PGPDSE FT Capstone Project – Final Report

Project Group Info:

BATCH DETAILS	DSE offline Bangalore June 2022
TEAM MEMBERS	• Aadhith Raj Sunil • Alfred C Elver • Ankit Patel • Masud Karim Laskar • Sanjay Rajesh.
DOMAIN OF PROJECT	Finance
PROJECT TITLE	Comparison of ML Classification Models to Predict Bank Marketing Outcomes
GROUP NUMBER	GROUP – 1
TEAM LEADER	Alfred C Elver
MENTOR NAME	Mrs. Vidhya Kannaiah

Abstract:

A Portuguese Bank has been doing marketing calls to randomly selected customers narrating details of a new term deposit plan. The response of the customers is a mixed bag with more inclination towards the negative side. The bank has data of previous campaigns which comprises of customer information, campaign details and other economic indicators. Using these details, we are building a predictive model to determine the kind of customers who has high chance of subscribing to the term deposit. The classification models assumed to use are logistic regression, Random Forest, XGBoost etc. This Final project report comprises of data visualization and Feature engineering and model building.

Objectives:

The objective is to device a model to predict whether a customer will respond positively to a telemarketing call.

Marketing selling campaigns constitute a typical strategy to enhance business. Companies use direct marketing when targeting segments of customers by contacting them to meet a specific goal. Centralizing customer remote interactions in contact centre eases operational management of campaigns. Such centres allow communicating with customers through various channels, telephone (fixed-line or mobile) being one of the most widely used. Marketing operationalized through a contact center is called telemarketing due to the remoteness characteristic

The Bank is now facing challenges in handling the time and cost incurred for such marketing campaigns as the success rate of such campaigns is very low. The organization has to come up with an effective solution to target customers who are more likely to take up the offer from the bank thereby saving organizational resources.

Industry Review:

The world of marketing stretches across thousands of different industries and products. From your local hairdressers to global companies, all of them have to market their services in some shape or form.

Banks are no different, and although we might think of them as a staple part of our high streets, the rise of online banking means there’s a battle for screen time as the retail giants go head-to-head with challenger banks.

Recent studies suggest that as of June 2020, companies and organisations within the banking and financial sector are spending an average of 13% of their overall budget on marketing.

This continues to rise from closer to 6% in 2017, to then around 12% in 2019. Which gives us clear evidence that the banks are putting more emphasis on digital marketing to gain new customers.

These small percentages when seen at first sum up to figures in millions when the bank wealth is discussed in billions. So, the need of perfecting the right target for marketing saves us resources and money. That is where AI comes in.

Current solution to the problem

At present the banks are relying on the methodology of allotting senior tele-communication executives on high probability customers based on the level of activity showcased by the customer. The unexperienced executives are given the mundane task of calling and impressing the less active customers hoping they will take the offer.

There is very little statistical or data driven methods to predict the response of the customers.

Proposed solution to the problem

Technology enables rethinking marketing by focusing on maximizing customer lifetime value through the evaluation of available information and customer metrics, thus allowing to build longer and tighter relations in alignment with business demand. Also, it should be stressed that the task of selecting the best set of clients, i.e., that are more likely to subscribe a product, is considered NP-hard in. Decision support systems (DSS) use information technology to support managerial decision making. There are several DSS subfields, such as personal and intelligent DSS. Personal DSS are related with small-scale systems that support a decision task of one manager, while intelligent DSS use artificial intelligence techniques to support decisions Another related DSS concept is Business Intelligence (BI), which is an umbrella term that includes information technologies, such as data warehouses and data mining (DM), to support decision making using business data. DM can play a key role in personal and intelligent DSS, allowing the semi-automatic extraction of explanatory and predictive knowledge from raw data. In particular, classification is the most common DM task and the goal is to build a data-driven model that learns an unknown underlying function that maps several input variables, which characterize an item (e.g., bank client), with one labelled output target (e.g., type of bank deposit sell: \failure" or \success").

Dataset and Domain:

The data is related to direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. 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. There are all 41188 rows and 20 inputs, ordered by date (from May 2008 to November 2010

Input variables:

# **bank client data:**

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')

# **Related with the last contact of the current campaign:**

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.

# Other attributes:

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')

# Social and economic context attributes

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):

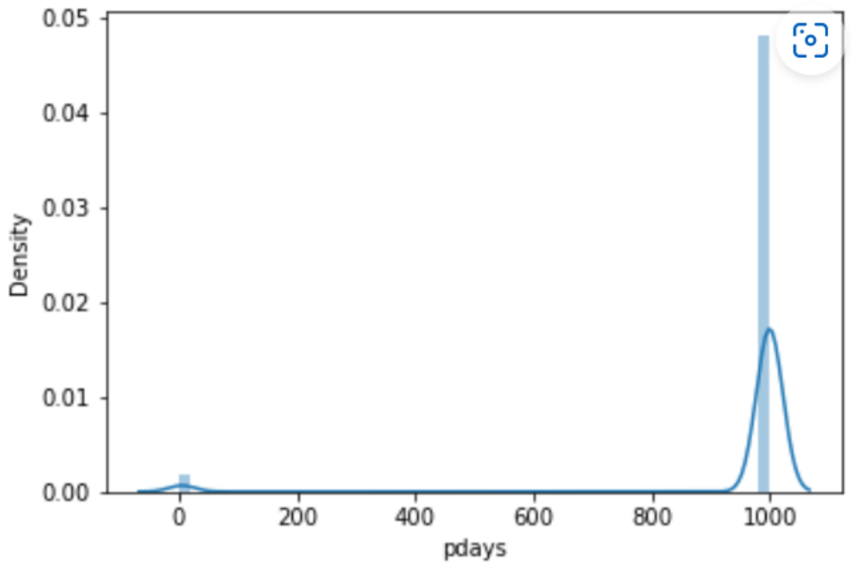
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

<u>Uni-Variate and Bi-Variate Analysis</u>

There are 10 categorical features and 10 numerical features we are checking the distribution of each variable and its influence on the target variable.

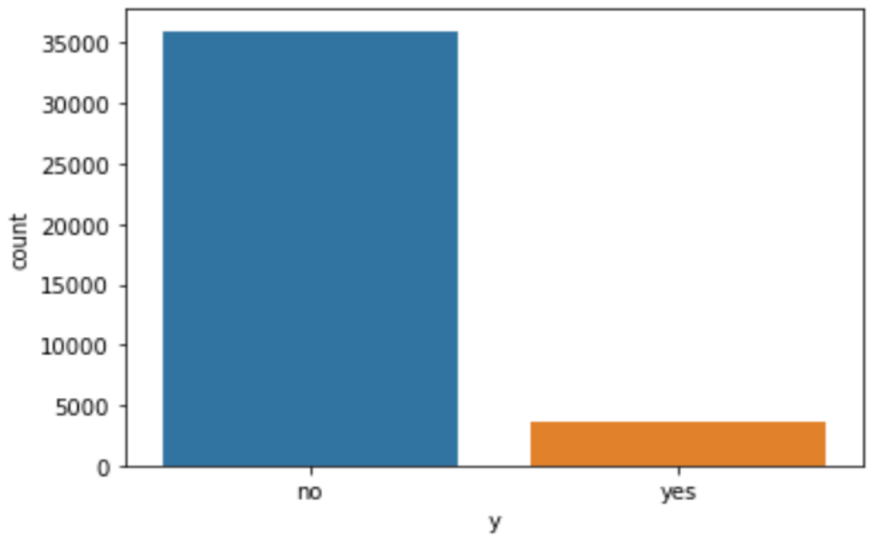
<u>Numerical</u>

<u>Pdays</u>



```
sns.countplot(x=df[df["pdays"]==999]["y"])
```

```
<AxesSubplot:xlabel='y', ylabel='count'>
```



```python
df[df["pdays"]==999]["y"].value_counts(normalize=True)
```

```
no     0.907418
yes    0.092582
Name: y, dtype: float64
```

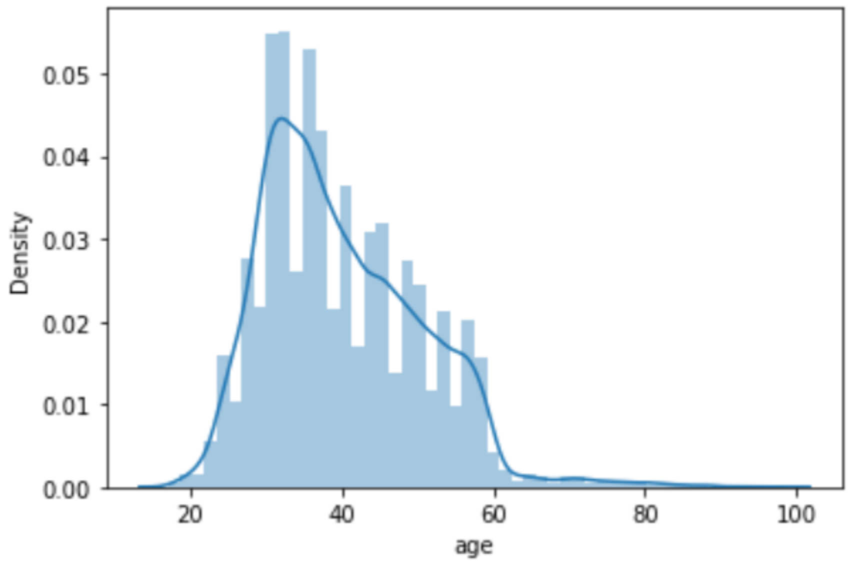
```python
df[df["pdays"]!=999]["y"].value_counts(normalize=True)
```

```
yes    0.638284
no     0.361716
Name: y, dtype: float64
```

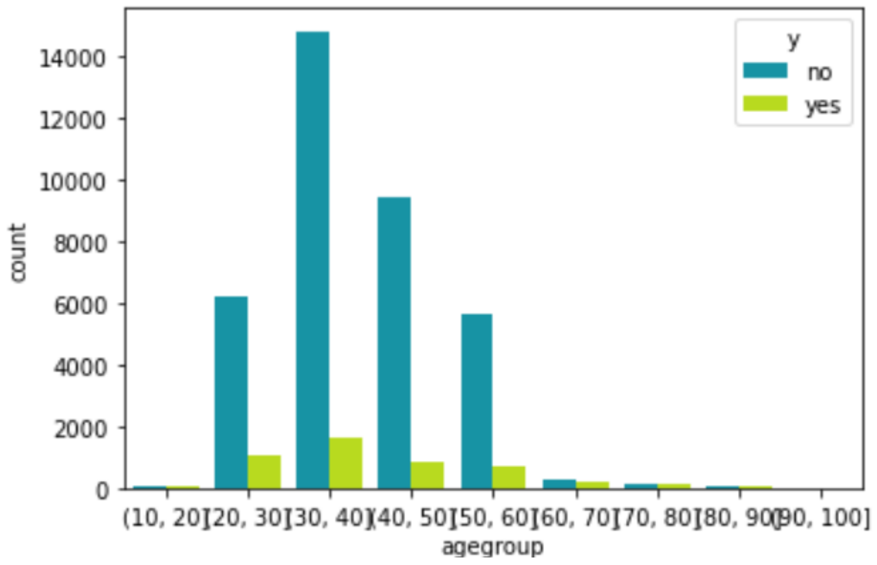
The feature pdays has more than 96% of the value as “999” which means that particular customer has not been contacted before. In that 95% yes/no is 10%/90%

The other set of customers who have been contacted before also has yes/no at 64%/36% Which means there is inverse in trend in the small subset of 4% where the yes is having major share and no is having less share.

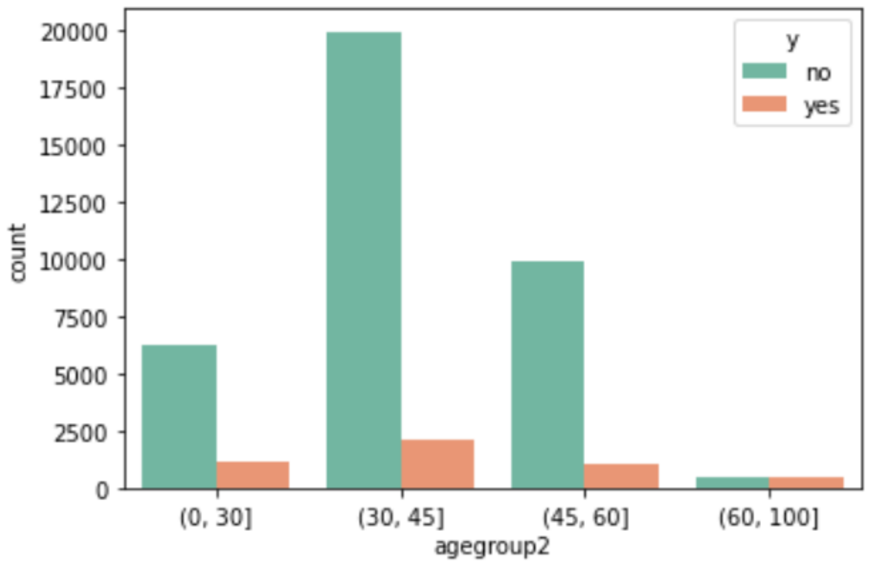
Age



Age is a continuous numerical variable and we are discretizing it into bins. Age is having a standard deviation of 10, max value of 98 and min value of 17 so we will make bins of 10 size starting 10 up to 100.



we can see a high segment are on 30 -60 and after 60 and below 30 there are not much people so for more clarity, we can re-compartmentalize with less than 30 ,30-60,60+. Since 60+ are retired people/senior citizen the division makes some sense also



it seems that first and last groups have more percent of conversion into deposit

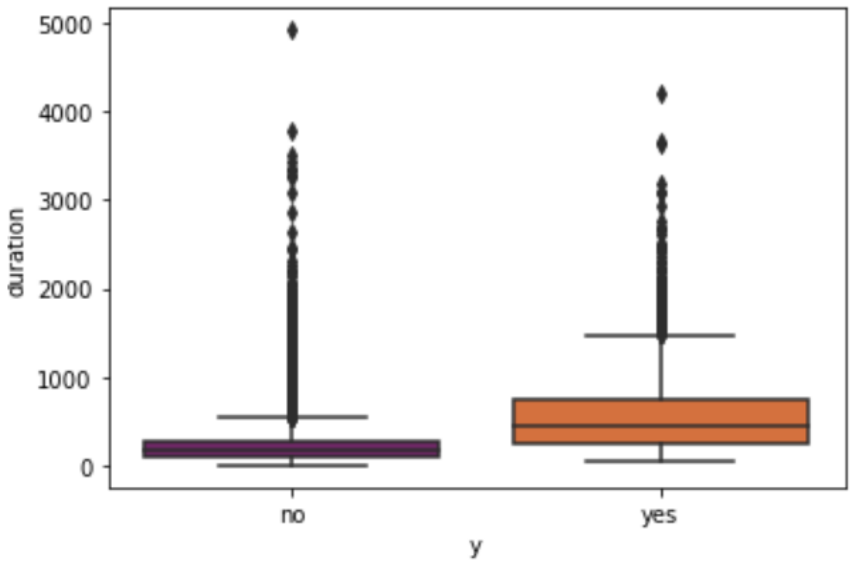
We will find that out numerically,

- For the age group from 0 to 30 there is 15% success
- For the age group from 60 onwards there is 45% success
- For those who are between 30 to 60 only 9% success is there

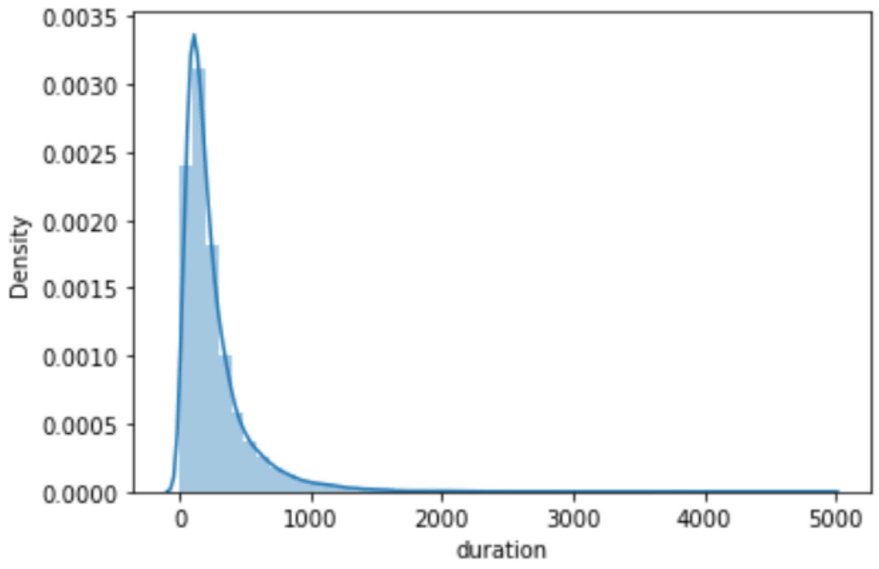
This means that the old and the young has more tendency towards savings than the working class.

Duration

The duration of the call is said to be a great predictor on the target variable as said by the dataset itself. We will try to see if that's true.



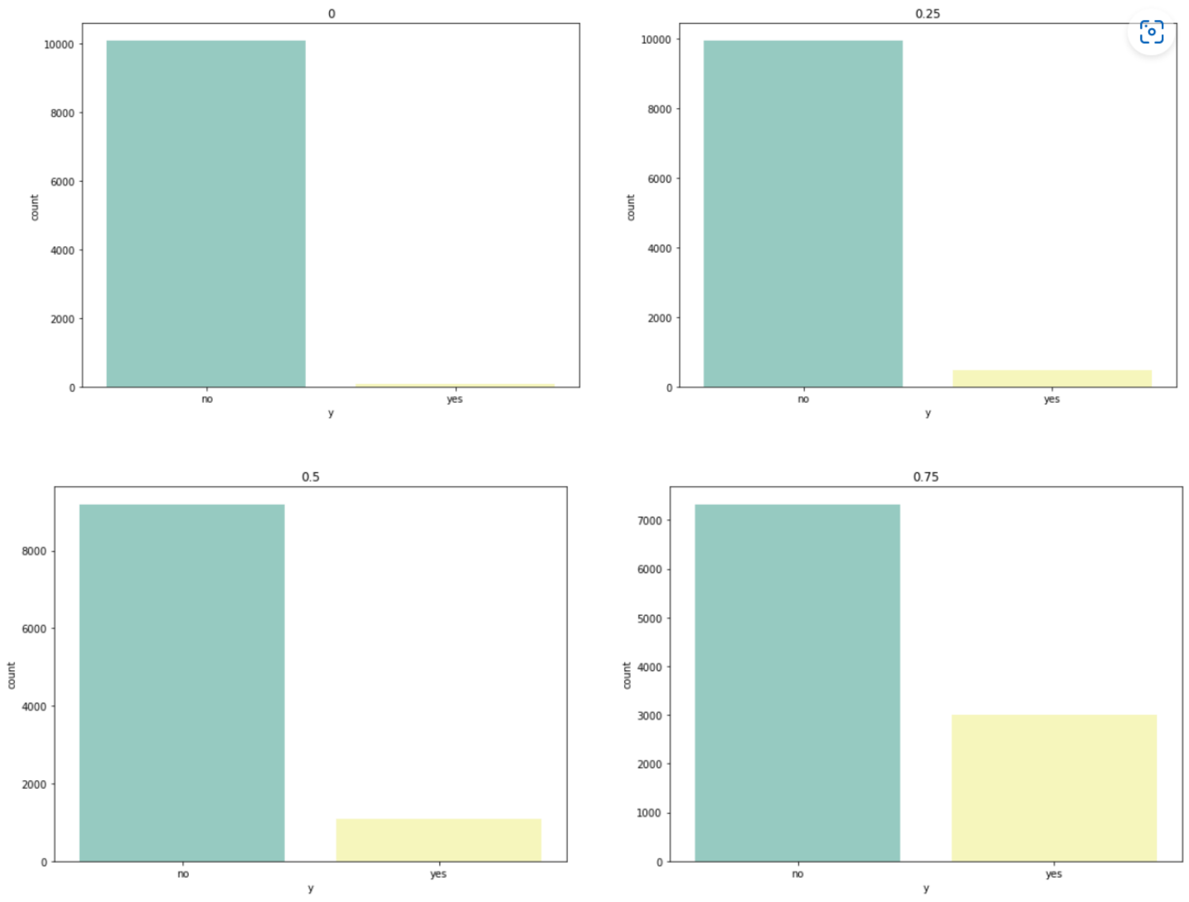
From the box plot we can see there is decent separation of classes we can numerically find these using value counts before that we need to check the distribution of values of duration feature.



