21a shows that customers who didn't like to speak or hear about the offers ended the calls soon. We can see that the mean of *y='no'* is less than about 250 seconds in the calls.



(a)

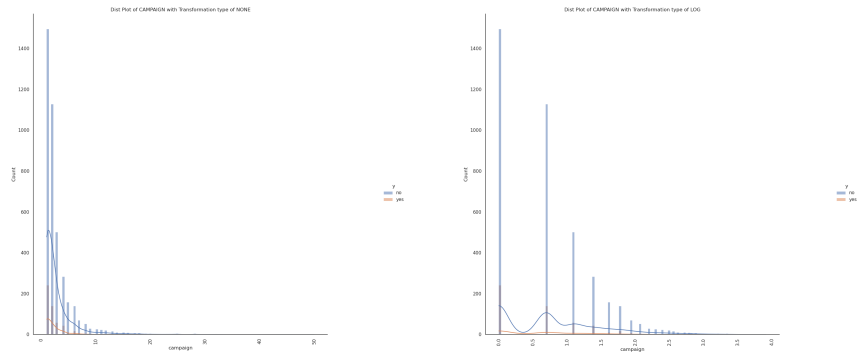
(b)

Figure 19: Duration Box plots

13. Campaign

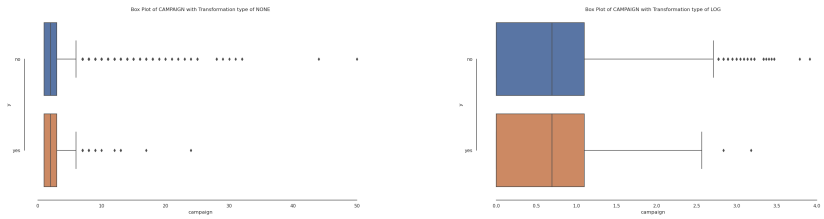
campaign is number of contacts performed during this campaign and for a client. This is a numerical feature and includes last contact. Most of customers were contacted less than 3 times in previous campaign.

According to Fig. 21a, some customers were contacted about 50 times. In my point of view, This is something strange to contact 50 times during a campagin. So, I consider these values as an outlier.



(a) Campaign has about 4.74 skewness (b) After log transformation, it is decreased to 1.55

Figure 20: Campaign distribution plots

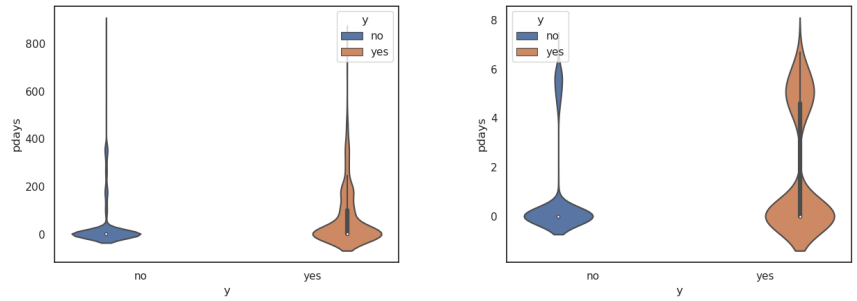


(a) (b)

Figure 21: Campaign Box plots

14. Pdays

This is the number of days that passed by after the client was last contacted from a previous campaign. Also it is numeric and -1 possibly means that the client wasn't contacted before or stands for missing data. The skewness of *pdays* is 2.71 and After applying log transformation, it is decreased to 0.99.



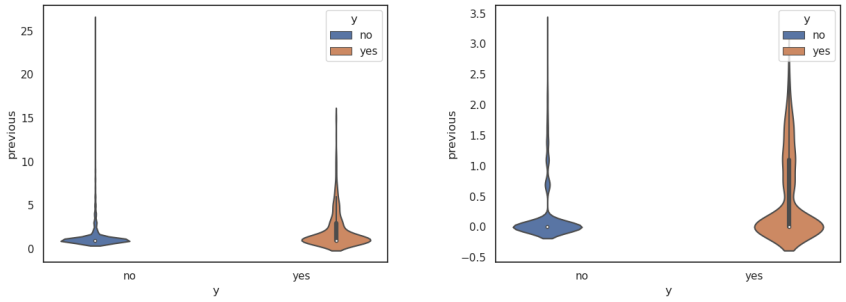
(a) Before transformation, most of values are around 0 (b) After log transformation, The distribution of data is changed and reduce the skewness

Figure 22: Pday Violin plot

Fig. 22 shows that most customers were contacted less than 39 days. Since we are not sure exactly what -1 means therefore, I suggest dropping this column, because -1 makes more than 50% of the values of the column.

15. Previous

    This is another numerical feature that is related to the number of contacts performed before this campaign and for this client.



(a) this feature has about 5.87 skewness (b) After Log transformaion the skewness is decreased to 1.77

Figure 23: Previous Violin plot

16. Poutcome

    poutcome refers to outcome of the previous marketing campaign. This is the last feature related to the campaign data. Also the categorical values inside this column are *"unknown"*, *"other"*, *"failure"*, *"success"*.

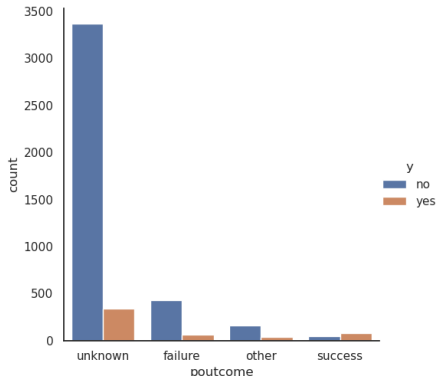


Figure 24: Poutcome Categories

The point is that *unknown* values are the most frequent values inside this feature. In my oppinion, it's better to ignore this column feature due to the large number of *unknown* values.

17. Y

    This is our target column. It is a binary feature with *"yes","no"* values. *yes* means a client subscribed a term deposit otherwise, *no* means clients didn't like to subscribe it.

    Fig. 25 shows that our dataset is imbalance and we should consider this point for building our model.

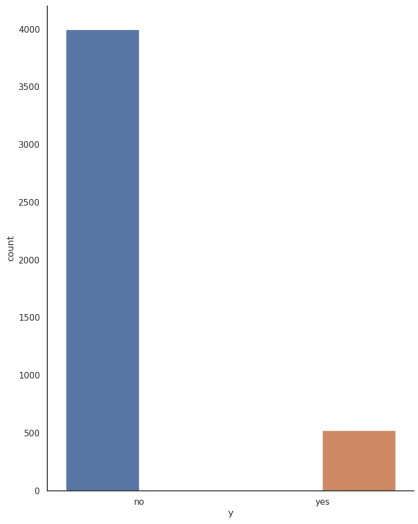


Figure 25: Target Values

3 Data Prepration

Previously, we speak about the data, and the quality of data that we have. In this section, we are going to consider data prepration steps.

3.1 Handling categorical features

Machine learning algorithms needs all inputs to be numeric and all categorical features should be converted to numerical values. So for this reason, we need to use categorical feature *Encoders* in order to convert string values to numerical ones. In the following, I will introduce a couple of these encoders briefly.

1. Label Encoding

   This encoder target labels with value between [0, # classes - 1]. As an example if you have data like *['red','green','blue']*, it will be converted to *[0,1,2]*. We should pay atttention that This transformer should be used to encode target values, i.e. y, and not the input X. [14]

$$[red, green, blue] \xrightarrow{LabelEncoder} [0, 1, 2] \tag{4}$$

2. One Hot Encoding

   Encode categorical features as a one-hot numeric array. The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. The features are encoded using a one-hot (aka 'one-of-K' or 'dummy') encoding scheme. This creates a binary column for each category and returns a sparse matrix or dense array (depending on the sparse parameter) By default, the encoder derives the categories based on the unique values in each feature. Alternatively, you can also specify the categories manually. This encoding is needed for feeding categorical data to many scikit-learn estimators, notably linear models and SVMs with the standard kernels. Note: a one-hot encoding of y labels should use a LabelBinarizer instead.[16]

   For instance, the previous example of will be:

$$[red, green, blue] \xrightarrow{One-HotEncoder} [[[1, 0, 0], [0, 1, 0], [0, 0, 1]]] \tag{5}$$

3. Binary Encoding

   Binary encoding is a combination of Hash encoding and one-hot encoding. In this encoding scheme, the categorical feature is first converted into numerical using an ordinal encoder. Then the numbers are transformed in the binary number. After that binary value is split into different columns. Binary encoding works really well when there are a high number of categories. For example the cities in a country where a company supplies its products. [10]

$$[red, green, blue] \xrightarrow{BinaryEncoder} [[[1,0,0],[0,1,0],[0,1,1]]] \tag{6}$$

4. Ordinal Encoding

   Encode categorical features as an integer array. The input to this transformer should be an array-like of integers or strings, denoting the values taken on by categorical (discrete) features. features are converted to ordinal integers. This results in a single column of integers (0 to n_categories - 1) per feature.[17]

$$[one, two, three] \xrightarrow{OrdinalEncoder} [0,1,2] \tag{7}$$

Table 5 shows encoders that we use for the categorical features.

Categorical Features	Type	Encoder
Job	Non-Ordinal	Binary
Marital	Non-Ordinal	Binary
Education	Ordinal	Ordinal
Default	Binary	Binary
Housing	Binary	Binary
Loan	Binary	Binary
Contact	Non-Ordinal	Binary
Month	Ordinal	Ordinal
Poutcome	Non-Ordinal	Binary
Y	Binary	Binary

Table 5: Encoders for categorical feature

3.2 Discretization Transforms

A discretization transform will map numerical variables onto discrete values. Binning, also known as categorization or discretization, is the process of translating a quantitative variable into a set of two or more qualitative buckets (i.e., categories) Values for the variable are grouped together into discrete bins and each bin is assigned a unique integer such that the ordinal relationship between the bins is preserved. [7]

I decided to convert *age* and *duration* columns' values to discrete bins that I describe below:

- *age* will be:
  1. $(age <= 32) \rightarrow 1$
  2. $(age > 32) \quad and \quad (age <= 47) \rightarrow 2$
  3. $(age > 47) \quad and \quad (age <= 70) \rightarrow 3$
  4. $(age > 70) \rightarrow 4$
- *duration* will be:
  1. $(duration <= 102) \rightarrow 1$
  2. $(duration > 102) \quad and \quad (duration <= 180) \rightarrow 2$

3. $(duration > 180) \quad and \quad (duration <= 319) \rightarrow 3$
4. $(duration > 319) \quad and \quad (duration <= 644.5) \rightarrow 4$
5. $(duration > 644.5) \rightarrow 5$

3.3 Normalization and Standardization

The two most popular techniques for scaling numerical data prior to modeling are normalization and standardization. *Normalization* scales each input variable separately to the range 0-1, which is the range for floating-point values where we have the most precision. *Standardization* scales each input variable separately by subtracting the mean (called centering) and dividing by the standard deviation to shift the distribution to have a mean of zero and a standard deviation of one.

1. Normalization:

   Normalization is a rescaling of the data from the original range so that all values are within the new range of 0 and 1.

$$z_i = \frac{x_i - min(x)}{max(x) - min(x)} \tag{8}$$

2. Standardizing:

   Standardizing a dataset involves rescaling the distribution of values so that the mean of observed values is 0 and the standard deviation is 1. in this equation, μ is mean of sample, σ is standard deviation of samples, x is observed values and Z is standard score.[24]

$$Z = \frac{x - \mu}{\sigma} \tag{9}$$

For this pipeline, in preprocessing step, I used *Standard Scaler* to change the distribution of values on this dataset.

3.4 How to handle imbalance dataset

Apart from using different evaluation criteria, one can also work on getting different dataset. Two approaches to make a balanced dataset out of an imbalanced one are under-sampling and over-sampling.

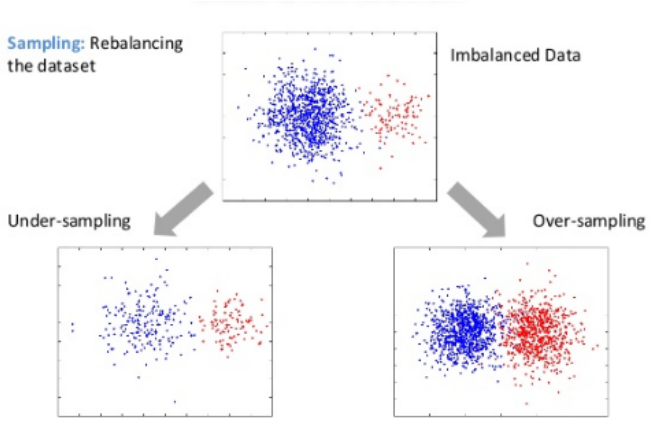
1. Under-sampling

   Under-sampling balances the dataset by reducing the size of the abundant class. This method is used when quantity of data is sufficient. By keeping all samples in the rare class and randomly selecting an equal number of samples in the abundant class, a balanced new dataset can be retrieved for further modelling.