From these we can understand that duration is a continuous numerical feature and distributed with right skew.

The values are given in seconds. But the maximum value from descriptive statistics provided by the describe function is being 4918 seconds i.e., 82 minutes it might be a mistake but since we are not going to use this feature there is no need for any rectification.

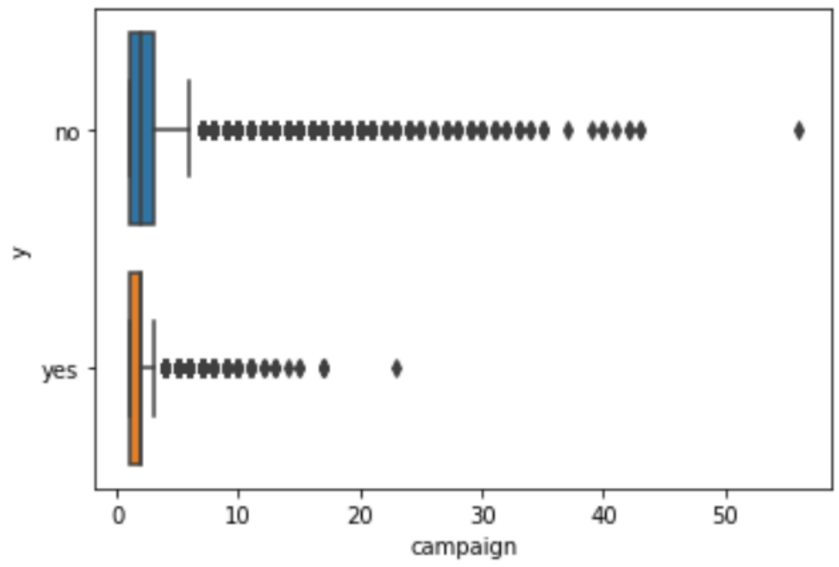
we will check the target variable distribution in each quartile



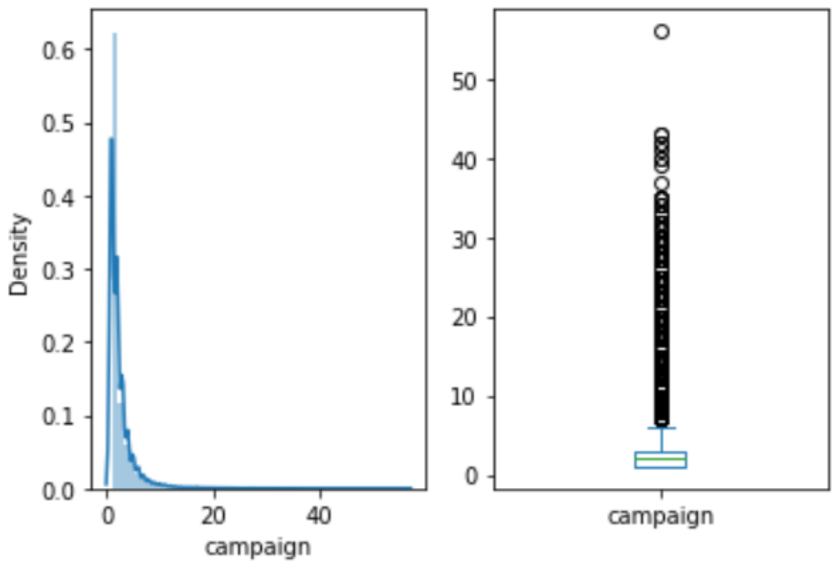
From the count plot we can see that as the duration of call increases the percentage of success is increasing

Campaign

This feature tells the number of contacts performed during this campaign and for this client.

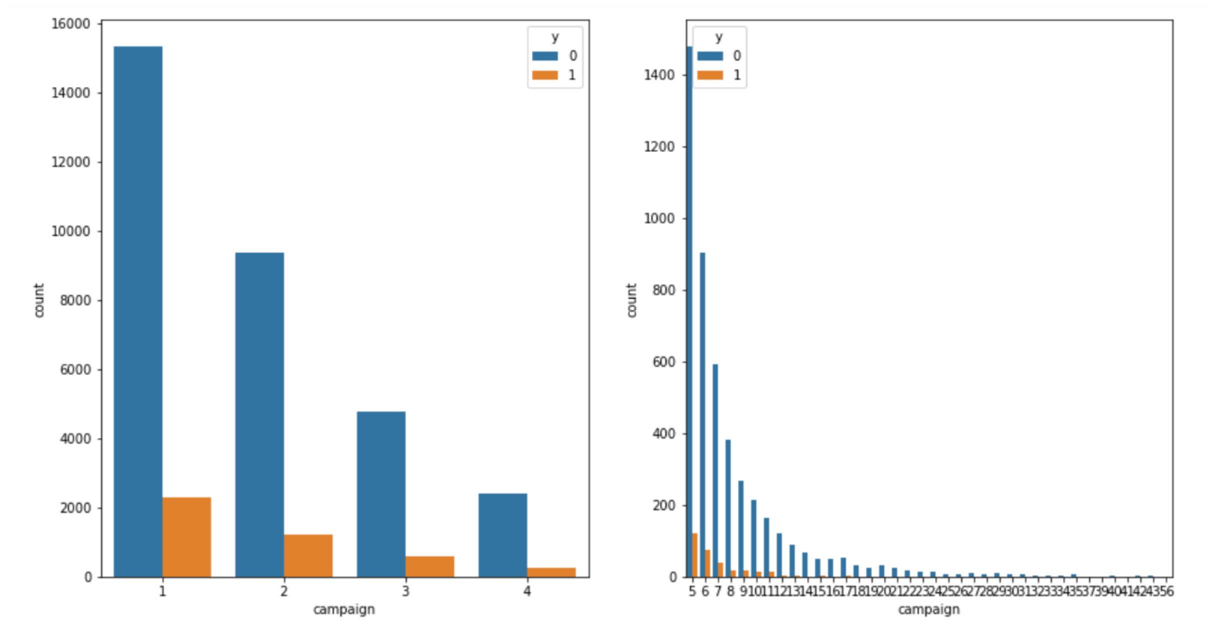


There is overlapping in this feature so it can’t outright predict the outcome

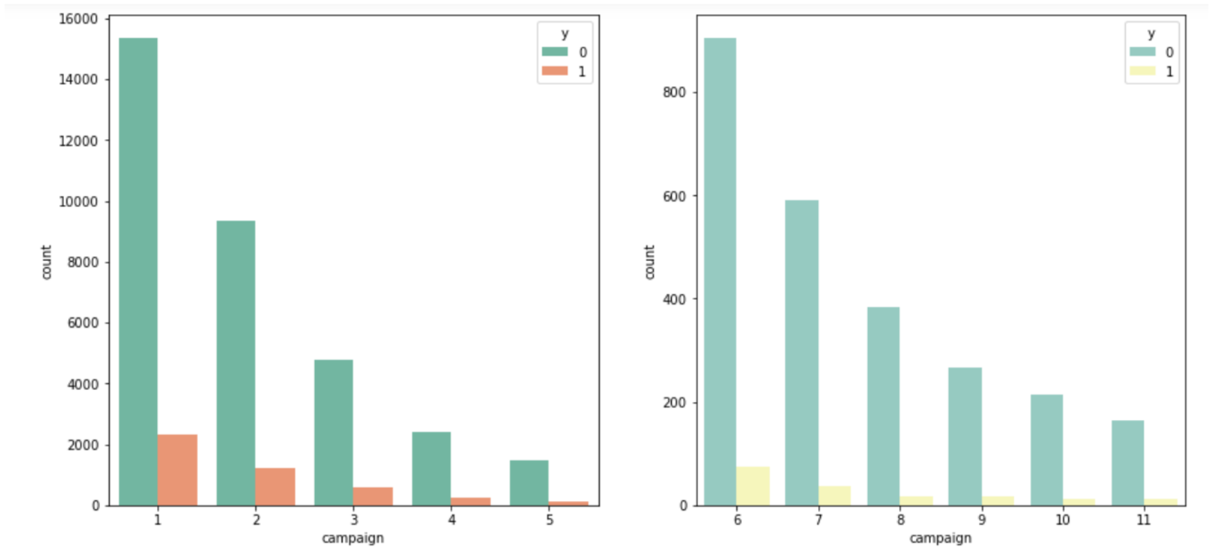


We can that see 75% of the people have been contacted less than or equal to 3 times.

We can also see the maximum number is coming to be 56 which can be a possible outlier if there were enough data for people who have been contacted more than 5 or 10 times, we could have made a subset for those people and done a separate model for them.



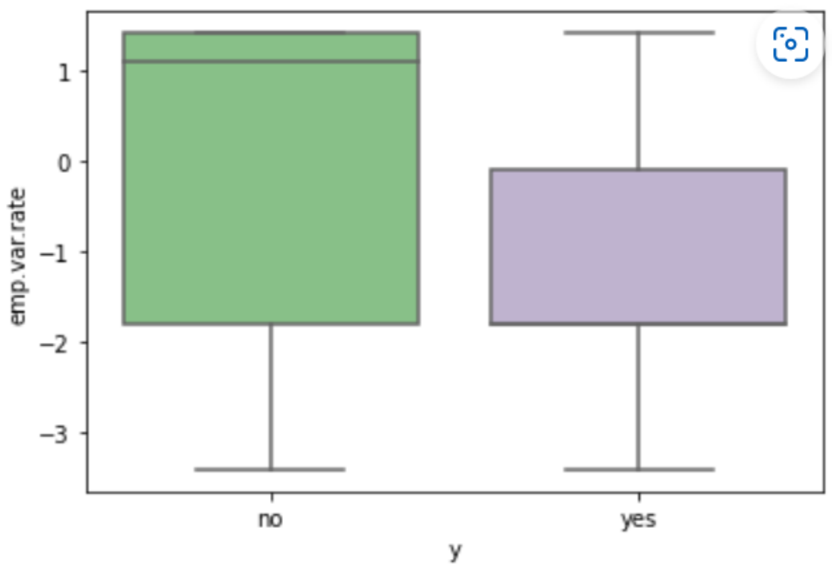
we can see that after 12 the values are not worth considering at all. We can change the graph plots to a meaningful co-ordinate.



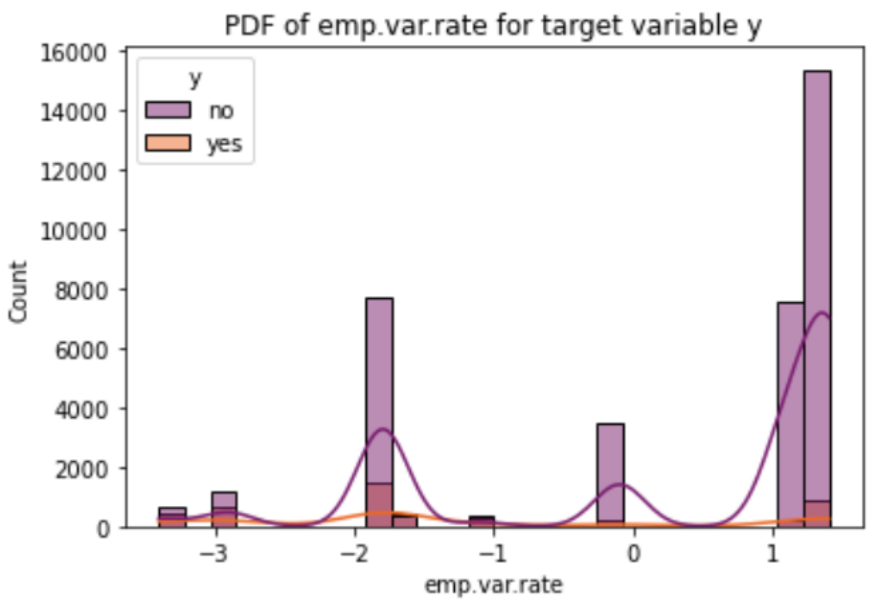
From this we can infer that as the no of instances where a contact has been established and the outcome doesn’t have much relationship. The ratio of yes and no in each x value is almost the same.

<u>Emp.var.rate</u>

Employment variation rate aka Cyclical employment variation is essentially the variation of how many people are being hired or fired due to the shifts in the conditions of the economy. When the economy is in a recession or depression, people should be more conservative with their money and how the spend it because their financial future is less clear due to cyclical unemployment. When the economy is at its peak, individuals can be more open to risky investments because their employment options are greater.



The boxplot reveals there is considerable separation between the two classes of target using the emp var rate making it a good predictor. Obviously so because if there is instability in the employment status people will be hesitant to invest

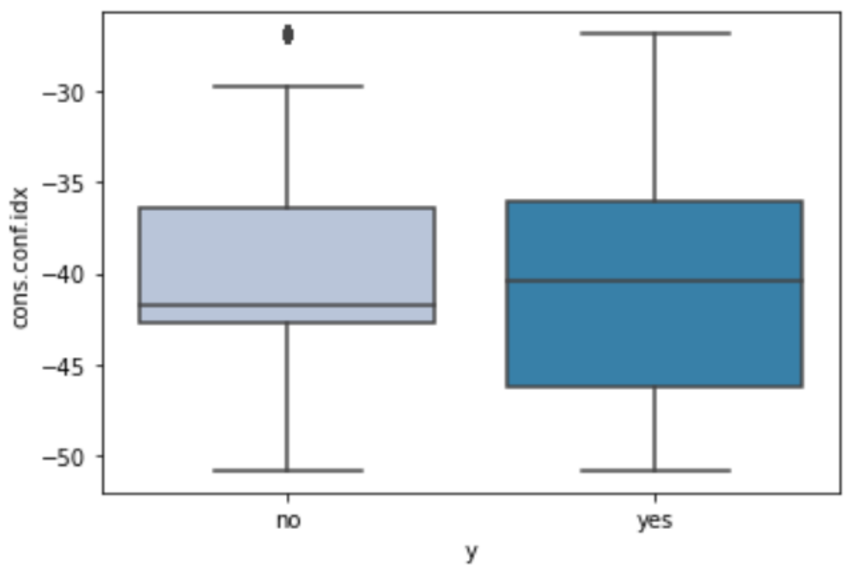


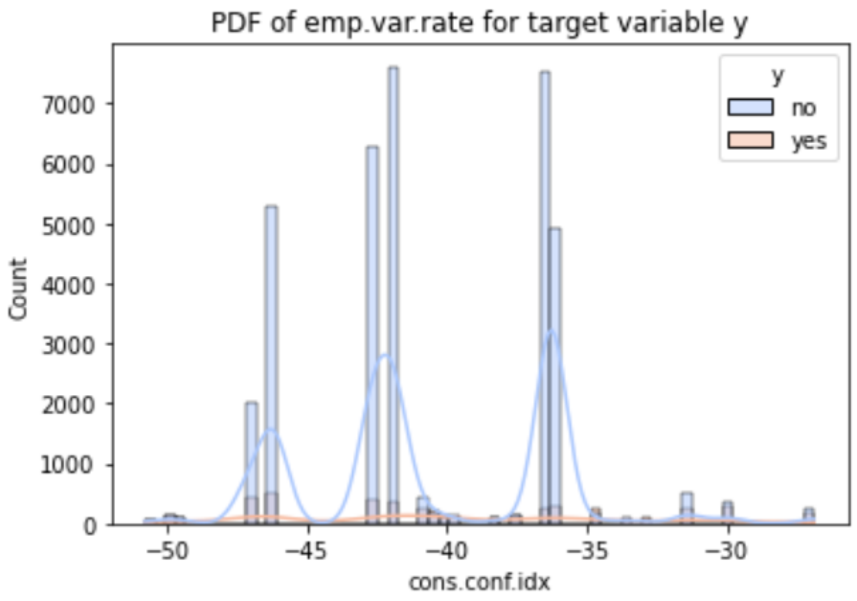
We can see the peaks for NO are increasing when the emp var rate is increasing. We can see the difference increasing between no and yes while we travel from -3.4 to 1.4,After -1.1 the no/yes ratio comes in the order of 90:10.

cons.conf.idx

A consumer confidence index (CCI) is an economic indicator published by various organizations in several countries.

In simple terms, increased consumer confidence indicates economic growth in which consumers are spending money, indicating higher consumption. Decreasing consumer confidence implies slowing economic growth, and so consumers are likely to decrease their spending. The idea is that the more confident people feel about the economy and their jobs and incomes, the more likely they are to make purchases. Declining consumer confidence is a sign of slowing economic growth and may indicate that the economy is headed into trouble.





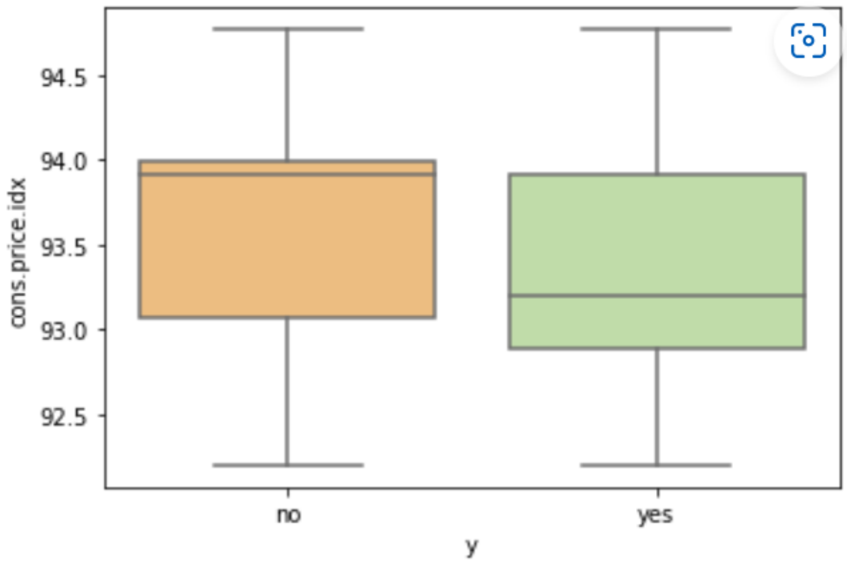
The analysis shows not much pattern or variation in target using teh consumer confidence index. This is opposite to the theory we have studied and we will look into this later.

<u>cons.price.idx</u>

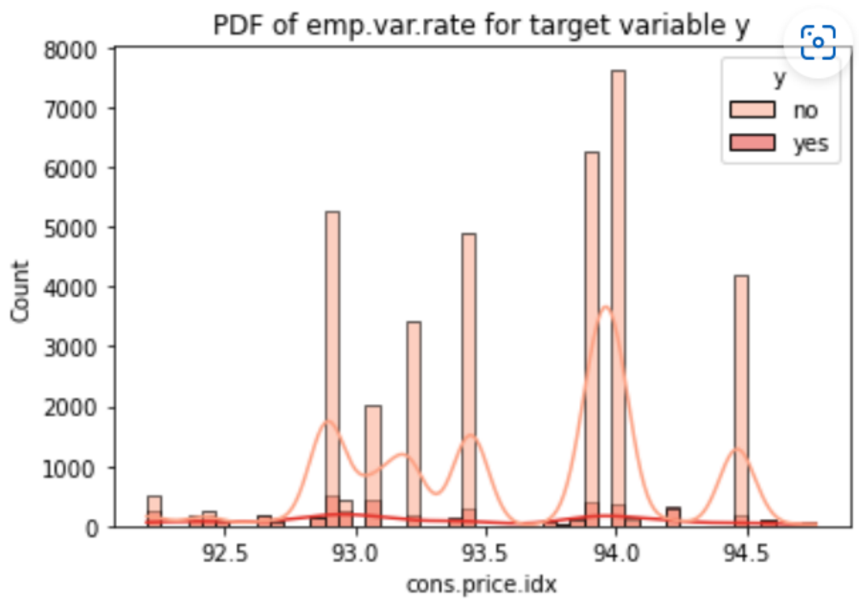
Consumer price index is referred to as that index that is used in calculating the retail inflation in the economy by tracking the changes in prices of most commonly used goods and services.

In other words, the consumer price index calculates the changes in price of a common basket of goods and services. It is also called a market basket and is used for calculating the price variations in fixed items.

CPI can be used to calculate the cost of living of the people of a country and also the changes in the purchasing power of the currency of a nation.



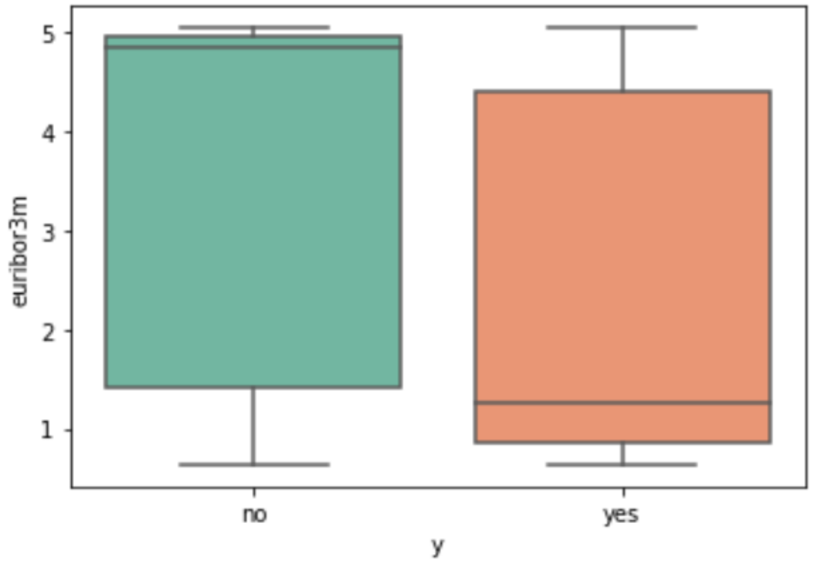
There is a clear separation in the median line of both classes but the values of cons. price index are very close to each other.

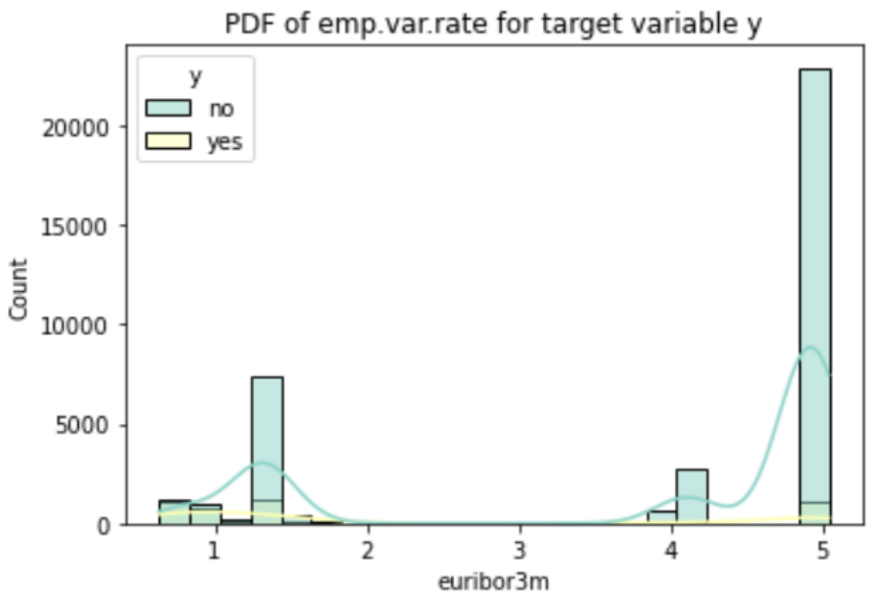


No visible pattern as the values of the consumer price index is in a very small range.

euribor3m

The 3-month Euribor interest rate is the interest rate at which a selection of European banks lends one another funds denominated in euros whereby the loans have a maturity of 3 months.

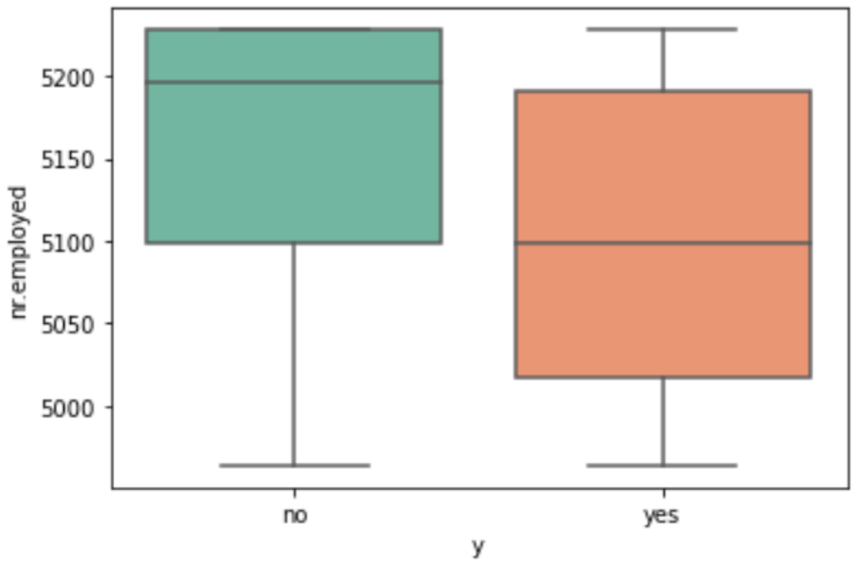


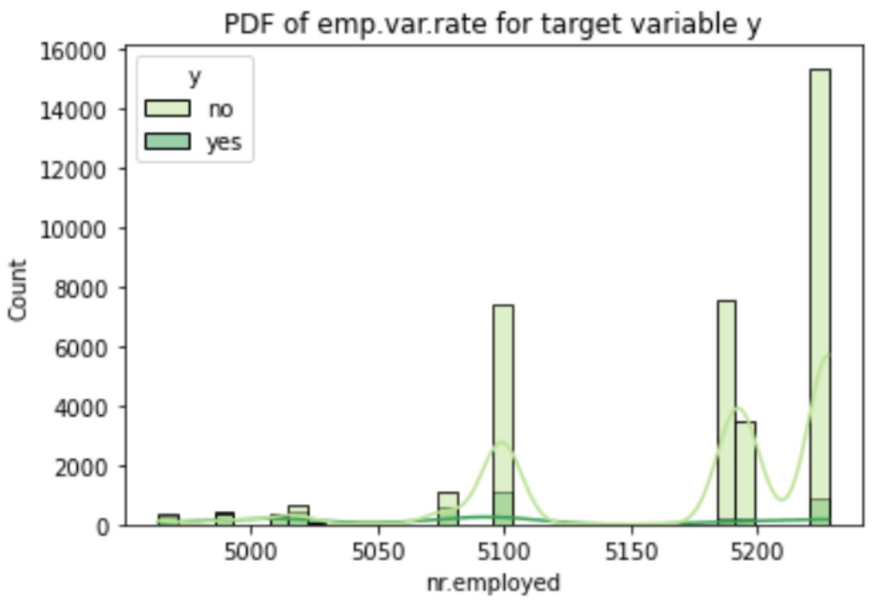


From the above plot, we can clearly see the difference in median for both the classes. This indicates that the feature can be very useful for our case study. But we can validate the assumption only by applying models and extracting feature importance.

<u>nr.employed</u>

number of employees - quarterly indicator

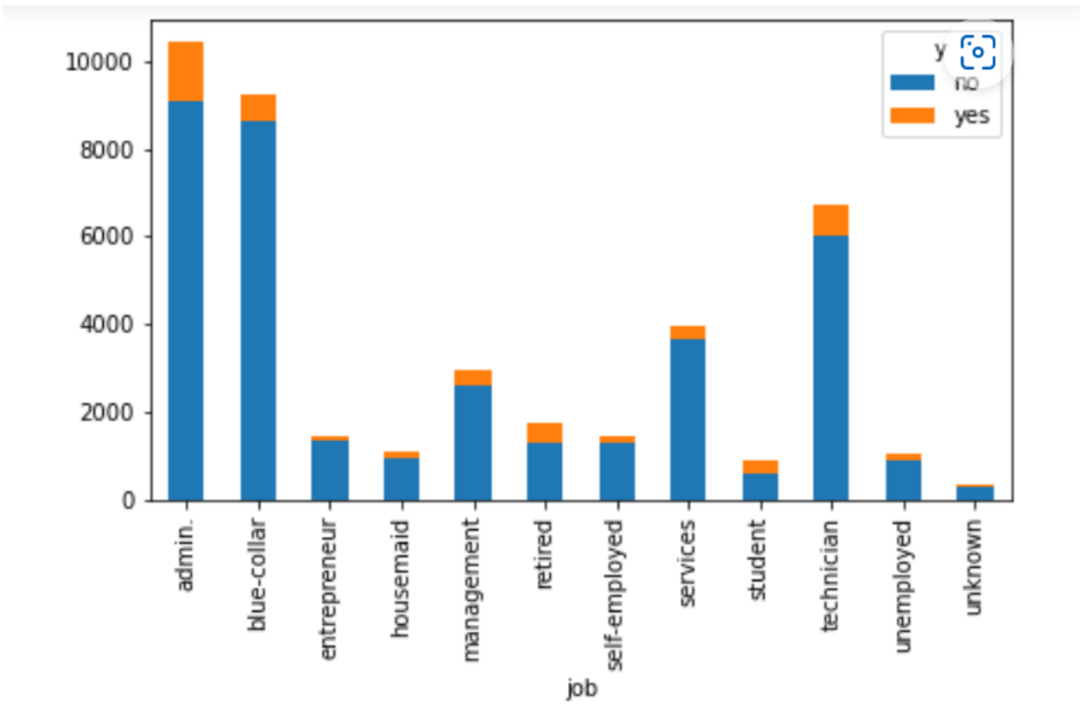




In this feature also, there is noticeable class separation which makes it a good indicator. The number of employees has a positive effect for turning people to subscribe to the term deposit. This can be due to the fact that the more employees the bank has, the more influential and prestigious this bank is.

Categorical Features

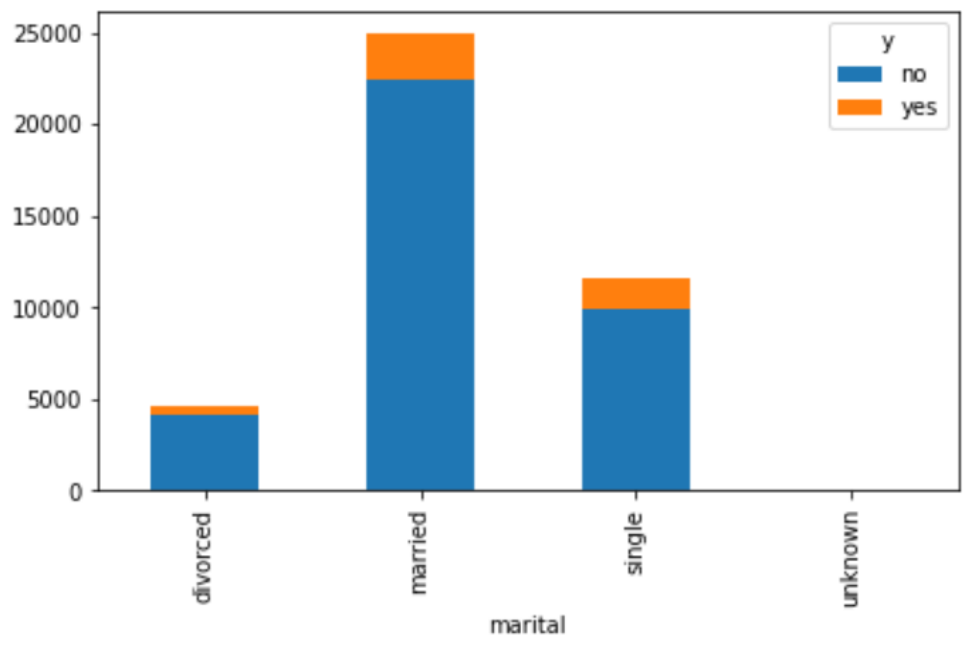
Job



From the above plot, we can see that the customers who have a job of admin have the highest rate of subscribing a term deposit, but they are also the highest when it comes to not subscribing. This is simply because we have more customers working as admin than any other profession.

The “Blue-Collar are the ones who have said NO to FD while the ones who are “Retired” or working at ”Admin” post comparatively said YES.

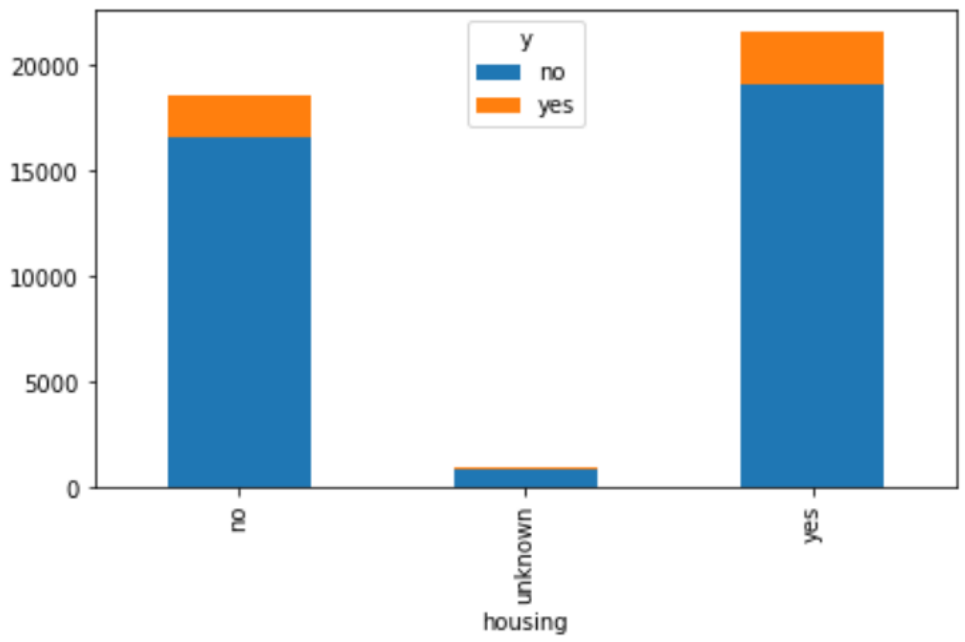
Marital status



Majority of the customers are married. Followed by Single, divorced and unknown.

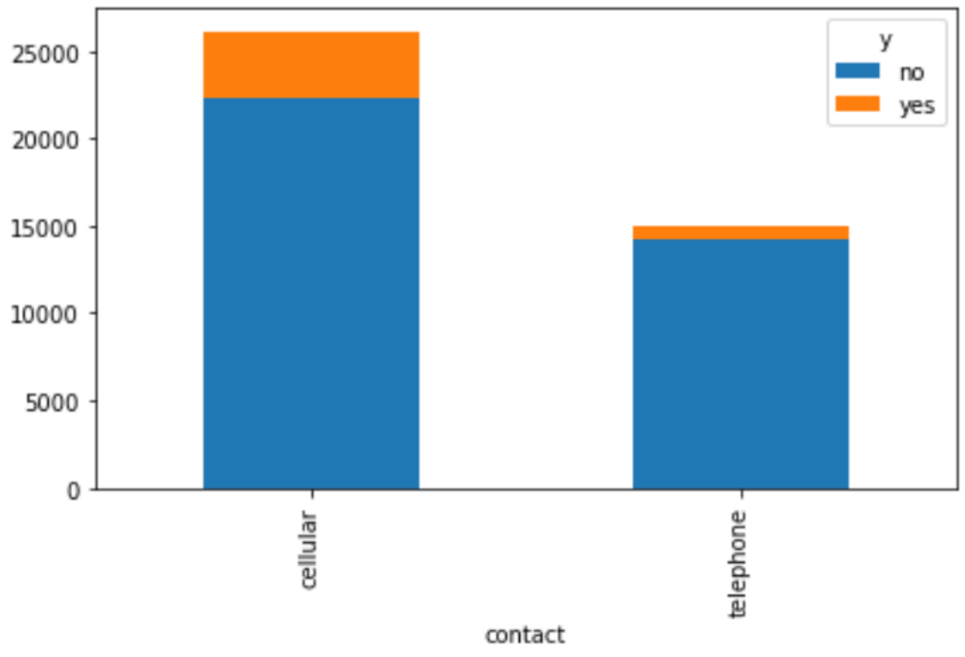
Housing

The customer is having a housing loan or not



Contact

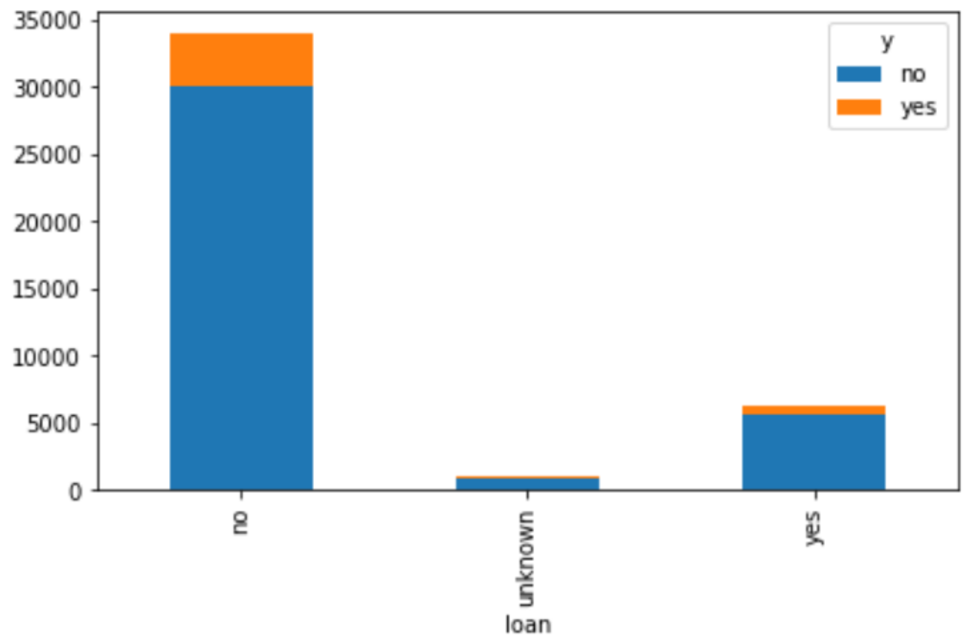
The feature mentions the mode of communication



There is comparatively high chance for yes when the contact is made through cellular. But we can’t justify a logical reason for that, since most of the population is having a cellphone.Another reason maybe the telephone numbers maybe office numbers which doesn’t point to a particular individual.