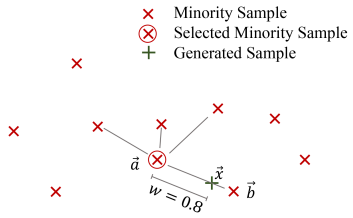
2. Over-sampling

   On the contrary, oversampling is used when the quantity of data is insufficient. It tries to balance dataset by increasing the size of rare samples. Rather than getting rid of abundant samples, new rare samples are generated by using e.g. SMOTE (Synthetic Minority Over-Sampling Technique)[20]

   SMOTE first selects a minority class instance a at random and finds its k nearest minority class neighbors. The synthetic instance is then created by choosing one of the k nearest neighbors b at random and connecting a and b to form a line segment in the feature space. The synthetic instances are generated as a convex combination of the two chosen instances a and b. [27]



(a) Over-sampling VS Under-sampling[9]



(b) SMOTE methodology[35]

Figure 26: Sampling Strategy

3.5 Principal Components Analysis (PCA)

In simple words, PCA is a method of obtaining important variables (in form of components) from a large set of variables available in a data set. It extracts low dimensional set of features by taking a projection of irrelevant dimensions from a high dimensional data set with a motive to capture as much information as possible. With fewer variables obtained while minimising the loss of information, visualization also becomes much more meaningful. PCA is more useful when dealing with 3 or higher dimensional data

- Let $\mathbf{X}$ be a data matrix with n samples, and p variables.
- From each variable, we subtract the mean of the column; i.e. we *center* the variables.
- To find the first principal component $\phi_1 = (\phi_{11}, \ldots, \phi_{p1})$, we solve the following optimization:

$$\max_{\phi_{11},\ldots,\phi_{p1}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} x_{ij}\phi_{j1} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j1}^2 = 1.$$

- The quantity:

$$\sum_{j=1}^{p} x_{ij}\phi_{j1} = X\phi_1$$

- Summing the square of the entries of Z computes the variance of the n samples projected onto ϕ_1. (This is a variance, because we have centered the columns.)
- Having found the maximizer $\hat{\phi}_1$, the *first* principal component *score* is

$$Z_1 = X\hat{\phi}_1$$

- To find the second principal component $\phi_2 = (\phi_{12}, \ldots, \phi_{p2})$, we solve the following optimization

$$\max_{\phi_{12},\ldots,\phi_{p2}} \left\{ \frac{1}{n} \sum_{i=1}^{n} \left(\sum_{j=1}^{p} \phi_{j2} x_{ij} \right)^2 \right\} \text{ subject to } \sum_{j=1}^{p} \phi_{j2}^2 = 1 \quad \text{and} \quad \sum_{j=1}^{p} \hat{\phi}_{j1} \phi_{j2} = 0.$$

- The second constraint $\hat{\phi}_1' \phi_2 = 0$ forces first and second principal components to be orthogonal.
- Having found the optimal $\hat{\phi}_2$, the *second* principal component *score*

$$Z_2 = X\hat{\phi}_2$$

- It turns out that the constraint $\hat{\phi}_1' \phi_2 = 0$ implies that the scores $Z_1 = (z_{11}, \ldots, z_{n1})$ and $Z_2 = (z_{12}, \ldots, z_{n2})$ are uncorrelated. [18]

4 Methodology

We are going to build a classification model in order to predict clients who will subscribe to deposits in our campaign. By exploring the dataset, we know that our data is imbalance and we should use *Resampling* methods or exploit algorithms that imbalance data won't affect on them e.g. *SVM*.

First, we should convert categorical features to numerical values that is shown in Table 5 then, we change the scale of our data.

4.1 Metrics

The metric that helps us to evaluate our model performance is *f1 score*. The f1 score can be interpreted as a weighted average of the precision and recall 27, where an f1 score reaches its best value at 1 and worst score at 0. The relative contribution of precision and recall to the f1 score are equal. The formula for the f1 score is: [8]

$$F1 = 2 * (precision * recall)/(precision + recall)$$

Another approach to evaluate our model is *cross-validation*. More precisely, Cross-validation is a resampling procedure used to evaluate machine learning models on a limited data sample. The procedure has a single parameter called k that refers to the number of groups that a given data sample is to be split into. As such, the procedure is often called k-fold cross-validation. When a specific value for k is chosen, it may be used in place of k in the reference to the model, such as k=10 becoming 10-fold cross-validation. The general procedure is as follows:[4]

- Shuffle the dataset randomly.
- Split the dataset into k groups
- For each unique group:
    1. Take the group as a hold out or test data set
    2. Take the remaining groups as a training data set
    3. Fit a model on the training set and evaluate it on the test set
    4. Retain the evaluation score and discard the model
- Summarize the skill of the model using the sample of model evaluation scores

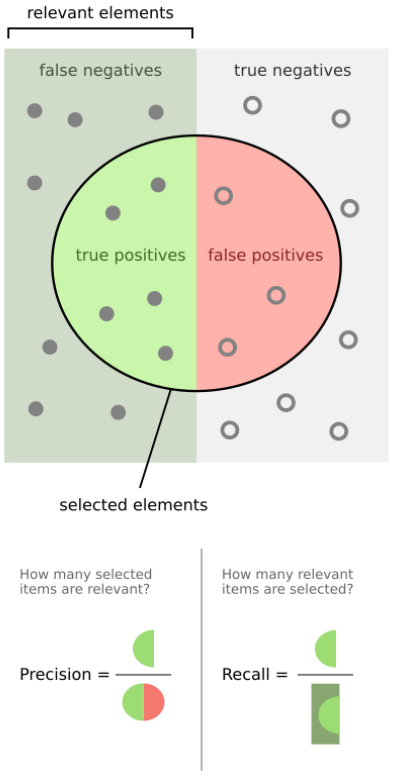


Figure 27: Precision and Recall

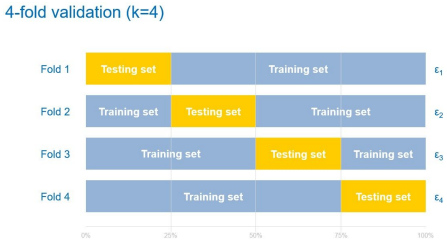


Figure 28: Cross validation

4.2 Hyperparameter optimization

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. A hyperparameter is a parameter whose value is used to control the learning process. By contrast, the values of other parameters (typically node weights) are learned.

The traditional way of performing hyperparameter optimization has been *grid search*, or a parameter sweep, which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.

Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

For example, a typical soft-margin SVM classifier equipped with an RBF kernel has at least two hyperparameters that need to be tuned for good performance on unseen data: a regularization constant C and a kernel hyperparameter γ. Both parameters are continuous, so to perform

grid search, one selects a finite set of "reasonable" values for each, say

$$C \in \{10, 100, 1000\}$$
$$\gamma \in \{0.1, 0.2, 0.5, 1.0\}$$

Grid search then trains an SVM with each pair (C, γ) in the Cartesian product of these two sets and evaluates their performance on a held-out validation set (or by internal cross-validation on the training set, in which case multiple SVMs are trained per pair). Finally, the grid search algorithm outputs the settings that achieved the highest score in the validation procedure.

Grid search suffers from the curse of dimensionality, but is often embarrassingly parallel because the hyperparameter settings it evaluates are typically independent of each other.[12]

4.3 Feature importance

Feature importance refers to techniques that assign a score to input features based on how useful they are at predicting a target variable. Feature importance scores play an important role in a predictive modeling project, including providing insight into the data, insight into the model, and the basis for dimensionality reduction and feature selection that can improve the efficiency and effectiveness of a predictive model on the problem.

We can fit a LogisticRegression model on the regression dataset and retrieve the *coeff_* property that contains the coefficients found for each input variable. These coefficients can provide the basis for a crude feature importance score. This assumes that the input variables have the same scale or have been scaled prior to fitting a model. Notice that the coefficients are both positive and negative. The positive scores indicate a feature that predicts positive class, whereas the negative scores indicate a feature that predicts negative class.[11]

According to the Fig. 29a, the most important features that are extracted from *Decision Tree Classifier* are in the following order: *duration*, *day*, *balance*, *month*, *job*, *previous*, *campaign*, *education*, *poutcome*, *age*, *pdays*, *marital*, *contact*, *housing*, *loan*, *default*.

Also, based on Fig, 29b, The features that are important for *Random Forest Classifier* are: *duration*, *balance*, *day*, *month*, *job*, *pdays*, *campaign*, *poutcome*, *age*, *education*, *marital*, *previous*, *housing*, *contact*, *loan*, *default*.

Fig. 29c shows that the most important feature for positive class are: *duration*, *pdays*, *previous*, *poutcome*, *education*, *age*, *marital*, *default*, *job*, *day*, *balance*. Moreover, the features of *contact*, *housing*, *loan*, *campaign*, *month* are important for negative class.

5 Methods

5.1 SVM

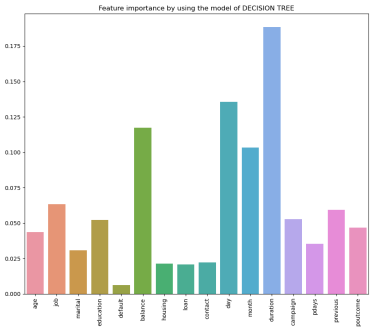
A support vector machine (SVM) is a supervised machine learning algorithm that can be used for both classification and regression tasks. In SVM, we plot data points as points in an n-dimensional space (n being the number of features you have) with the value of each feature being the value of a particular coordinate. [30]

Hyperplanes can be considered decision boundaries that classify data points into their respective classes in a multi-dimensional space. Data points falling on either side of the hyperplane can be attributed to different classes. Let's consider a two-dimensional space. The two-dimensional linearly separable data can be separated by the equation of a line—with the data points lying on either sides representing the respective classes. The function of the line is

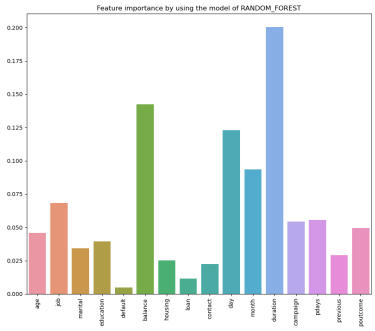
$$y = ax + by$$

. Considering x and y as features and naming them as $x_1, x_2, \ldots, x_n$, it can be re-written as:

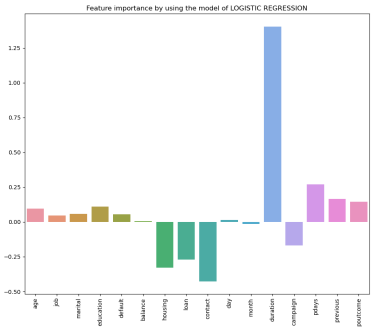
$$ax_1 - x_2 + b = 0$$



(a) Important features by Decision Tree Classifier



(b) Important features by Random Forest Classifier



(c) Important features by Logistic Regression

Figure 29: Feature Importance

If we define $x = (x_1, x_2)$ and $w = (a, -1)$, we get:

$$w \cdot x + b = 0$$

This equation is derived from two-dimensional vectors. But in fact, it also works for any number of dimensions. This is the equation for a hyperplane that is shown in Fig. 30. [30]

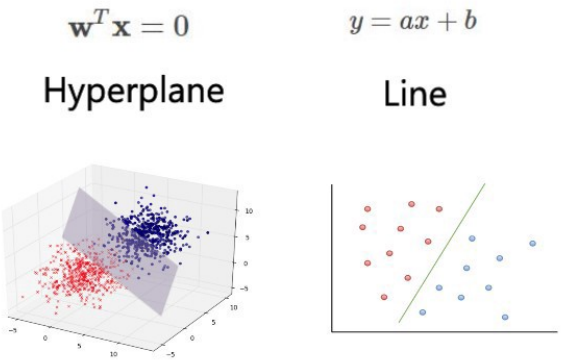


Figure 30: Hyperplane

Now we are going to speak about the hyper-parameters briefly:

1. kernel: The learning of the hyperplane in linear SVM is done by transforming the problem using some linear algebra. This is where the kernel plays role. For linear kernel the equation for prediction for a new input using the dot product between the input x and each support vector x_i is calculated as $f(x) = B(0) + sum(a_i * (x, x_i))$

   This is an equation that involves calculating the inner products of a new input vector (x) with all support vectors in training data. The coefficients B_0 and a_i (for each input) must be estimated from the training data by the learning algorithm. The polynomial kernel can be written as $K(x, x_i) = 1 + sum(x * x_i)^d$ and exponential as $K(x, x_i) = exp(-gamma * sum((x - x_i^2))$. Polynomial and exponential kernels calculates separation line in higher dimension. This is called *kernel trick*.

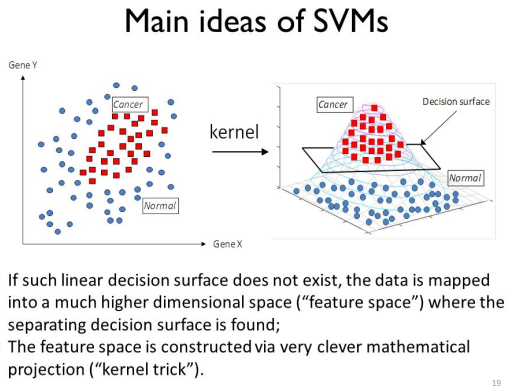


Figure 31: Kernel Trick

2. Regularization: The Regularization parameter (often termed as C parameter in python's sklearn library) tells the SVM optimization how much you want to avoid misclassifying each training example. For large values of C, the optimization will choose a smaller-margin hyperplane if that hyperplane does a better job of getting all the training points classified correctly. Conversely, a very small value of C will cause the optimizer to look for a larger-margin separating hyperplane, even if that hyperplane misclassifies more points.

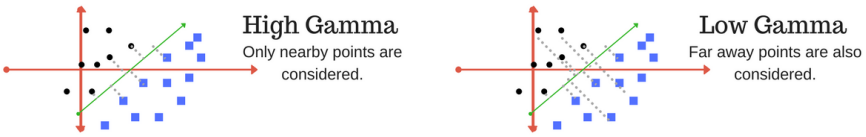
(a) Some misclassification due to lower regularization value.

(b) Higher value leads to results like right one.

Figure 32: Regularization parameter

3. Gamma: The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. In other words, with low gamma, points far away from plausible seperation line are considered in calculation for the seperation line. Where as high gamma means the points close to plausible line are considered in calculation.