Loan

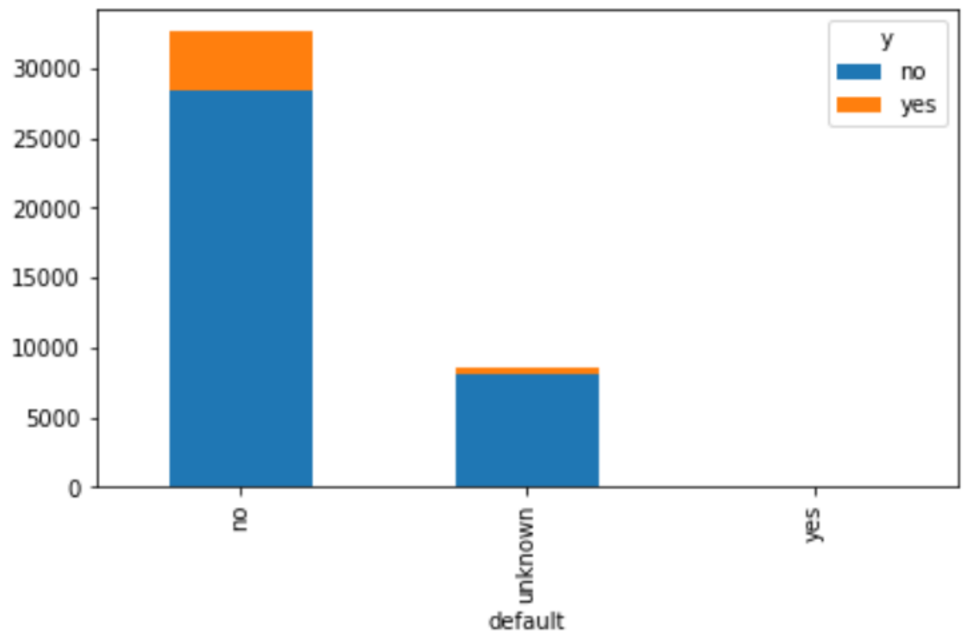
The consumer has personal loan or not



People having personal Loan or Housing Loans didn't Prefer to join for FD's.

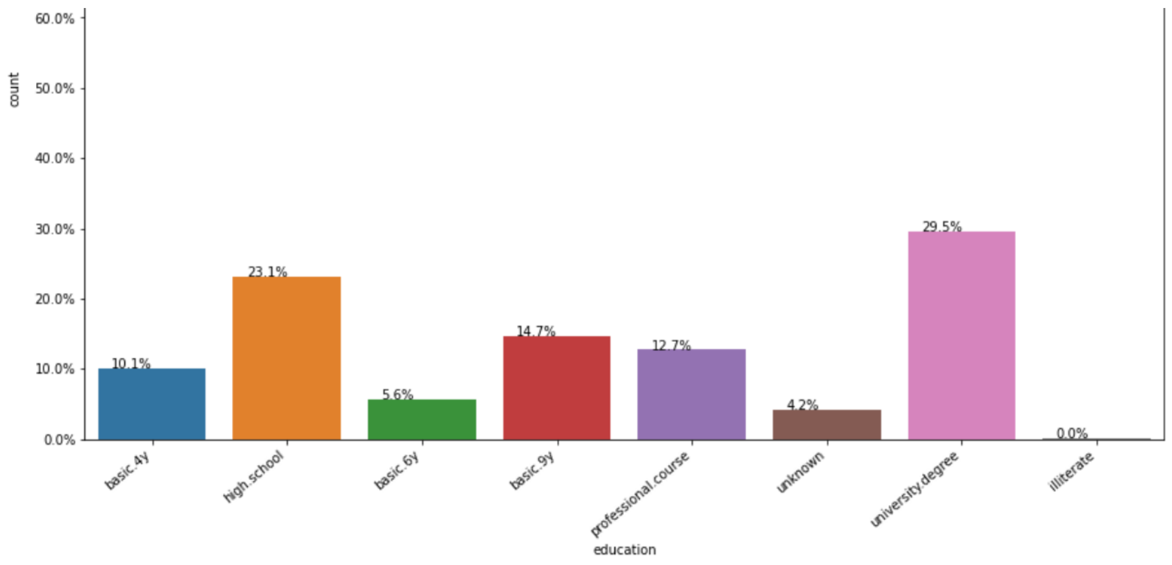
Default

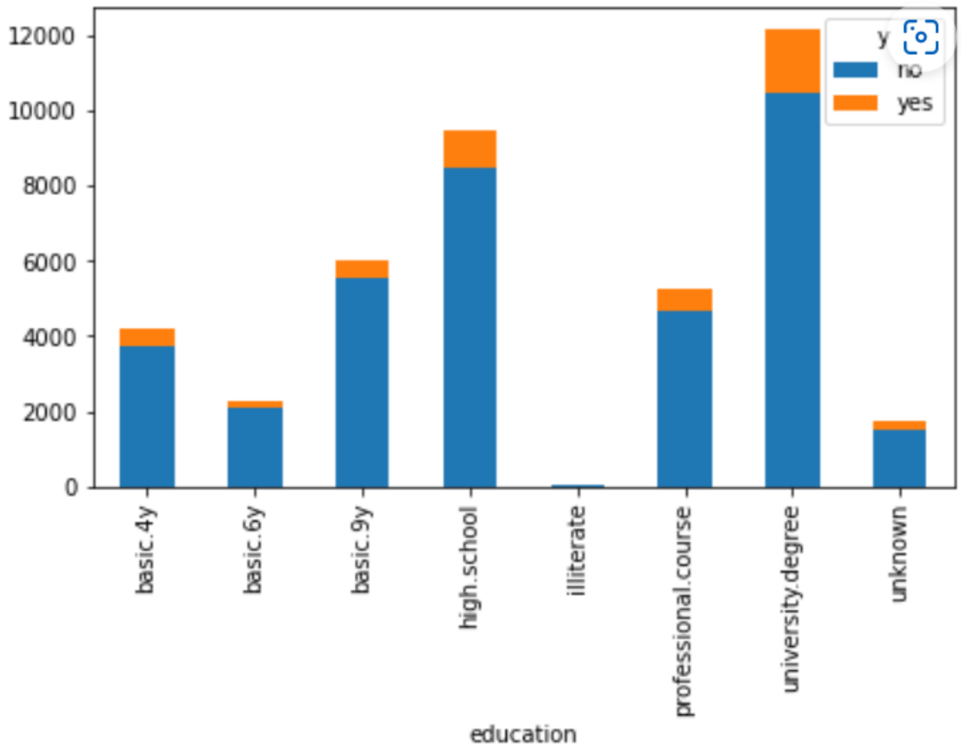
The consumer has defaulted the credit card or not



There is no data in the category of loan defaulters so this feature division is not logical

Education

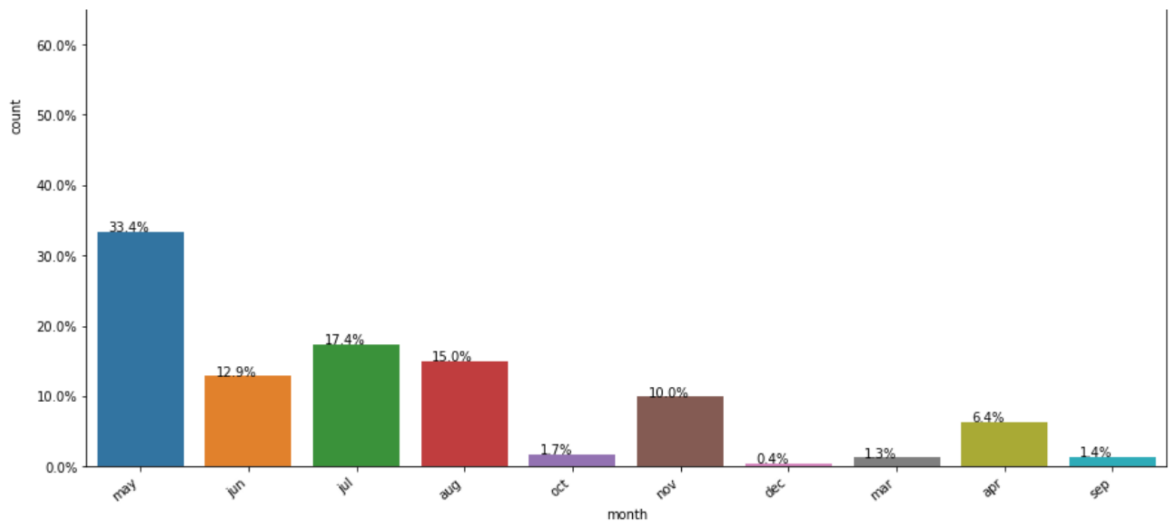


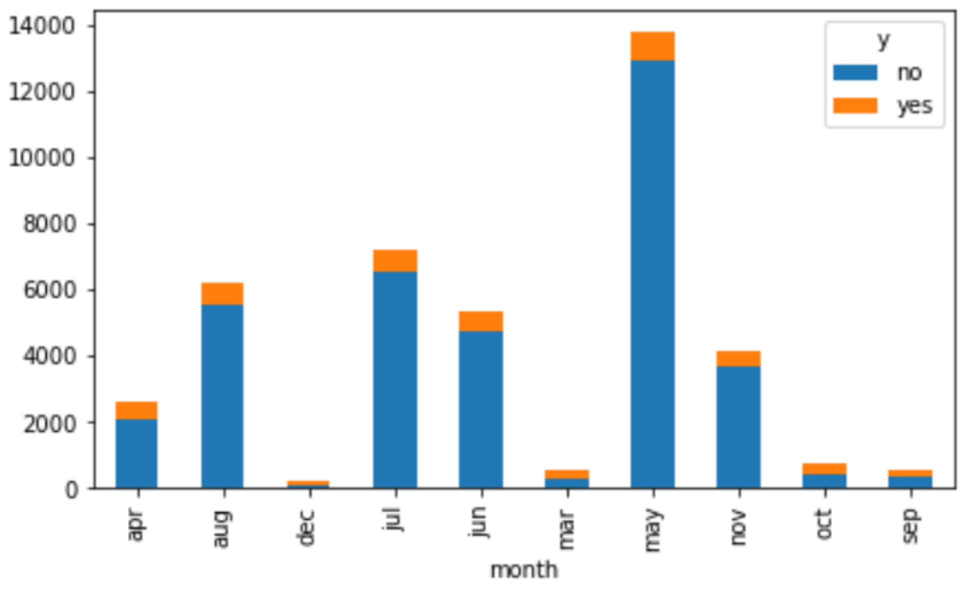


People with university degree have a high tendency of saying yes to a marketing call. We can assume that people who are having university degree has a chance of having good jobs so more money to invest in savings.

Month

The month when the contact has been made

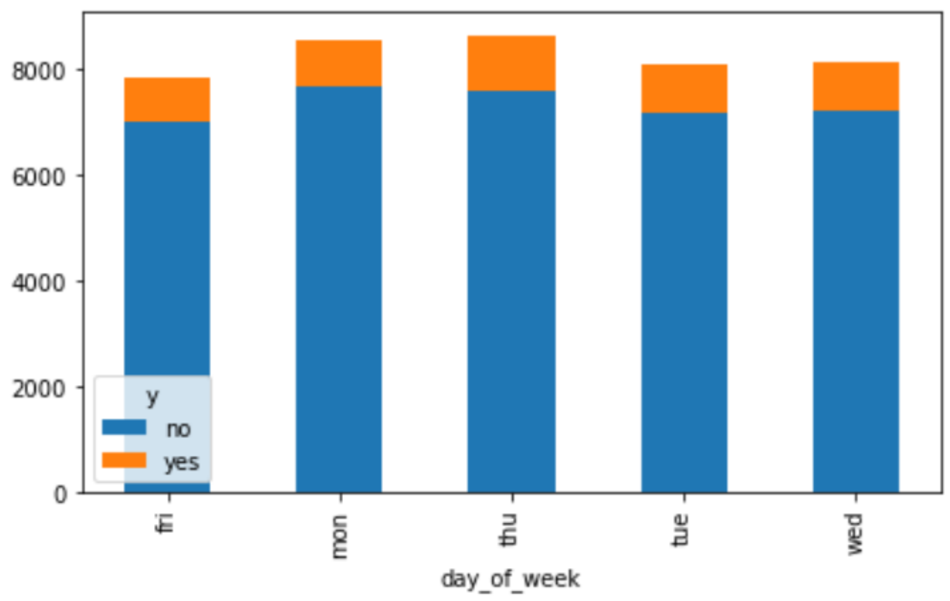




In the month of April, we have seen large numbers getting converted while in the May maximum turning up to say no. the only reason we could figure out was in the month of April they had Financial Year End.

<u>Day of the week</u>

The day of the week when the contact has been made

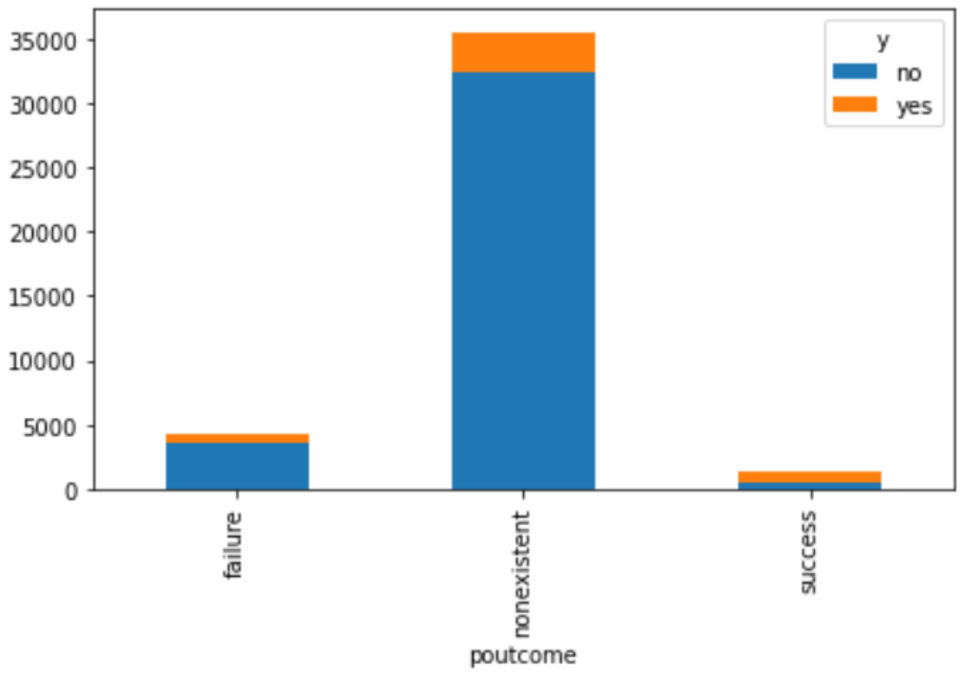


Almost every day has same rate of acceptance

The calls made are also same throughout the week(weekdays)

Poutcome

The outcome of the previous campaigning call that has been made



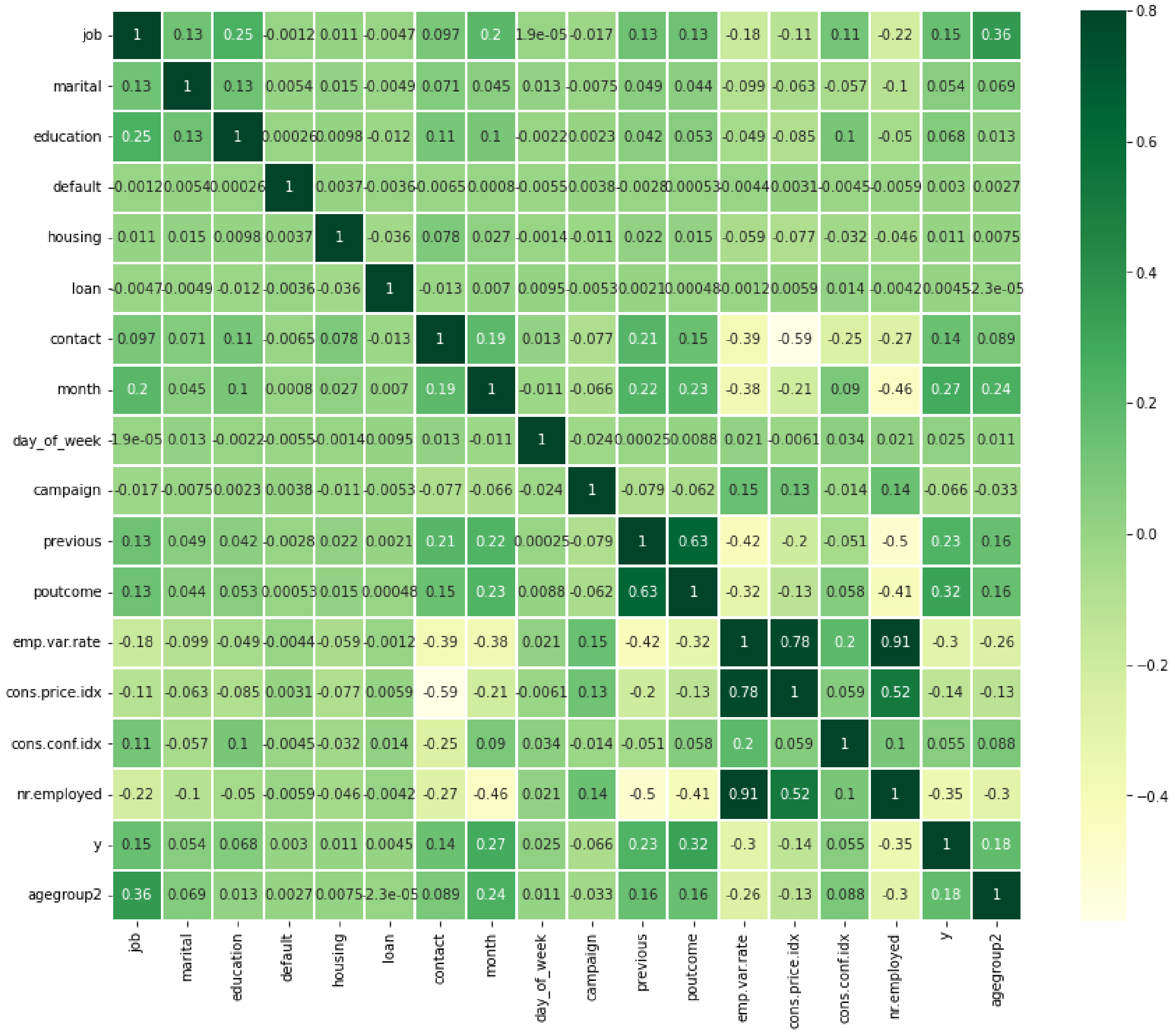
We have received great Response from the people who already had engagements with the bank previously for some other campaigns

But 86% of the datapoints are missing values so any model building with this feature is not practical.

Multi-Variate Analysis

Heatmap

We are going to plot the heatmap and understand the correlation between the features



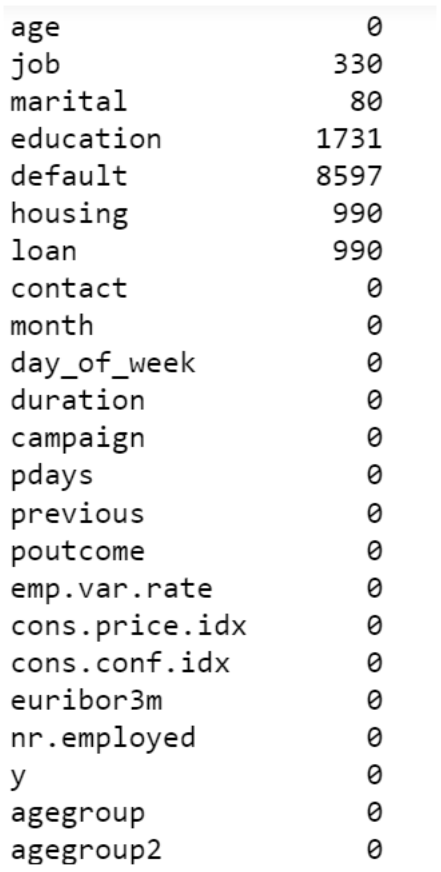
From the heatmap we can see that euribor3m and emp.var.rate are highly correlated(0.97),also euribor3m and nr.employed having an correlation of 0.95 so either of these features can be retained and others omitted.