4. Margin: And finally last but very importrant characteristic of SVM classifier. SVM to core tries to achieve a good margin. A margin is a separation of line to the closest class points. A good margin is one where this separation is larger for both the classes. Images below gives to visual example of good and bad margin. A good margin allows the points to be in their respective classes without crossing to other class.[26]



(a) High Gamma (b) Low Gamma

Figure 33: Gama parameter

(a) Good margin (b) Bad margin

Figure 34: Margin parameter

The advantages of support vector machines are: [25]

- Effective in high dimensional spaces.
- Still effective in cases where number of dimensions is greater than the number of samples.
- Uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.
- Versatile: different Kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

The disadvantages of support vector machines include:

- If the number of features is much greater than the number of samples, avoid over-fitting in choosing Kernel functions and regularization term is crucial.
- SVMs do not directly provide probability estimates, these are calculated using an expensive five-fold cross-validation .

5.2 Decision Tree Classifier

Decision Tree is a Supervised learning technique that can be used for both classification and Regression problems, but mostly it is preferred for solving Classification problems. It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome. In a Decision tree, there are two nodes, which are the Decision Node and Leaf Node. Decision nodes are used to make any decision and have multiple branches, whereas Leaf nodes are the output of those decisions and do not contain any further branches. The decisions or the test are performed on the basis of features of the given dataset. It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions. It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like structure. In order to build a tree, we use the CART algorithm, which stands for Classification and Regression Tree algorithm. A decision tree simply asks a question, and based on the answer (Yes/No), it further split the tree into subtrees.

Decision Tree Terminologies are

- Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

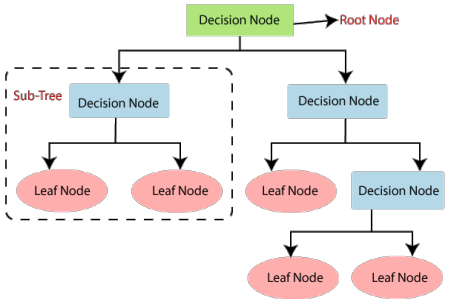


Figure 35: Diagram explains the general structure of a decision tree

- Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.
- Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.
- Branch/Sub Tree: A tree formed by splitting the tree.
- Pruning: Pruning is the process of removing the unwanted branches from the tree.
- Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

In a decision tree, for predicting the class of the given dataset, the algorithm starts from the root node of the tree. This algorithm compares the values of root attribute with the record (real dataset) attribute and, based on the comparison, follows the branch and jumps to the next node. For the next node, the algorithm again compares the attribute value with the other sub-nodes and move further. It continues the process until it reaches the leaf node of the tree. The complete process can be better understood using the below algorithm:[5]

- Step-1: Begin the tree with the root node, says S, which contains the complete dataset.
- Step-2: Find the best attribute in the dataset using Attribute Selection Measure (ASM).
- Step-3: Divide the S into subsets that contains possible values for the best attributes.
- Step-4: Generate the decision tree node, which contains the best attribute.
- Step-5: Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

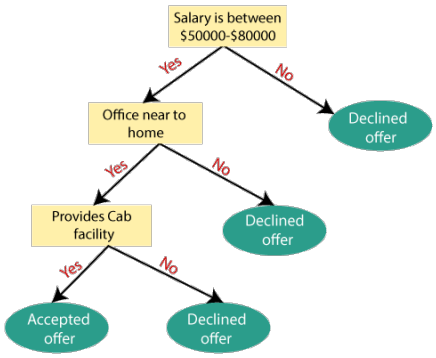


Figure 36: Example of a decision tree

Attribute selection measure is a heuristic for selecting the splitting criterion that partition data into the best possible manner. It is also known as splitting rules because it helps us to determine breakpoints for tuples on a given node. ASM provides a rank to each feature(or attribute) by explaining the given dataset. Best score attribute will be selected as a splitting attribute. In the case of a continuous-valued attribute, split points for branches also need to define. Most popular selection measures are Information Gain, Gain Ratio, and Gini Index.[6] Some advantages of decision trees are:[28]

- Simple to understand and to interpret. Trees can be visualised.
- Requires little data preparation. Other techniques often require data normalisation, dummy variables need to be created and blank values to be removed. Note however that this module does not support missing values.
- The cost of using the tree (i.e., predicting data) is logarithmic in the number of data points used to train the tree.
- Able to handle both numerical and categorical data. However scikit-learn implementation does not support categorical variables for now. Other techniques are usually specialised in analysing datasets that have only one type of variable. See algorithms for more information.
- Able to handle multi-output problems.
- Uses a white box model. If a given situation is observable in a model, the explanation for the condition is easily explained by boolean logic. By contrast, in a black box model (e.g., in an artificial neural network), results may be more difficult to interpret.
- Possible to validate a model using statistical tests. That makes it possible to account for the reliability of the model.

The disadvantages of decision trees include:

- Decision-tree learners can create over-complex trees that do not generalise the data well. This is called overfitting. Mechanisms such as pruning, setting the minimum number of samples required at a leaf node or setting the maximum depth of the tree are necessary to avoid this problem.
- Decision trees can be unstable because small variations in the data might result in a completely different tree being generated. This problem is mitigated by using decision trees within an ensemble.
- Predictions of decision trees are neither smooth nor continuous, but piecewise constant approximations as seen in the above figure. Therefore, they are not good at extrapolation.
- The problem of learning an optimal decision tree is known to be NP-complete under several aspects of optimality and even for simple concepts. Consequently, practical decision-tree learning algorithms are based on heuristic algorithms such as the greedy algorithm where locally optimal decisions are made at each node. Such algorithms cannot guarantee to return the globally optimal decision tree. This can be mitigated by training multiple trees in an ensemble learner, where the features and samples are randomly sampled with replacement.
- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems.
- Decision tree learners create biased trees if some classes dominate. It is therefore recommended to balance the dataset prior to fitting with the decision tree.

5.3 Random Forest Classifier

Random forests or random decision forests are an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For classification tasks, the output of the random forest is the class selected by most trees. For regression tasks, the mean or average prediction of the individual trees is returned. Random decision forests correct for decision trees' habit of overfitting to their training set.[3]:587–588 Random forests generally outperform decision trees, but their accuracy is lower than gradient boosted trees. However, data characteristics can affect their performance.[19]

Random forest uses gini importance or mean decrease in impurity (MDI) to calculate the importance of each feature. Gini importance is also known as the total decrease in node impurity. This is how much the model fit or accuracy decreases when you drop a variable. The larger the decrease, the more significant the variable is. Here, the mean decrease is a significant parameter for variable selection. The Gini index can describe the overall explanatory power of the variables.

It works in four steps:[29]

1. Select random samples from a given dataset.
2. Construct a decision tree for each sample and get a prediction result from each decision tree.
3. Perform a vote for each predicted result.
4. Select the prediction result with the most votes as the final prediction.

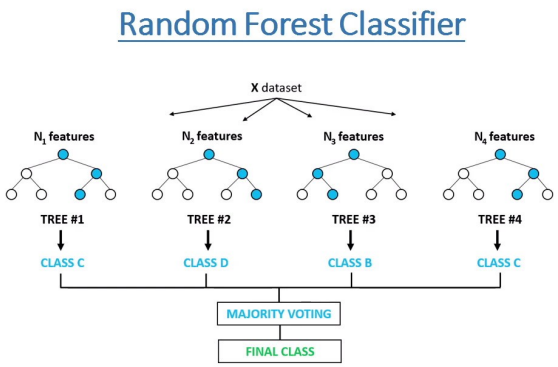


Figure 37: Random forest Classifier and majority voting

Advantages of Random forest:

- Random forests is considered as a highly accurate and robust method because of the number of decision trees participating in the process.
- It does not suffer from the overfitting problem. The main reason is that it takes the average of all the predictions, which cancels out the biases.
- The algorithm can be used in both classification and regression problems
- Random forests can also handle missing values. There are two ways to handle these: using median values to replace continuous variables, and computing the proximity-weighted average of missing values.
- You can get the relative feature importance, which helps in selecting the most contributing features for the classifier.

Disadvantages of random forest:

- Random forests is slow in generating predictions because it has multiple decision trees. Whenever it makes a prediction, all the trees in the forest have to make a prediction for the same given input and then perform voting on it. This whole process is time-consuming.
- The model is difficult to interpret compared to a decision tree, where you can easily make a decision by following the path in the tree.

5.4 Logistic Regression

A solution for classification is logistic regression. Instead of fitting a straight line or hyperplane, the logistic regression model uses the logistic function to squeeze the output of a linear equation between 0 and 1. The logistic function is defined as:

$$\text{logistic}(\eta) = \frac{1}{1 + exp(-\eta)}$$

And it looks like this:

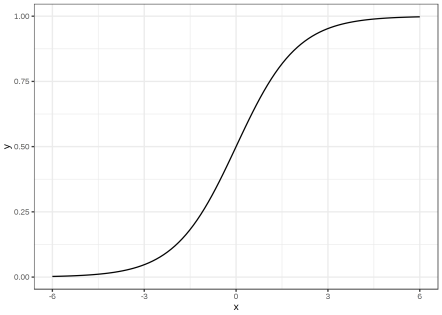


Figure 38: Random forest

The step from linear regression to logistic regression is kind of straightforward. In the linear regression model, we have modelled the relationship between outcome and features with a linear equation:

$$\hat{y}^{(i)} = \beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}$$

For classification, we prefer probabilities between 0 and 1, so we wrap the right side of the equation into the logistic function. This forces the output to assume only values between 0 and 1.

$$P(y^{(i)} = 1) = \frac{1}{1 + exp(-(\beta_0 + \beta_1 x_1^{(i)} + \ldots + \beta_p x_p^{(i)}))}$$

The interpretation of the weights in logistic regression differs from the interpretation of the weights in linear regression, since the outcome in logistic regression is a probability between 0 and 1. The weights do not influence the probability linearly any longer. The weighted sum is transformed by the logistic function to a probability. Therefore we need to reformulate the equation for the interpretation so that only the linear term is on the right side of the formula. [15]

$$log\left(\frac{P(y=1)}{1-P(y=1)}\right) = log\left(\frac{P(y=1)}{P(y=0)}\right) = \beta_0 + \beta_1 x_1 + \ldots + \beta_p x_p$$

The coefficients $\beta_0, \beta_1, \ldots, \beta_p$ are unknown, and must be estimated based on the available training data using Maximum likelihood method.

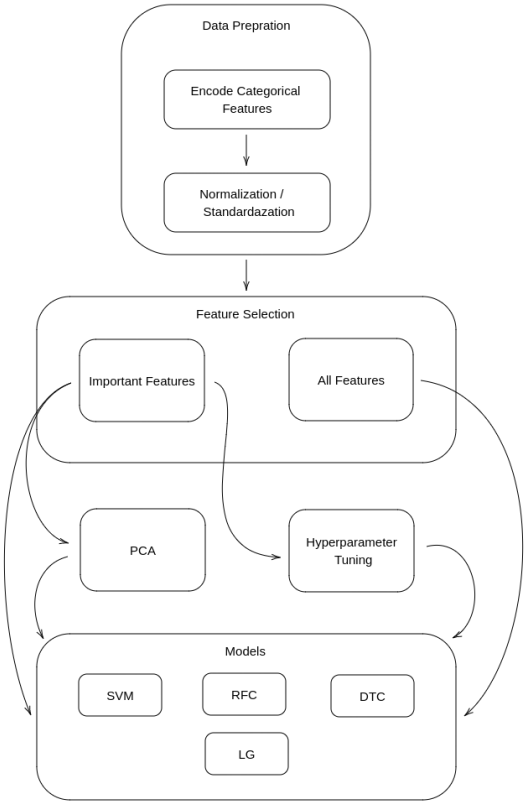


Figure 39: Project Pipeline

6 Evaluation

In this section, I am going to compare the performance of the models which is introduced before. For each model, I performed 4 different tests. At first examination, I ran the model with all features and its default hyperparameters. At the second examination, I selected the important features that are detected by models such as logistic Regression. Then, by doing hyperparameter-tuning, I found the best parameters for training my model and again I evaluated the impact of these params with important features on models. Finally, I applied PCA in order to decrease the number of important features. Fig. 39 shows the structure of this project's pipeline briefly.

In addition, I use cross validation approach in order to estimate the performance of the models that is shown in Table 6. Note that the result in this table is the best one among 5 folds.

As what you can see in Fig. 41, After applying pca to important features that were retrieved from Decision Tree classifier, the 4 best principal components are able to separate the targets more precisely. In more details, Fig. 40 interpret the Decision Tree classifier model and how this model can classify the clients by using just using 5 principal components.

According to the results that are shown in Table 6, we can conclude that using important features and principal components have a significant impacts on the classification results.

Finally, the *SVM* model obtained the score of **99%** with principal components of important features.

7 Conclusion

In conclusion, choosing a right metric for evaluation of a model is important specially, in our case, this dataset is imbalance and for this reason I selected the *f1 score* metric. Although I applied both under-sampling and oversampling techniques to balance my dataset, both of them

		Parameters	F1 Score
SVM	Cross Validation	K = 5	0.90
	All features	DEFUALTS	0.86
	Important Features	duration, poutcome, default, age	0.88
	Hyperparameter tuning	'C': 10, 'degree': 1, 'gamma': 'scale', 'kernel': 'rbf'	0.88
	PCA	IMPORTANT FEATURES	0.99
Random Forest Classifier	Cross Validation	K = 5	0.91
	All features	DEFUALTS	0.88
	Important Features	duration, day, month,job poutcome, age, education, contact, loan	0.91
	Hyperparameter tuning	'bootstrap': True, 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 2, 'min_samples_split': 5, 'n_estimators': 1000	0.90
	PCA	IMPORTANT FEATURES	0.98
Decision Tree Classifier	Cross Validation	K = 5	0.87
	All features	DEFUALTS	0.86
	Important Features	duration, day, month, previous, campaign, poutcome, pdays	0.89
	Hyperparameter tuning	'criterion': 'gini', 'max_depth': 10, 'max_features': 'auto', 'min_samples_leaf': 1, 'min_samples_split': 10, 'splitter': 'best'	0.88
	PCA	BEST FEATURES	0.99
Logistic Regression	Cross Validation	K = 5	0.90
	All features	DEFUALTS	0.86
	Important Features	duration, pdays, contact,marital loan, default, poutcome	0.89
	Hyperparameter tuning	'C': 0.1, 'max_iter': 100, 'solver': 'liblinear'	0.89
	PCA	IMPORTANT FEATURES	0.95

Table 6: Experiments results

didn't work for me, and I preferred to ignore them. As what you see in the results, the features related with the last contact of the current campaign are more important for our models. Also, using principal component analysis with important features changed the results for all models significantly.