Feature Engineering

Missing values

If we look at the is null command results, we can see there are no missing values present in the dataset, but if we check the unique values of each column, we can see variables like "unknown" and "999" in some columns which refer to missing values. Therefore, we need to impute meaningful values instead of these terms.

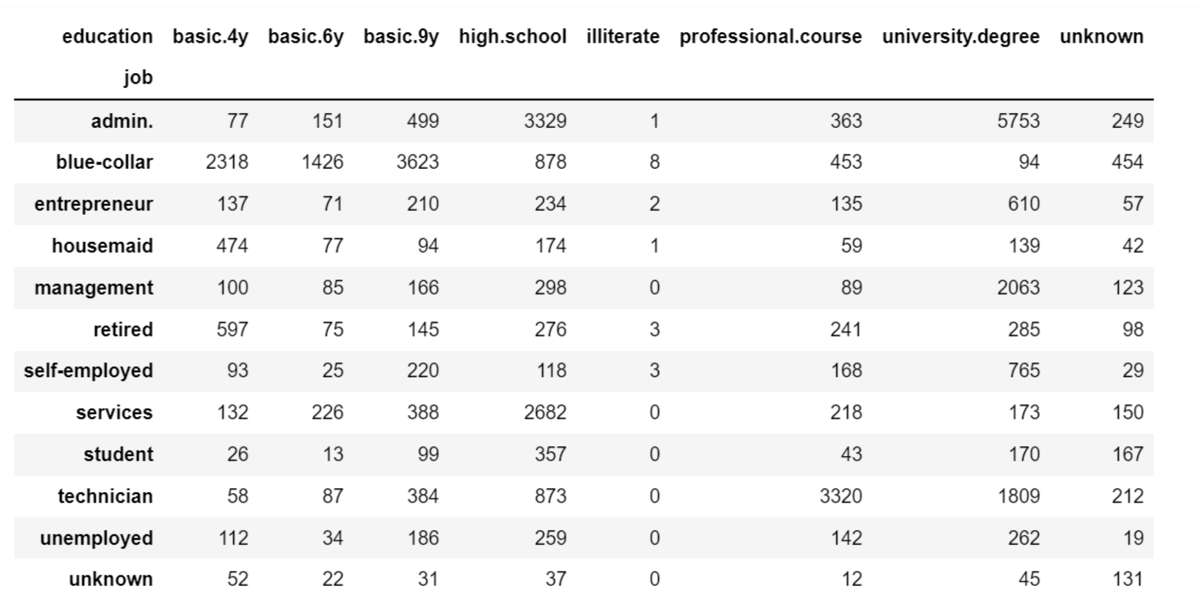
We will replace the unknown with NaN value to understand the number of missing values.



```
age                  0
job                330
marital             80
education         1731
default           8597
housing            990
loan               990
contact              0
month                0
day_of_week          0
duration             0
campaign             0
pdays                0
previous             0
poutcome             0
emp.var.rate         0
cons.price.idx       0
cons.conf.idx        0
euribor3m            0
nr.employed          0
y                    0
agegroup             0
agegroup2            0
```

One way to clean dataset is to delete it, but it gives a bad image of the data and it is unrealistic. Another way is to find some correlation between the data and fill it in

Now, to infer the missing values in ‘job’ and ‘education’, we make use of the cross-tabulation between ‘job’ and ‘education’. Our hypothesis here is that ‘job’ is influenced by the ‘education’ of a person. Hence, we can infer ‘job’ based on the education of the person. Moreover, since we are just filling the missing values, we are not much concerned about the causal inference. We, therefore, can use the job to predict education.



education	basic.4y	basic.6y	basic.9y	high.school	illiterate	professional.course	university.degree	unknown
job								
admin.	77	151	499	3329	1	363	5753	249
blue-collar	2318	1426	3623	878	8	453	94	454
entrepreneur	137	71	210	234	2	135	610	57
housemaid	474	77	94	174	1	59	139	42
management	100	85	166	298	0	89	2063	123
retired	597	75	145	276	3	241	285	98
self-employed	93	25	220	118	3	168	765	29
services	132	226	388	2682	0	218	173	150
student	26	13	99	357	0	43	170	167
technician	58	87	384	873	0	3320	1809	212
unemployed	112	34	186	259	0	142	262	19
unknown	52	22	31	37	0	12	45	131

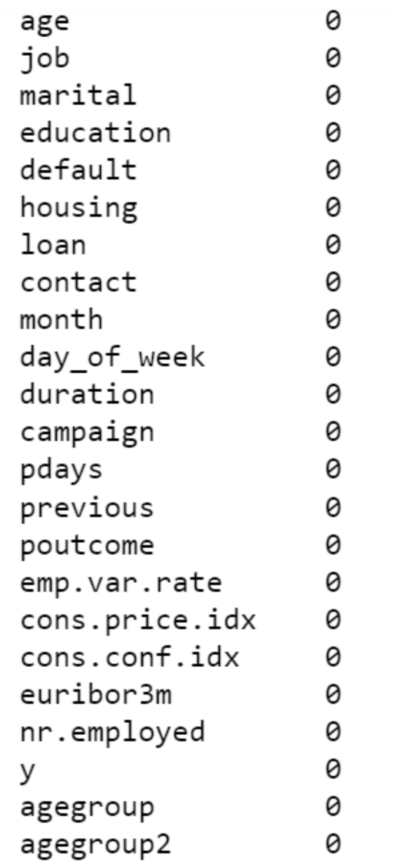
From the cross-tabulation, it can be seen that people with management jobs usually have a university degree. Hence wherever 'job' = management and 'education' = unknown, we can replace 'education' with 'university.degree'. Similarly, 'job' = 'services' → 'education' = 'high.school' and 'job' = 'housemaid' → 'education' = 'basic.4y'.

Similarly, we can do imputation of job from education also

We can also see, if 'age' > 60, then the 'job' is 'retired,' which makes sense.

Rest of the missing values we can fill using the mode of the columns

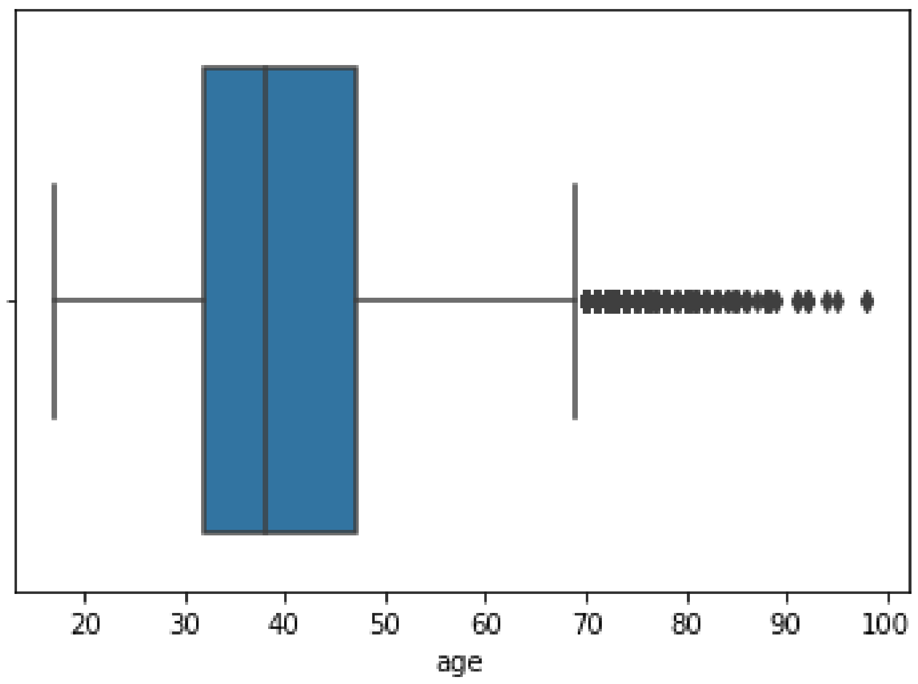
We can check the missing values after all the imputation

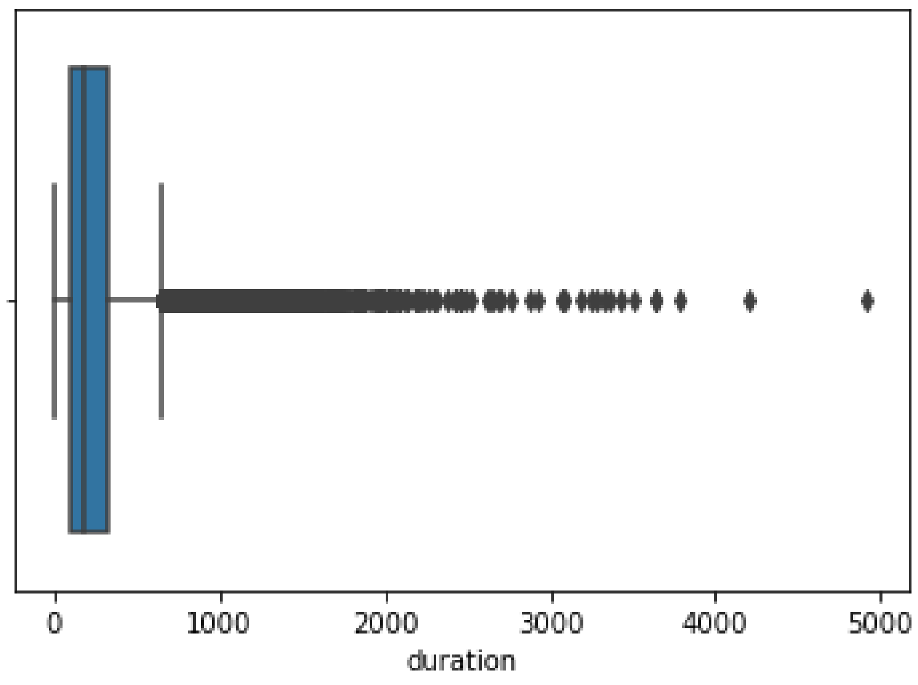

```
age               0
job               0
marital           0
education         0
default           0
housing           0
loan              0
contact           0
month             0
day_of_week       0
duration          0
campaign          0
pdays             0
previous          0
poutcome          0
emp.var.rate      0
cons.price.idx    0
cons.conf.idx     0
euribor3m         0
nr.employed       0
y                 0
agegroup          0
agegroup2         0
```

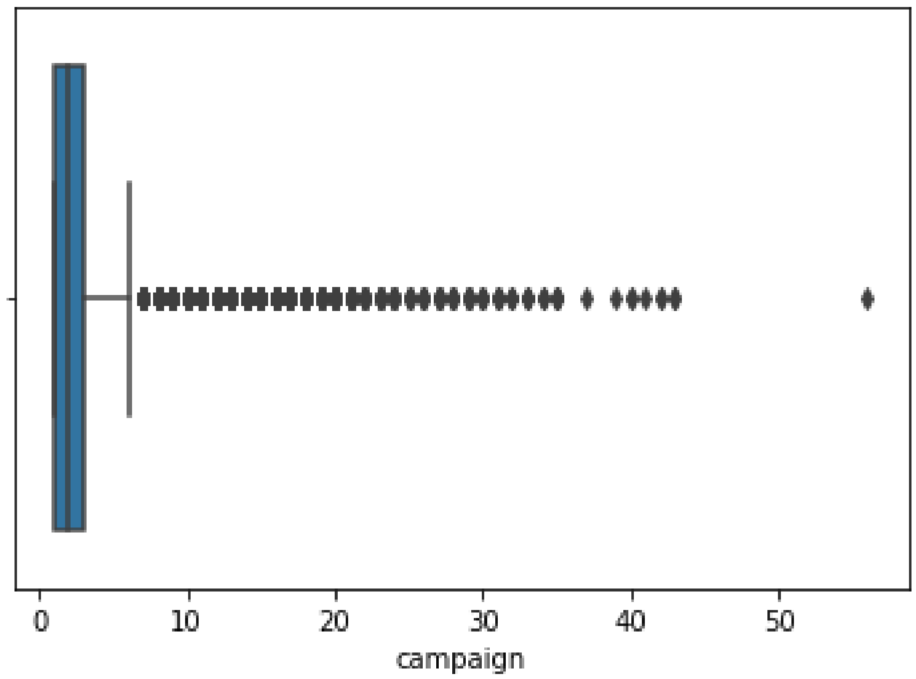
We can now see there are no missing values in the dataset right now

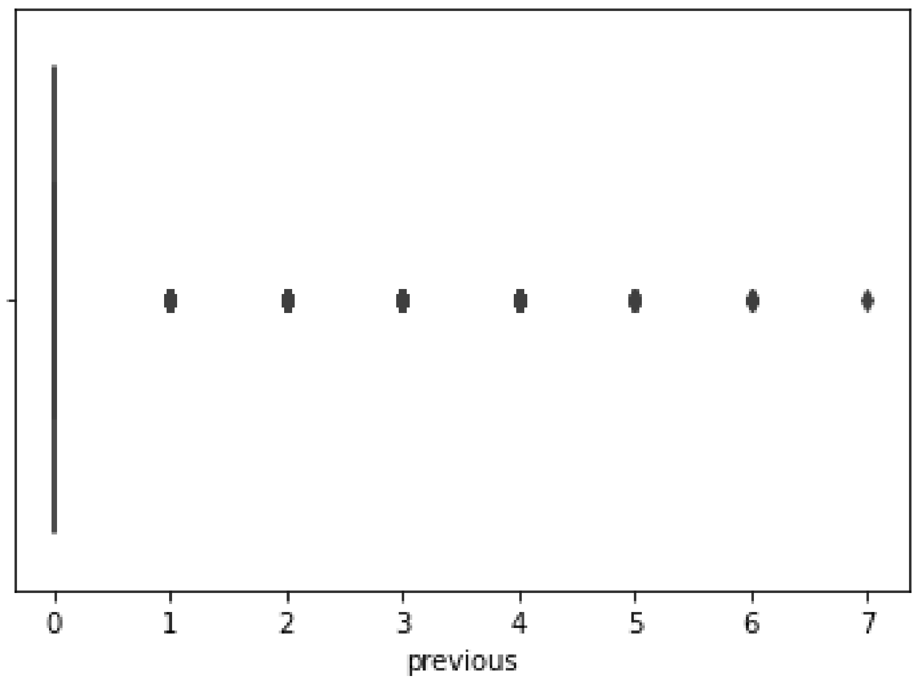
For the feature pdays we can see that 97% of the people has not been contacted before and given the value so it’s better to delete this particular feature which contribute very less variation but adding to the dimensionality

Outlier Detection









From the boxplot we can see that age, previous, duration, campaign is having outliers.

Since duration cannot be used for the prediction, we can ignore that

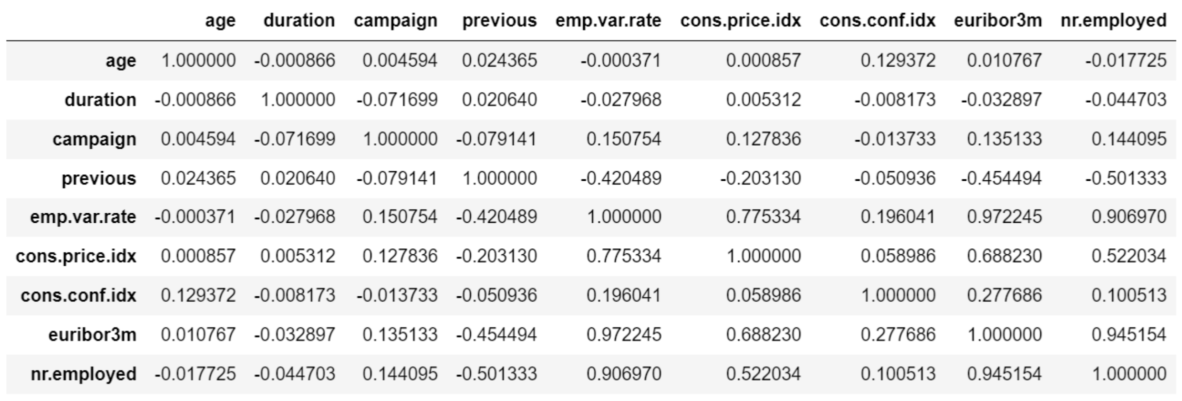
The values for age and previous are practical values and should not be deleted.

The campaign feature is having some outliers and might be unrealistic but we can't be so sure. So, it is better we keep the outliers as much and proceed with the model building.

The outliers may not affect when we do a standardization.

Dropping unnecessary columns if multicollinearity exists

We will plot the correlation matrix for checking if there is any multicollinearity



	age	duration	campaign	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
age	1.000000	-0.000866	0.004594	0.024365	-0.000371	0.000857	0.129372	0.010767	-0.017725
duration	-0.000866	1.000000	-0.071699	0.020640	-0.027968	0.005312	-0.008173	-0.032897	-0.044703
campaign	0.004594	-0.071699	1.000000	-0.079141	0.150754	0.127836	-0.013733	0.135133	0.144095
previous	0.024365	0.020640	-0.079141	1.000000	-0.420489	-0.203130	-0.050936	-0.454494	-0.501333
emp.var.rate	-0.000371	-0.027968	0.150754	-0.420489	1.000000	0.775334	0.196041	0.972245	0.906970
cons.price.idx	0.000857	0.005312	0.127836	-0.203130	0.775334	1.000000	0.058986	0.688230	0.522034
cons.conf.idx	0.129372	-0.008173	-0.013733	-0.050936	0.196041	0.058986	1.000000	0.277686	0.100513
euribor3m	0.010767	-0.032897	0.135133	-0.454494	0.972245	0.688230	0.277686	1.000000	0.945154
nr.employed	-0.017725	-0.044703	0.144095	-0.501333	0.906970	0.522034	0.100513	0.945154	1.000000

From this correlation matrix we can see that euribor3m and emp.var.rate are having a correlation of 0.972.

So, we can choose one to keep. We will drop euribor3m since it has high correlation with nr.employed also.

We have age columns where have done discretization. We can keep one discretized column and drop the others.

Dropping duration since it is not a metric we get before the call is made and is mentioned in the dataset itself.

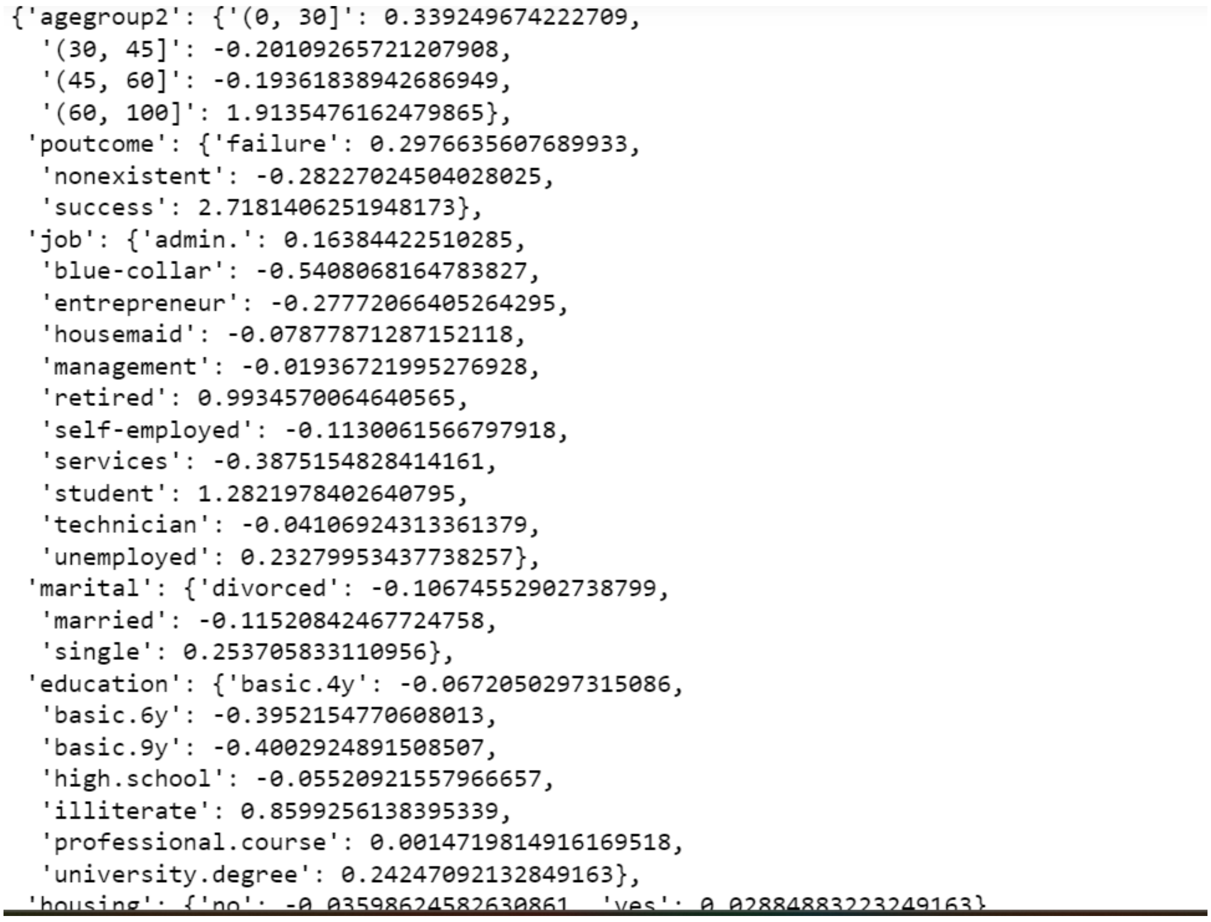
Weight of Evidence Encoding

Weight of Evidence (WoE) was developed primarily for the credit and financial industries to help build more predictive models to evaluate the risk of loan default. That is, to predict how likely the money lent to a person or institution is to be lost. Thus, Weight of Evidence is a measure of the "strength" of a grouping technique to separate good and bad risk (default).

WoE will be 0 if the P(Goods) / P(Bads) = 1, that is, if the outcome is random for that group. If P(Bads) > P(Goods) the odds ratio will be < 1 and, WoE will be < 0 if, P(Goods) > P(Bads). WoE is well suited for Logistic Regression, because the Logit transformation is simply the log of the odds, i.e., ln(P(Goods)/P(Bads)). Therefore, by using WoE-coded predictors in logistic regression, the predictors are all prepared and coded to the same scale, and the parameters in the linear logistic regression equation can be directly compared.

The WoE transformation has three advantages:

It creates a monotonic relationship between the target and the independent variables. It orders the categories on a "logistic" scale which is natural for logistic regression The transformed variables can then be compared because they are on the same scale. Therefore, it is possible to determine which one is more predictive.



```
{'agegroup2': {'(0, 30]': 0.339249674222709,
  '(30, 45]': -0.20109265721207908,
  '(45, 60]': -0.19361838942686949,
  '(60, 100]': 1.9135476162479865},
 'poutcome': {'failure': 0.2976635607689933,
  'nonexistent': -0.28227024504028025,
  'success': 2.7181406251948173},
 'job': {'admin.': 0.16384422510285,
  'blue-collar': -0.5408068164783827,
  'entrepreneur': -0.27772066405264295,
  'housemaid': -0.07877871287152118,
  'management': -0.01936721995276928,
  'retired': 0.9934570064640565,
  'self-employed': -0.1130061566797918,
  'services': -0.3875154828414161,
  'student': 1.2821978402640795,
  'technician': -0.04106924313361379,
  'unemployed': 0.23279953437738257},
 'marital': {'divorced': -0.10674552902738799,
  'married': -0.11520842467724758,
  'single': 0.253705833110956},
 'education': {'basic.4y': -0.0672050297315086,
  'basic.6y': -0.3952154770608013,
  'basic.9y': -0.4002924891508507,
  'high.school': -0.05520921557966657,
  'illiterate': 0.8599256138395339,
  'professional.course': 0.0014719814916169518,
  'university.degree': 0.24247092132849163},
 'housing': {'no': -0.03598624582630861, 'yes': 0.02884883223249163}
```

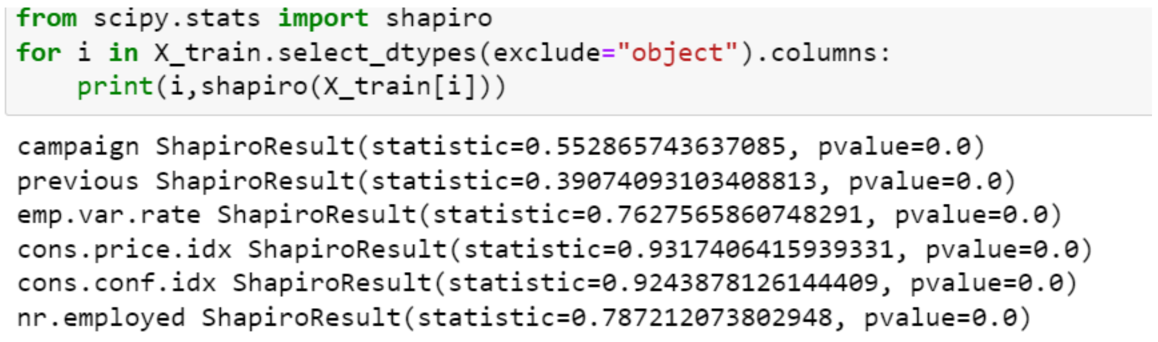
After encoding all the columns has become numerical values and is ready to go for model building

Transformation

Transformation is done when the variables are not normally distributed and we wanted to make them normally distributed Since we have negative values also we are going with yeo_johnson method of powertransformer.

First we checked with Shapiro test to find whether the data is normal or not.

since all p values are less than 0.05 none of the parameters are normally distributed



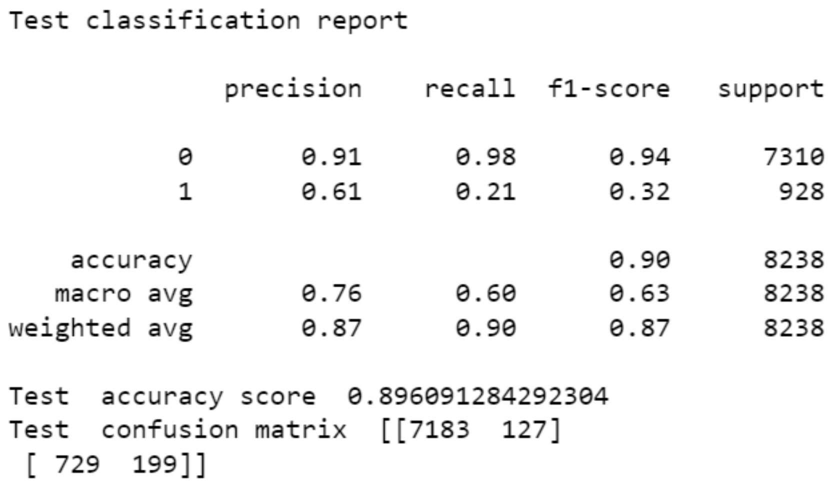
```
from scipy.stats import shapiro
for i in X_train.select_dtypes(exclude="object").columns:
    print(i,shapiro(X_train[i]))
```

```
campaign ShapiroResult(statistic=0.552865743637085, pvalue=0.0)
previous ShapiroResult(statistic=0.39074093103408813, pvalue=0.0)
emp.var.rate ShapiroResult(statistic=0.7627565860748291, pvalue=0.0)
cons.price.idx ShapiroResult(statistic=0.9317406415939331, pvalue=0.0)
cons.conf.idx ShapiroResult(statistic=0.924387812614409, pvalue=0.0)
nr.employed ShapiroResult(statistic=0.787212073802948, pvalue=0.0)
```

So we used yeo johnson power transformer to transform the data into normal

Model After Transformation + WoE encoding

We are using Logistic Regression to compare the performance of different preprocessing done on the dataset



```
Test classification report

              precision    recall  f1-score   support

           0       0.91      0.98      0.94      7310
           1       0.61      0.21      0.32       928

    accuracy                           0.90      8238
   macro avg       0.76      0.60      0.63      8238
weighted avg       0.87      0.90      0.87      8238

Test  accuracy score  0.896091284292304
Test  confusion matrix  [[7183  127]
 [ 729  199]]
```

.

Scaling

We can use Robust scaler for transformation as the dataset contains outliers.

One approach to standardizing input variables in the presence of outliers is to ignore the outliers from the calculation of the mean and standard deviation, then use the calculated values to scale the variable.

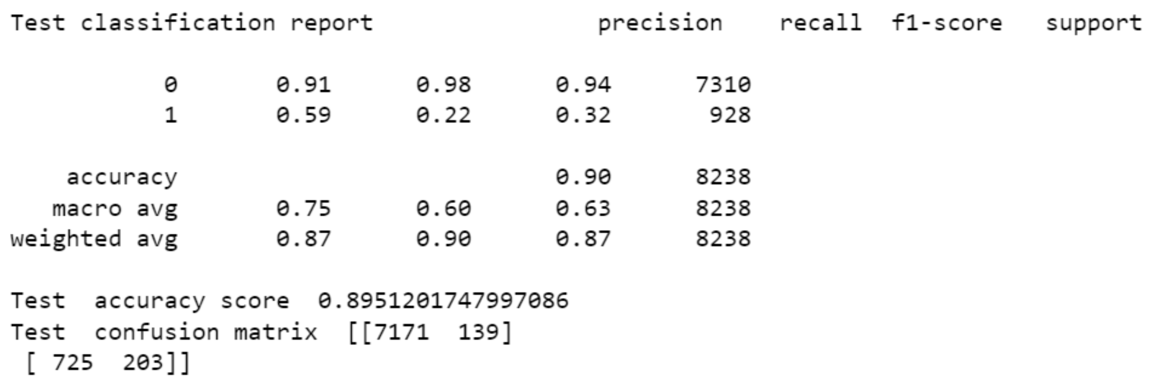
This is called robust standardization or robust data scaling.

This can be achieved by calculating the median (50th percentile) and the 25th and 75th percentiles. The values of each variable then have their median subtracted and are divided by the interquartile range (IQR) which is the difference between the 75th and 25th percentiles.

value = (value – median) / (p75 – p25)

The resulting variable has a zero mean and median and a standard deviation of 1, although not skewed by outliers and the outliers are still present with the same relative relationships to other values.

Model After Transformation + WoE encoding +Robustscaling



```
Test classification report                precision    recall  f1-score   support

           0       0.91      0.98      0.94      7310
           1       0.59      0.22      0.32       928

    accuracy                           0.90      8238
   macro avg       0.75      0.60      0.63      8238
weighted avg       0.87      0.90      0.87      8238

Test  accuracy score  0.8951201747997086
Test  confusion matrix  [[7171  139]
 [ 725  203]]
```

There is no impact of scaling on the model accuracy

One reason for this maybe majority of the features have been encoded using log and numerical features have been transformed and standardized.

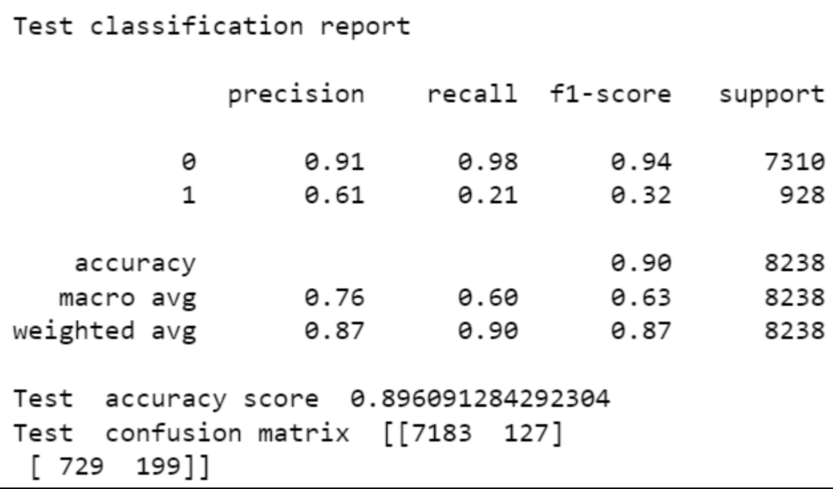
Recursive Feature Elimination

In the linear regression module, we learn about various techniques for selecting the significant features in the dataset. In this example, let us consider the RFE method for feature selection.

We build the whole model and the beta which is least significant is eliminated. Least beta value will be the least significant value. We can eliminate more than 2 at a time. How does it work. It assigns a rank. Say we have 10 features. The first feature that requires elimination is given the highest rank. That is 10. Second feature to be eliminated is given rank of 9. If the ask is for top 3(as mentioned in the parameter) then it assigns values from 10,9,8,7,6,5,4. Top 3 are given the value of 1. At this point, it does not eliminate the features but gives the ranking. We take the decision to eliminate and pull out the only one which is important.

From Recursive feature elimination we got the best 7 features for prediction as 'contact', 'previous', 'poutcome', 'emp.var.rate', 'cons.price.idx', 'cons.conf.idx' and 'nr. employed'.

Model After Transformation + WoE encoding + RFE

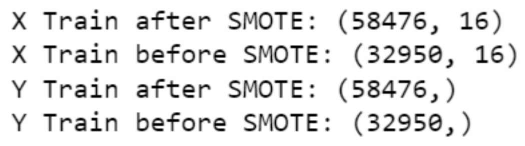


```
Test classification report

              precision    recall  f1-score   support

           0       0.91      0.98      0.94      7310
           1       0.61      0.21      0.32       928

    accuracy                           0.90      8238
   macro avg       0.76      0.60      0.63      8238
weighted avg       0.87      0.90      0.87      8238

Test  accuracy score  0.896091284292304
Test  confusion matrix  [[7183  127]
 [ 729  199]]
```

The model with features from RFE is actually decreasing the model F1 score eventhough the precision is increasing slightly

SMOTE

SMOTE stands for Synthetic Minority Oversampling Technique. The method was proposed in a 2002 paper in the Journal of Artificial Intelligence Research. SMOTE is an improved method of dealing with imbalanced data in classification problems.



```
X Train after SMOTE: (58476, 16)
X Train before SMOTE: (32950, 16)
Y Train after SMOTE: (58476,)
Y Train before SMOTE: (32950,)
```

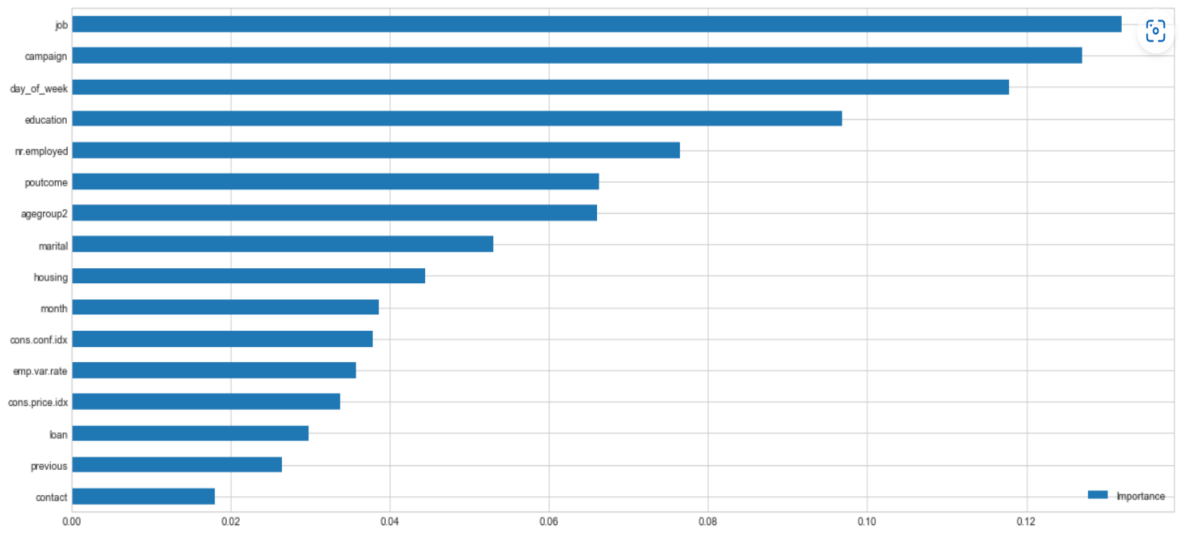
Model after Transformation+WoE+SMOTE

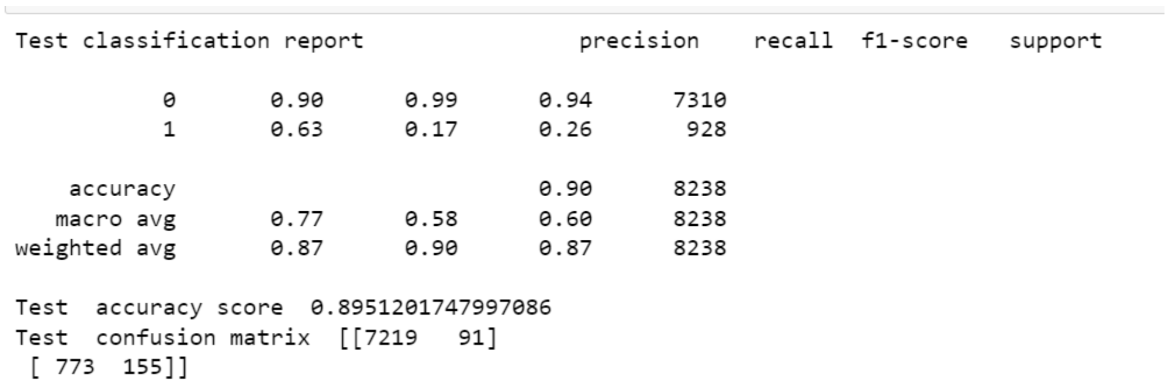
```
Test classification report

              precision    recall  f1-score   support

           0       0.95      0.73      0.83      7310
           1       0.25      0.70      0.37       928

    accuracy                           0.73      8238
   macro avg       0.60      0.71      0.60      8238
weighted avg       0.87      0.73      0.77      8238

Test  accuracy score  0.725782957028405
Test  confusion matrix  [[5329 1981]
 [ 278  650]]
```

SMOTE has increased the recall of the minority class but the precision went down as much. When we see the F1 score of minority class it went up by 0.4 only but the total accuracy went down from 0.9 to 0.73

Feature Importance

Random forest consists of a number of decision trees. Every node in the decision trees is a condition on a single feature, designed to split the dataset into two so that similar response values end up in the same set. The measure based on which the (locally) optimal condition is chosen is called impurity. When training a tree, it can be computed how much each feature decreases the weighted impurity in a tree. For a forest, the impurity decrease from each feature can be averaged and the features are ranked according to this measure. This is the feature importance measure exposed in sklearn's Random Forest implementations.