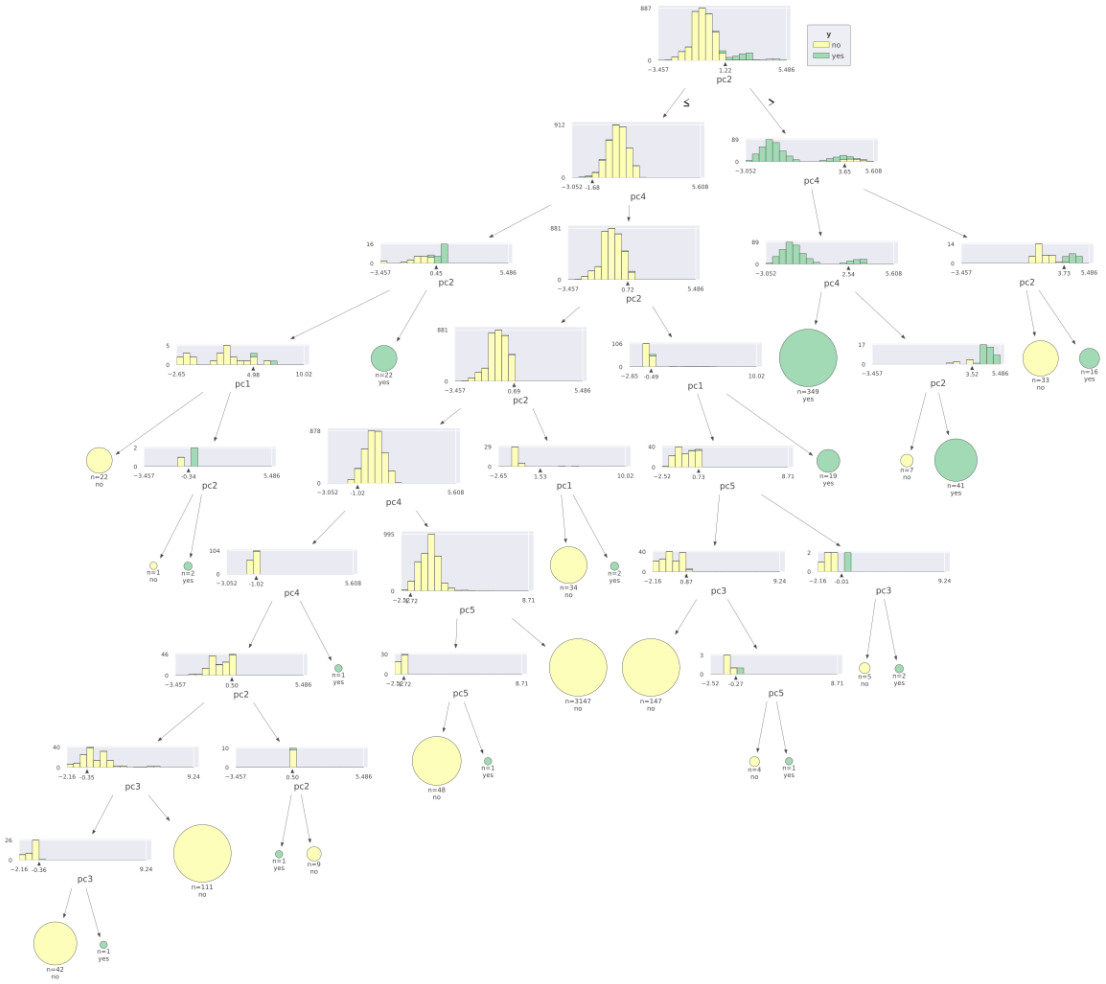


Figure 40: Decision Tree with PCA features

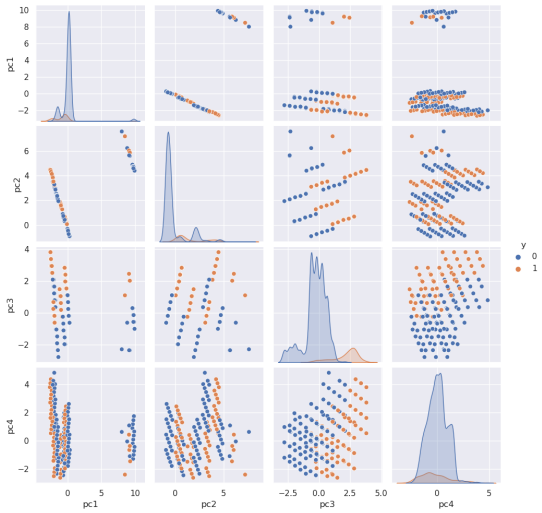


Figure 41: PCA components of important features from Decision Tree classifier

A Appendix

You can clone this project from [2]. I implemented a clean code template in order to modify the structure of this project more easily. The folders structure of this project are:

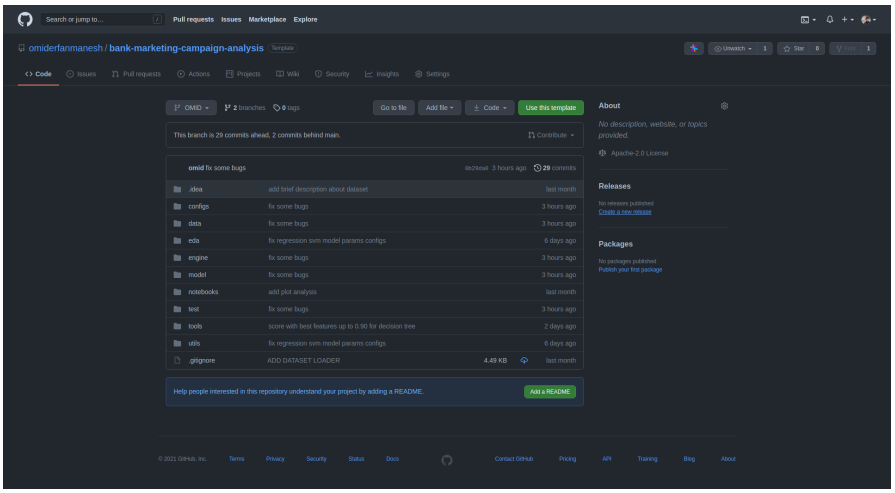


Figure 42: Project Structure

- configs
  - defaults.py: here's the specific config file for specific model or dataset.
- data
  - based
    - based_dataset.py: a super class for all dataset's class, it contains data manipulations methods
    - encoder_enum.py: contains constant of encoders
    - file_types.py: contains constant of file type e.g. CSV, TSV, TXT, ...
    - sampling_types.py: contains constant of sampling types e.g. SMOTE, ...
    - scale_types.py: contains constant of scalers e.g. MinMax Scaler
    - transformers_enums.py: contains types of data transformers e.g. Log Transformer
  - dataset: include all datasets files
  - preprocessing
    - encoders.py: *Encoder* helps to encode data and has all codes needed for this purpose
    - pca.py: *PCA* is for applying pca analysis and drawing plots
    - scalers.py: *Scaler* is for changing the scale of values in a dataset
  - bank.py: *Bank* is a sub-class of *BasedDataset* and it contains all configs to load the dataset from files
  - loader.py: it loads the dataset that we need
- eda: contains files and folders that are needed for analysing and retrieving information about the dataset, features, categories, ...
  - based
    - based_analyser.py: contains codes for retrieving information from a dataset
    - based_plots.py: contains codes related to draw plots from a dataset

  - bank_analyser.py: *BankAnalyzer* is a sub-class of *BasedAnalyzer* and it is help to analyse features
  - bank_plot.py: *BankPlots* is a sub-class of *BasedPlot* that helps to plot features' values
- engine
  - analyser.py: this file runs *BankAnalyzer*
  - trainer.py: this file is for training, hyperparameters tuning and doing cross-validation
- model
  - based
    - based_model.py: this is a super class for all models that contains methods for training, evaluation, ploting, ...
    - metric_types.py: contains constants of metrics
    - model_selection.py: contains the name of models that you can run for your dataset
    - task_mode.py: it helps you to select what task you are going to do e.g. classification or regression ...
    - tuning_mode.py: contains the types of tuining algorithms e.g GridSearch
  - decision_tree.py: contains all configuration needed to run *Decision Tree* algorithms
  - logistic_regression.py: contains all configuration needed to run *Logistic Regression* algorithm
  - random_forest.py: contains all configuration needed to run *Random forest* algorithms
  - svm.py: contains all configuration needed to run *Support vector machine* algorithms
- notebooks: contains all notebooks that is needed when you want to run this project by jupiter notebook
  - data_exploration.py
  - EDA.py
- test: test file
  - test.py
- tools: run files
  - analysis.py: contains main() for running the analyser.py
  - train.py: contains main() for running the trainer.py
- utils: all utilities files are here
  - runtime_mode.py: all types of running modes for this project e.g. training, tuning or cross-validation

References

[1] Bank marketing. https://archive.ics.uci.edu/ml/datasets/Bank+Marketing.

[2] bank-marketing-campaign-analysis. https://github.com/omiderfanmanesh/bank-marketing-campaign-analysis.git.

[3] The box–cox transformation. https://www.frontiersin.org/articles/10.3389/fams.2015.00012/full.

[4] cross validation. https://machinelearningmastery.com/k-fold-cross-validation/.

[5] Decision tree classification algorithm. https://www.javatpoint.com/machine-learning-decision-tree-classification-algorithm.

[6] Decision tree classification python. https://www.datacamp.com/community/tutorials/decision-tree-classification-python.

[7] discretization transforms for machine learning. https://machinelearningmastery.com/discretization-transforms-for-machine-learning/.

[8] f1 score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.f1_score.html/.

[9] Handling imbalanced dataset in supervised learning using family of smote algorithm. https://www.datasciencecentral.com/profiles/blogs/handling-imbalanced-data-sets-in-supervised-learning-using-family/.

[10] Here's all you need to know about encoding categorical data (with python code). https://www.analyticsvidhya.com/blog/2020/08/types-of-categorical-data-encoding/.

[11] How to calculate feature importance with python. https://machinelearningmastery.com/calculate-feature-importance-with-python/.

[12] Hyperparameter optimization. https://en.wikipedia.org/wiki/Hyperparameter_optimization.

[13] Kurtosis. https://en.wikipedia.org/wiki/Kurtosis.

[14] Lable encoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.LabelEncoder.html#sklearn.preprocessing.LabelEncoder.

[15] Logistic regression. https://christophm.github.io/interpretable-ml-book/logistic.html.

[16] Onehotencoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html#sklearn.preprocessing.OneHotEncoder.

[17] Ordinalencoder. https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html#sklearn.preprocessing.OrdinalEncoder.

[18] Principal components analysis. http://web.stanford.edu/class/stats202/notes/Unsupervised/PCA.html#principal-components-analysis.

[19] Random forest. https://en.wikipedia.org/wiki/Random_forest.

[20] Resample the training set. https://www.kdnuggets.com/2017/06/7-techniques-handle-imbalanced-data.html/.

[21] Skewness. https://en.wikipedia.org/wiki/Skewness.

[22] Square root transformation. http://fmwww.bc.edu/repec/bocode/t/transint.html#:~:text=exploited%20if%20useful.-,Square%20root,be%20applied%20to%20zero%20values.

[23] Square root transformation: A beginner's guide. https://quantifyinghealth.com/square-root-transformation/.

[24] standardscaler and minmaxscaler transforms in python. https://machinelearningmastery.com/standardscaler-and-minmaxscaler-transforms-in-python/.

[25] Support vector machines. http://scikit-learn.org/stable/modules/svm.html#:~:text=Support%20vector%20machines%20(SVMs)%20are,than%20the%20number%20of%20samples.

[26] Svm (support vector machine) — theory. https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72.

[27] Synthetic minority oversampling technique. https://machinelearningmastery.com/smote-oversampling-for-imbalanced-classification/.

[28] Tree. https://scikit-learn.org/stable/modules/tree.html#tree.

[29] Understanding random forests classifiers in python. https://www.datacamp.com/community/tutorials/random-forests-classifier-python.

[30] Understanding the mathematics behind support vector machines. https://heartbeat.fritz.ai/understanding-the-mathematics-behind-support-vector-machines-5e20243d64d5.

[31] What are outliers in the data? https://www.itl.nist.gov/div898/handbook/prc/section1/prc16.htm.

[32] Why is skewness important. https://www.analyticsvidhya.com/blog/2020/07/what-is-skewness-statistics/.

[33] Peter Bickel and Kjell Doksum. An analysis of transformation revisited. *Journal of the American Statistical Association*, 76, 06 1981.

[34] Feng Changyong. Log-transformation and its implications for data analysis. *Shanghai Arch Psychiatry*, 26:105–9, 2014.

[35] Georgios Douzas, Fernando Bacao, and Felix Last. Improving imbalanced learning through a heuristic oversampling method based on k-means and smote. *Information Sciences*, 465:1–20, Oct 2018.

[36] Sérgio Moro, Paulo Cortez, and Paulo Rita. A data-driven approach to predict the success of bank telemarketing. *Decision Support Systems*, 62:22–31, 2014.