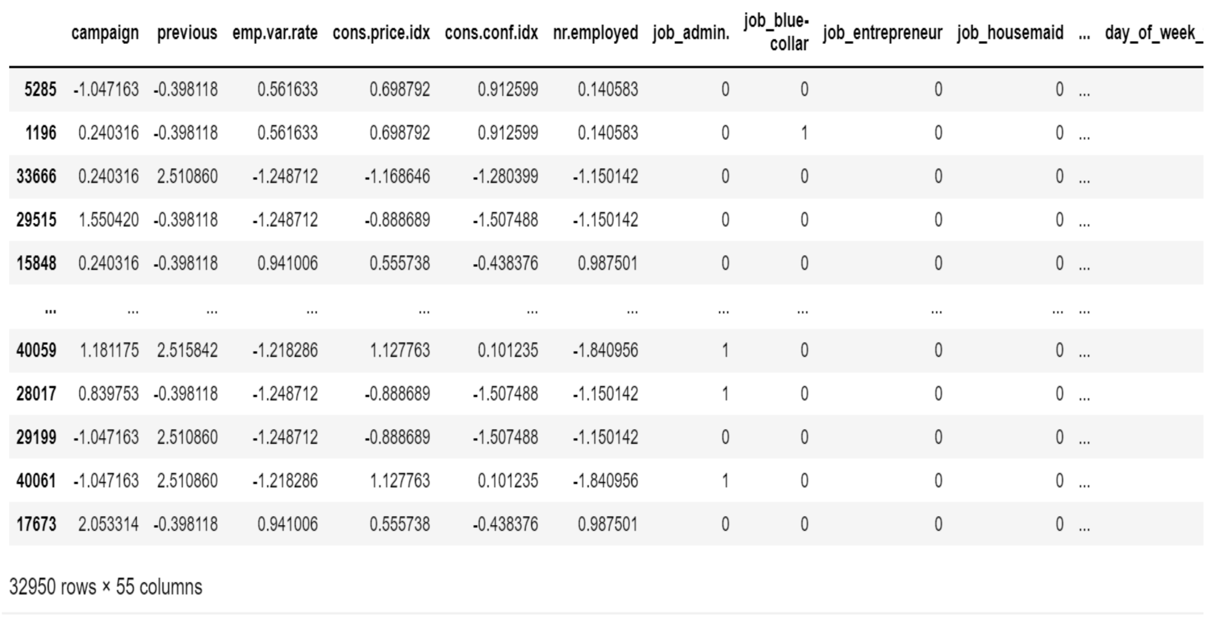


```
Test classification report                precision    recall  f1-score   support

           0       0.90      0.99      0.94      7310
           1       0.63      0.17      0.26       928

    accuracy                           0.90      8238
   macro avg       0.77      0.58      0.60      8238
weighted avg       0.87      0.90      0.87      8238

Test  accuracy score  0.8951201747997086
Test  confusion matrix  [[7219   91]
 [ 773  155]]
```

The Model built from the features obtained from feature importance is also not contributing to model performance

Doing One Hot encoding

We initially have done WoE encoding.Now we are trying to do one hot encoding instead of WoE and assess if there is any change in model performance

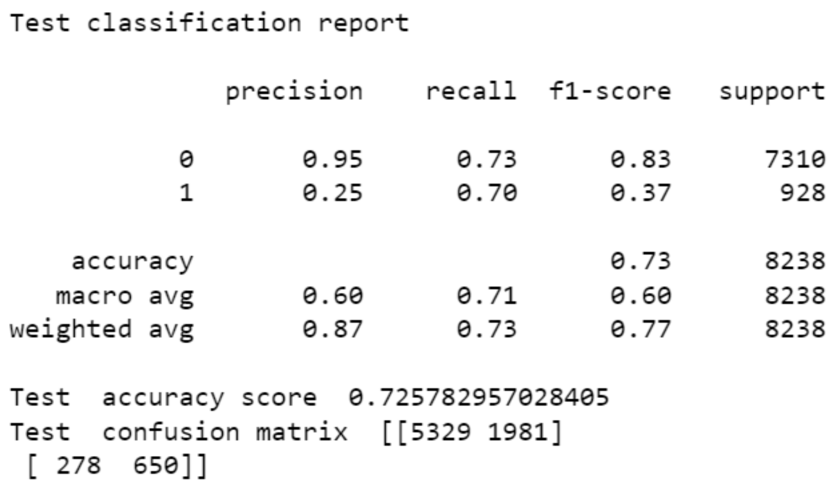


	campaign	previous	emp.var.rate	cons.price.idx	cons.conf.idx	nr.employed	job_admin.	job_blue-collar	job_entrepreneur	job_housemaid	...	day_of_week_
5285	-1.047163	-0.398118	0.561633	0.698792	0.912599	0.140583	0	0	0	0	...	
1196	0.240316	-0.398118	0.561633	0.698792	0.912599	0.140583	0	1	0	0	...	
33666	0.240316	2.510860	-1.248712	-1.168646	-1.280399	-1.150142	0	0	0	0	...	
29515	1.550420	-0.398118	-1.248712	-0.888689	-1.507488	-1.150142	0	0	0	0	...	
15848	0.240316	-0.398118	0.941006	0.555738	-0.438376	0.987501	0	0	0	0	...	
...	...	...	...	...	...	...	...	...	...	...	...	
40059	1.181175	2.515842	-1.218286	1.127763	0.101235	-1.840956	1	0	0	0	...	
28017	0.839753	-0.398118	-1.248712	-0.888689	-1.507488	-1.150142	1	0	0	0	...	
29199	-1.047163	2.510860	-1.248712	-0.888689	-1.507488	-1.150142	0	0	0	0	...	
40061	-1.047163	2.510860	-1.218286	1.127763	0.101235	-1.840956	1	0	0	0	...	
17673	2.053314	-0.398118	0.941006	0.555738	-0.438376	0.987501	0	0	0	0	...	

32950 rows × 55 columns

After One hot we got 55 columns and with this new dataset we are going to do model building

Model after Transformation + One Hot Encoding



```
Test classification report

              precision    recall  f1-score   support

           0       0.95      0.73      0.83      7310
           1       0.25      0.70      0.37       928

    accuracy                           0.73      8238
   macro avg       0.60      0.71      0.60      8238
weighted avg       0.87      0.73      0.77      8238

Test  accuracy score  0.725782957028405
Test  confusion matrix  [[5329 1981]
 [ 278  650]]
```

One Hot encoding also give similar results to that of WoE encoding

Conclusions after preprocessing

1.After trying out different methods we understood that encoding using WoE and Power transformation works best for our model.

2.SMOTE was tried, It improved the test F1 Score of minority class a little.

3.But it decreased the train and test precision also the total accuracy of both.

4.Class_weight = balanced also give similar results to SMOTE.

Model Building

We are going to use different algorithms to find the best model for the business problem

The proposed Algorithms are

- Logistic Regression

- Random Forest

- Naive Bayes

- Linear SVC

- Decision Tree

- Gradient Boosted Trees

- BaggingClassifier

- AdaBoostClassifier

- XGBoost

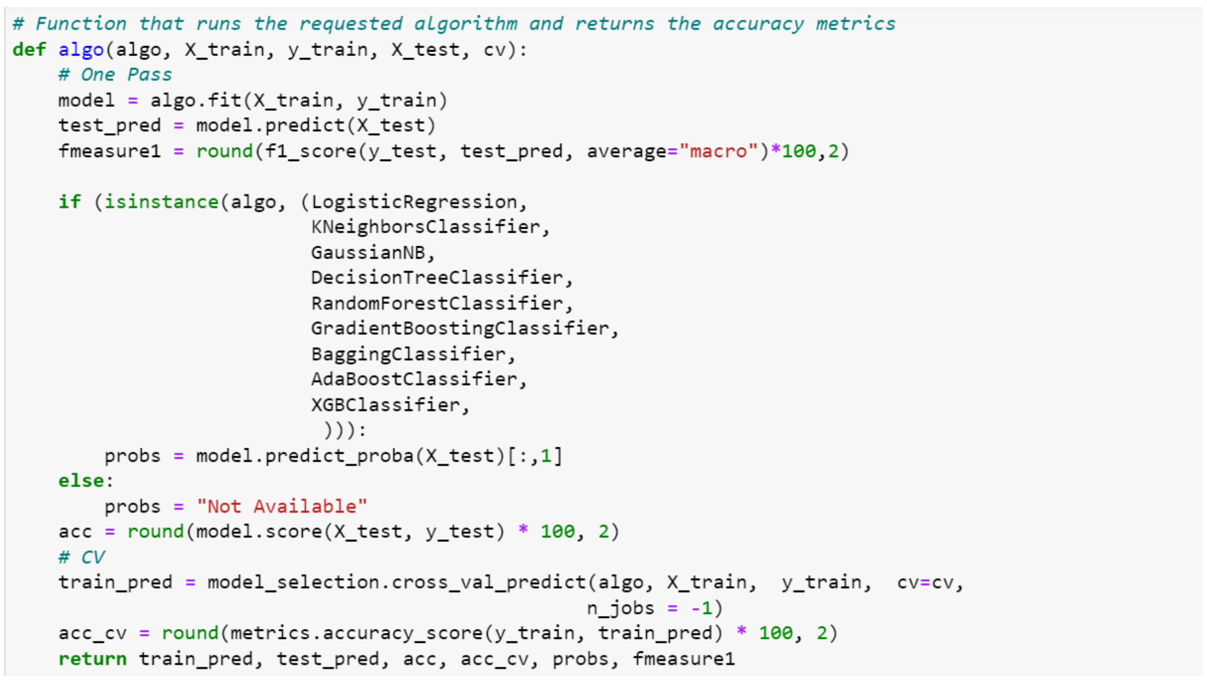
- Multiple Ensemble – Voting Classifier

<u>User Defined Functions</u>

There is a possibility that there will be lot of repetition of code since we are using multiple algorithms to compare.

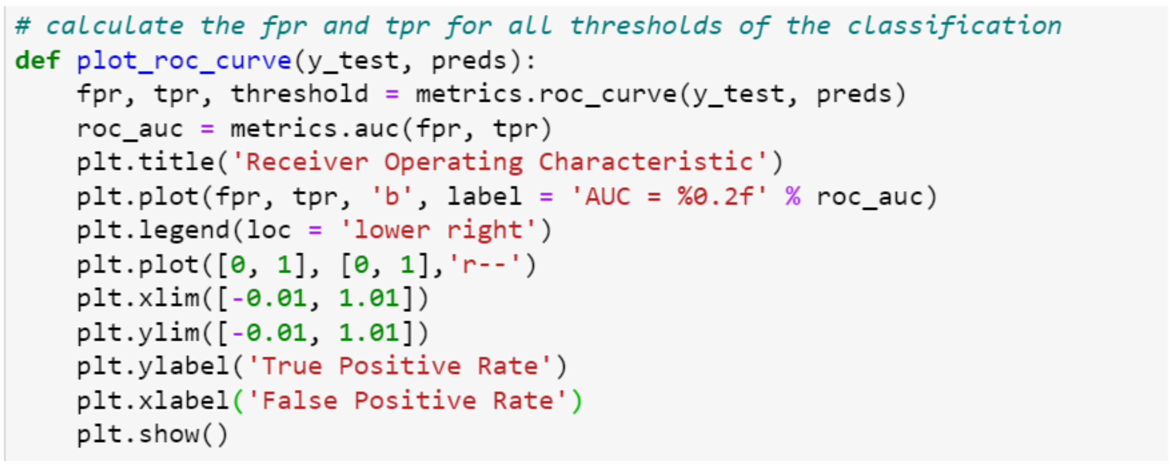
User defined functions are written to avoid this repetition

- Function to run the defined algorithms and return accuracy metrics



```python
# Function that runs the requested algorithm and returns the accuracy metrics
def algo(algo, X_train, y_train, X_test, cv):
    # One Pass
    model = algo.fit(X_train, y_train)
    test_pred = model.predict(X_test)
    fmeasure1 = round(f1_score(y_test, test_pred, average="macro")*100,2)

    if (isinstance(algo, (LogisticRegression,
                          KNeighborsClassifier,
                          GaussianNB,
                          DecisionTreeClassifier,
                          RandomForestClassifier,
                          GradientBoostingClassifier,
                          BaggingClassifier,
                          AdaBoostClassifier,
                          XGBClassifier,
                           ))):
        probs = model.predict_proba(X_test)[:,1]
    else:
        probs = "Not Available"
    acc = round(model.score(X_test, y_test) * 100, 2)
    # CV
    train_pred = model_selection.cross_val_predict(algo, X_train,  y_train,  cv=cv,
                                                   n_jobs = -1)
    acc_cv = round(metrics.accuracy_score(y_train, train_pred) * 100, 2)
    return train_pred, test_pred, acc, acc_cv, probs, fmeasure1
```

- To calculate the FPR and TPR for all thresholds of the classification



```
# calculate the fpr and tpr for all thresholds of the classification
def plot_roc_curve(y_test, preds):
    fpr, tpr, threshold = metrics.roc_curve(y_test, preds)
    roc_auc = metrics.auc(fpr, tpr)
    plt.title('Receiver Operating Characteristic')
    plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
    plt.legend(loc = 'lower right')
    plt.plot([0, 1], [0, 1],'r--')
    plt.xlim([-0.01, 1.01])
    plt.ylim([-0.01, 1.01])
    plt.ylabel('True Positive Rate')
    plt.xlabel('False Positive Rate')
    plt.show()
```

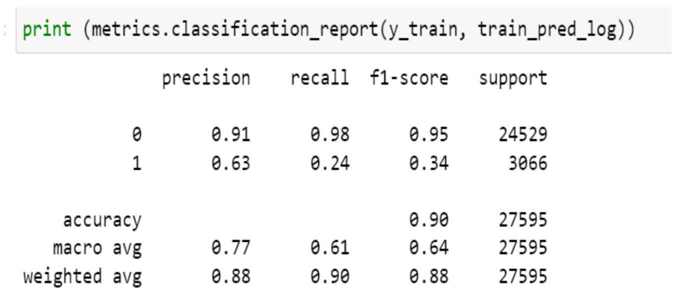
LOGISTIC REGRESSION

Logistic regression is a binary classification algorithm.

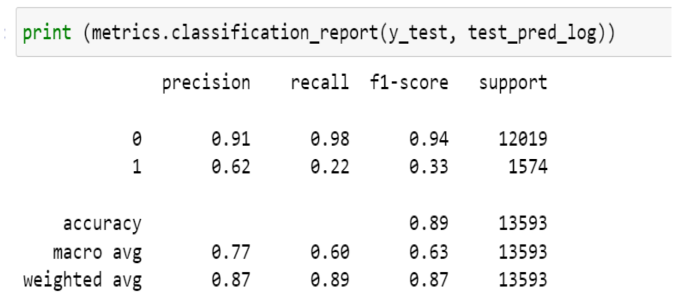
It predicts the probability of occurrence of a class label.

Based on these probabilities the data points are labelled based on a threshold (or cut-off; commonly a threshold of 0.5 is used) which is fixed.

Classification Report

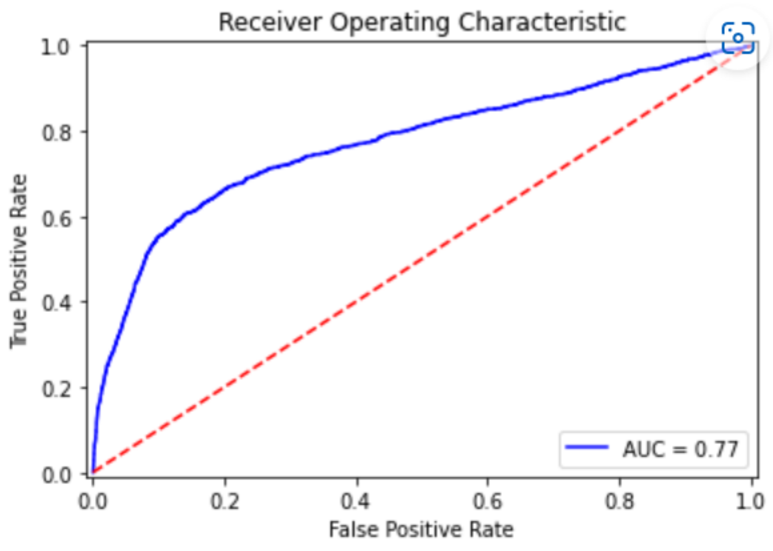


```
print (metrics.classification_report(y_train, train_pred_log))

              precision    recall  f1-score   support

           0       0.91      0.98      0.95     24529
           1       0.63      0.24      0.34      3066

    accuracy                           0.90     27595
   macro avg       0.77      0.61      0.64     27595
weighted avg       0.88      0.90      0.88     27595
```



```
print (metrics.classification_report(y_test, test_pred_log))

              precision    recall  f1-score   support

           0       0.91      0.98      0.94     12019
           1       0.62      0.22      0.33      1574

    accuracy                           0.89     13593
   macro avg       0.77      0.60      0.63     13593
weighted avg       0.87      0.89      0.87     13593
```

ROC Curve

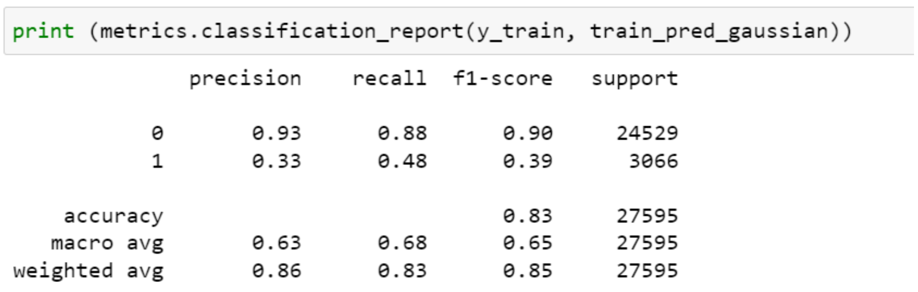


Naive Bayes

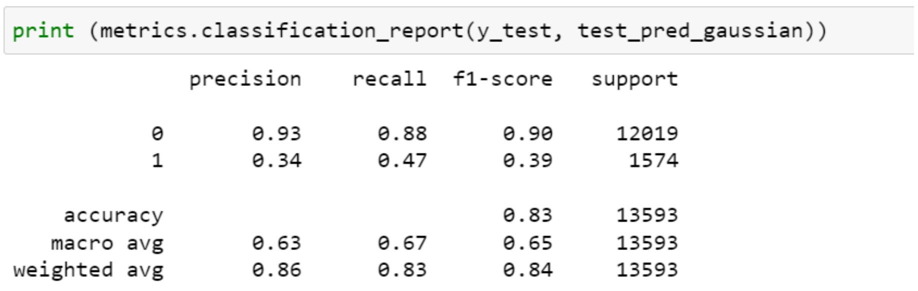
It uses a Bayes' Theorem with the assumption of independence of predictor variables.

The sklearn library provides different naive bayes classifiers, as GaussianNB, MultinomialNB and so on.

Classification Report

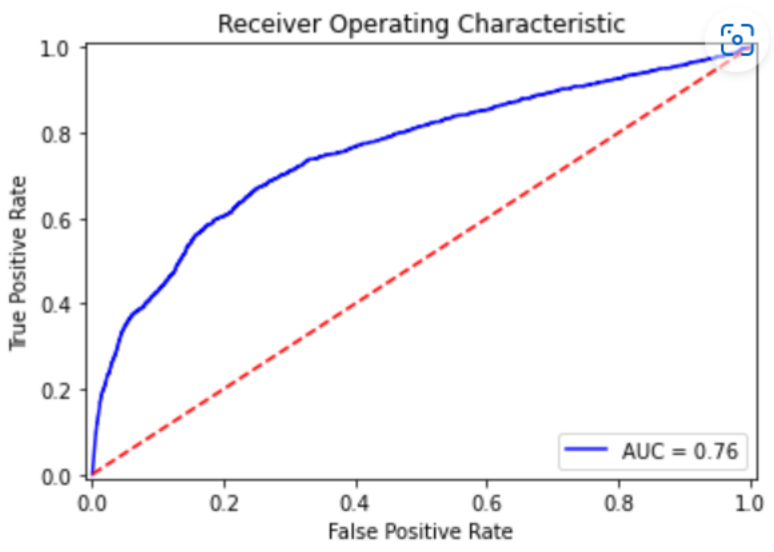


```
print (metrics.classification_report(y_train, train_pred_gaussian))

              precision    recall  f1-score   support

           0       0.93      0.88      0.90     24529
           1       0.33      0.48      0.39      3066

    accuracy                           0.83     27595
   macro avg       0.63      0.68      0.65     27595
weighted avg       0.86      0.83      0.85     27595
```



```
print (metrics.classification_report(y_test, test_pred_gaussian))

              precision    recall  f1-score   support

           0       0.93      0.88      0.90     12019
           1       0.34      0.47      0.39      1574

    accuracy                           0.83     13593
   macro avg       0.63      0.67      0.65     13593
weighted avg       0.86      0.83      0.84     13593
```

ROC Curve



Decision Tree Classifier

A decision node is a node on which a decision of split is to be made. A node that cannot be split further is known as the terminal/leaf node. A leaf node represents the decision. A decision tree can work with both numerical and categorical variables.

Classification Report

```
]: print (metrics.classification_report(y_train, train_pred_dt))

              precision    recall  f1-score   support

           0       0.92      0.92      0.92     24529
           1       0.33      0.32      0.32      3066

    accuracy                           0.85     27595
   macro avg       0.62      0.62      0.62     27595
weighted avg       0.85      0.85      0.85     27595
```

```
]: print (metrics.classification_report(y_test, test_pred_dt))

              precision    recall  f1-score   support

           0       0.91      0.92      0.91     12019
           1       0.32      0.29      0.31      1574

    accuracy                           0.85     13593
   macro avg       0.61      0.61      0.61     13593
weighted avg       0.84      0.85      0.84     13593
```

RANDOM FOREST

It is the method of constructing multiple decision trees on randomly selected data samples. We can use the bootstrap sampling method to select the random samples of the same size from the dataset to construct multiple trees. This method is used for both regression and classification analysis.
The random forest returns the prediction based on all the individual decision trees prediction.
It avoids the over-fitting problem as it considers a random data sample to construct a decision tree.

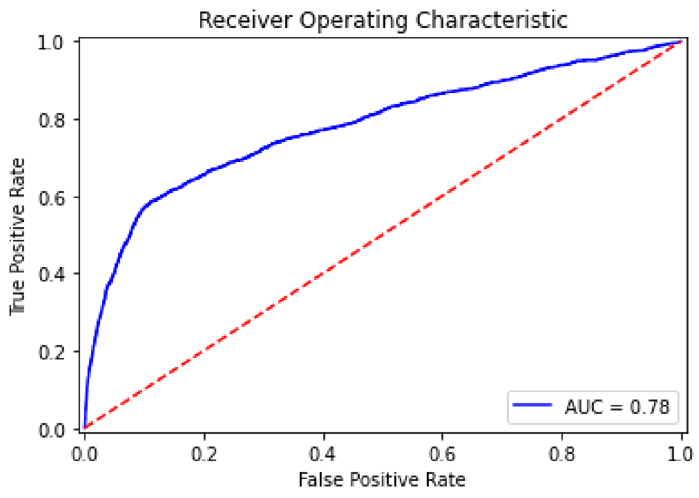
Classification Report

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.95     24529
           1       0.65      0.25      0.36      3066

    accuracy                           0.90     27595
   macro avg       0.78      0.62      0.65     27595
weighted avg       0.88      0.90      0.88     27595
```

```
: print (metrics.classification_report(y_test, test_pred_rf))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     12019
           1       0.64      0.22      0.32      1574

    accuracy                           0.90     13593
   macro avg       0.77      0.60      0.63     13593
weighted avg       0.87      0.90      0.87     13593
```

ROC Curve



ADA Boost

The model creates several stumps (decision tree with only a single decision node and two leaf nodes) on the train set and predicts the class based on these weak learners (stumps).

For the first model, it assigns equal weights to each sample. It assigns the higher weight for the wrongly predicted samples and lower weight for the correctly predicted samples. This method continues till all the observations are correctly classified or the predefined number of stumps is created.

Classification Report

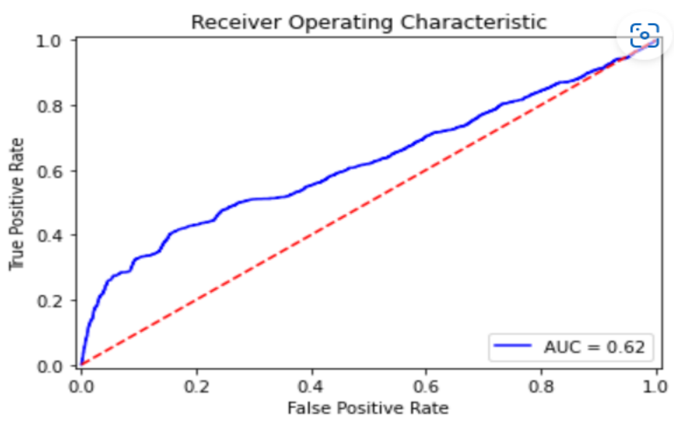
```
print (metrics.classification_report(y_train, train_pred_abcl))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     24529
           1       0.40      0.29      0.34      3066

    accuracy                           0.87     27595
   macro avg       0.66      0.62      0.64     27595
weighted avg       0.86      0.87      0.86     27595
```

```
print (metrics.classification_report(y_test, test_pred_abcl))
```

```
              precision    recall  f1-score   support

           0       0.91      0.95      0.93     12019
           1       0.40      0.27      0.32      1574

    accuracy                           0.87     13593
   macro avg       0.65      0.61      0.62     13593
weighted avg       0.85      0.87      0.86     13593
```

ROC Curve



Gradient Boost

This method optimizes the differentiable loss function by building the number of weak learners (decision trees) sequentially.
It considers the residuals from the previous model and fits the next model to the residuals. The algorithm uses a gradient descent method to minimize the error.

Classification Report

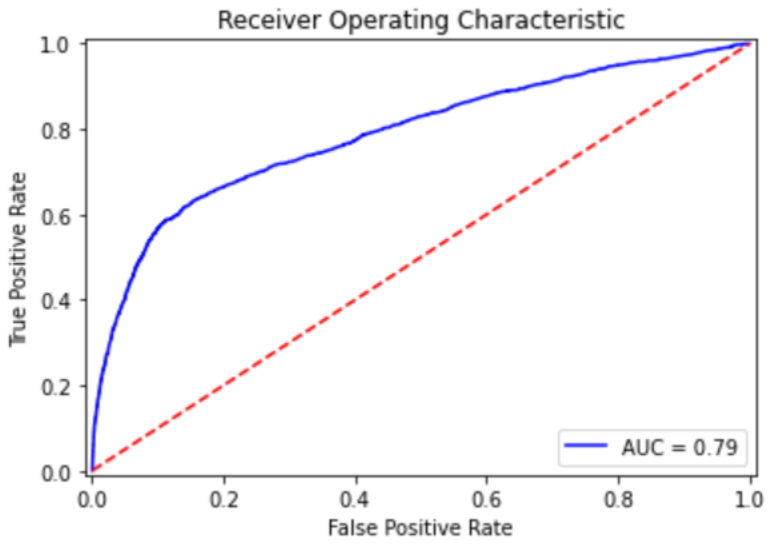
```
print (metrics.classification_report(y_train, train_pred_gbt))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.95     24529
           1       0.67      0.26      0.38      3066

    accuracy                           0.90     27595
   macro avg       0.79      0.62      0.66     27595
weighted avg       0.89      0.90      0.88     27595
```

```
print (metrics.classification_report(y_test, test_pred_gbt))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     12019
           1       0.64      0.24      0.35      1574

    accuracy                           0.90     13593
   macro avg       0.78      0.61      0.65     13593
weighted avg       0.88      0.90      0.87     13593
```

ROC Curve

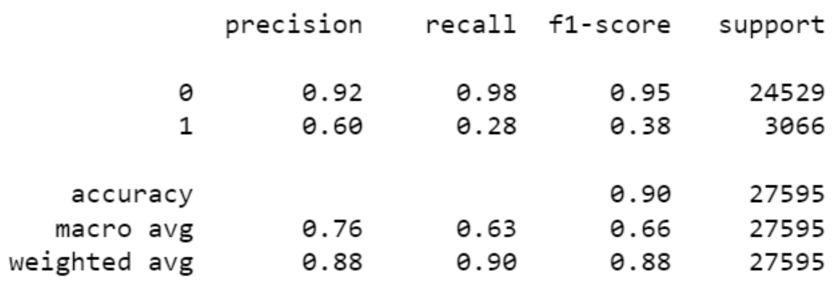


XGBoost

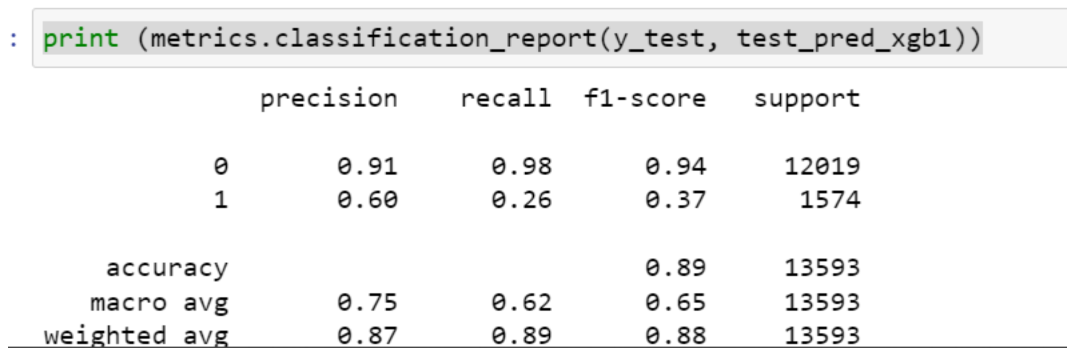
XGBoost (extreme gradient boost) is an alternative form of gradient boosting method. This method generally considers the initial prediction as 0.5 and build the decision tree to predict the residuals.

It considers the regularization parameter to avoid overfitting.

Classification Report



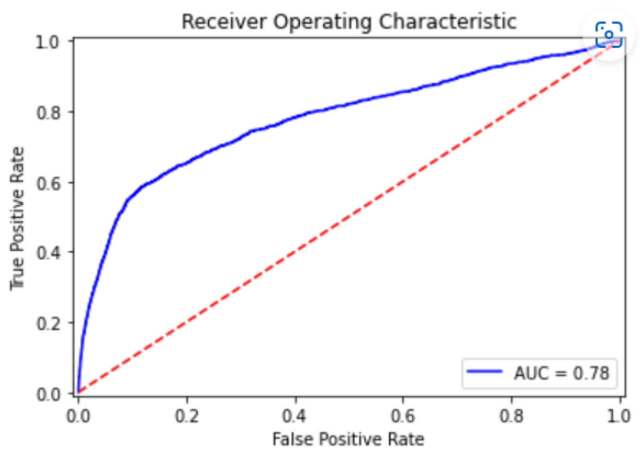
```
              precision    recall  f1-score   support

           0       0.92      0.98      0.95     24529
           1       0.60      0.28      0.38      3066

    accuracy                           0.90     27595
   macro avg       0.76      0.63      0.66     27595
weighted avg       0.88      0.90      0.88     27595
```



```
: print (metrics.classification_report(y_test, test_pred_xgb1))
```

```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94     12019
           1       0.60      0.26      0.37      1574

    accuracy                           0.89     13593
   macro avg       0.75      0.62      0.65     13593
weighted avg       0.87      0.89      0.88     13593
```

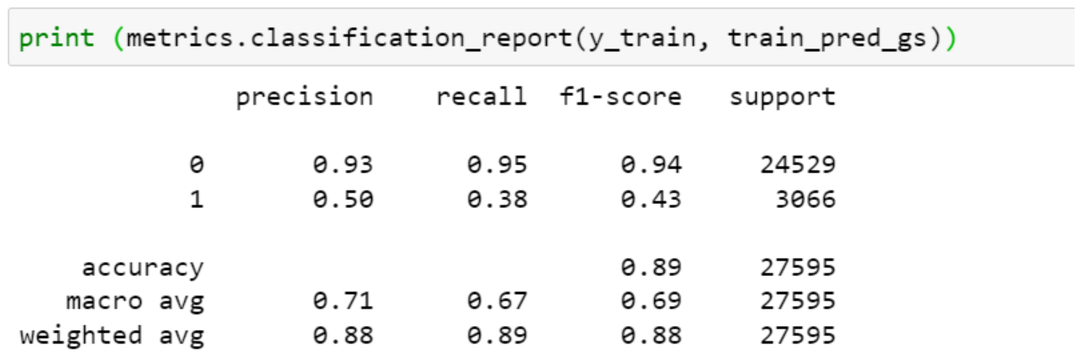
ROC Curve



Multiple model Ensemble – VotingClassifier

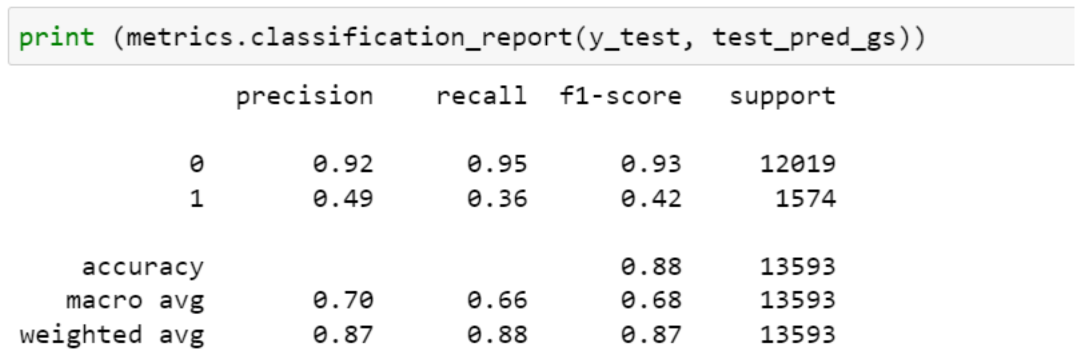
We are using three different algorithms and taking the vote of them to predict the output
Here we have used Logistic, Naïve bayes and Random Forest

Classification Report



```
print (metrics.classification_report(y_train, train_pred_gs))
```

```
              precision    recall  f1-score   support

           0       0.93      0.95      0.94     24529
           1       0.50      0.38      0.43      3066

    accuracy                           0.89     27595
   macro avg       0.71      0.67      0.69     27595
weighted avg       0.88      0.89      0.88     27595
```



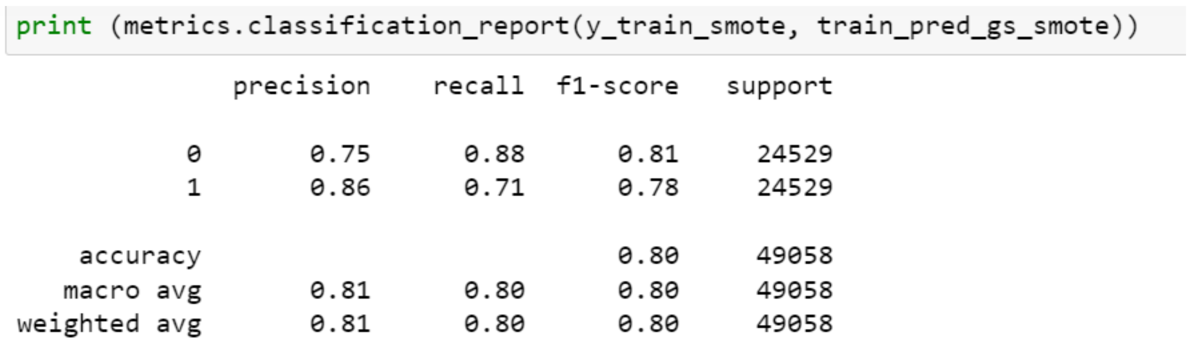
```
print (metrics.classification_report(y_test, test_pred_gs))
```

```
              precision    recall  f1-score   support

           0       0.92      0.95      0.93     12019
           1       0.49      0.36      0.42      1574

    accuracy                           0.88     13593
   macro avg       0.70      0.66      0.68     13593
weighted avg       0.87      0.88      0.87     13593
```

We are getting the best results with Multiple model Ensemble - VotingClassifier.We can do SMOTE and check for performance boost

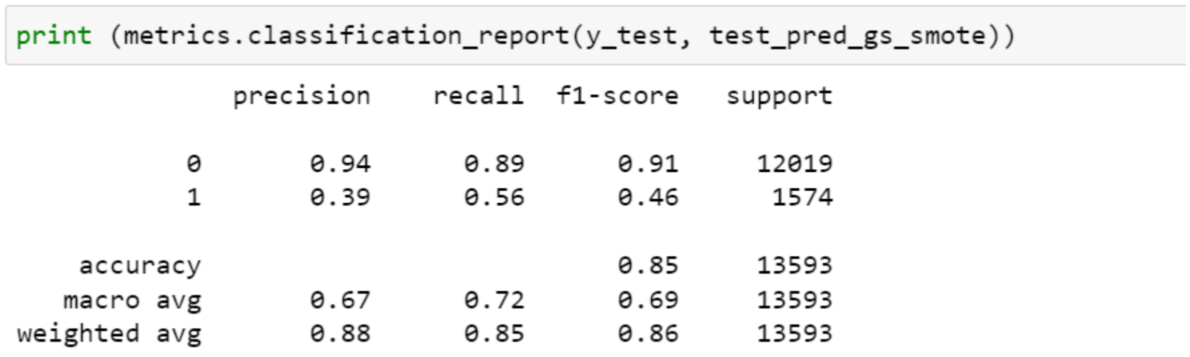
Voting Classifier after SMOTE

Classification Report



```
print (metrics.classification_report(y_train_smote, train_pred_gs_smote))
```

```
              precision    recall  f1-score   support

           0       0.75      0.88      0.81     24529
           1       0.86      0.71      0.78     24529

    accuracy                           0.80     49058
   macro avg       0.81      0.80      0.80     49058
weighted avg       0.81      0.80      0.80     49058
```



```
print (metrics.classification_report(y_test, test_pred_gs_smote))
```

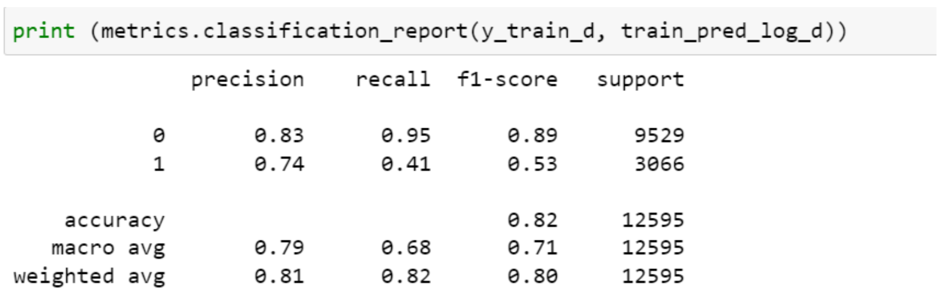
```
              precision    recall  f1-score   support

           0       0.94      0.89      0.91     12019
           1       0.39      0.56      0.46      1574

    accuracy                           0.85     13593
   macro avg       0.67      0.72      0.69     13593
weighted avg       0.88      0.85      0.86     13593
```

Downsampled Dataset

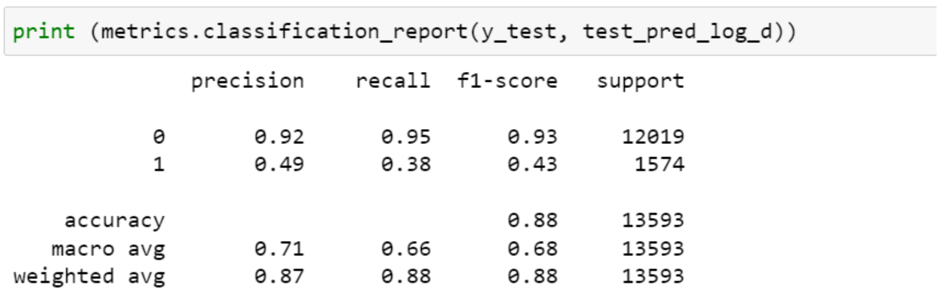
We are now going to downsample the train and see whether there is any change in the performance metrics after that.

Logistic- Downsampled

Classification Report



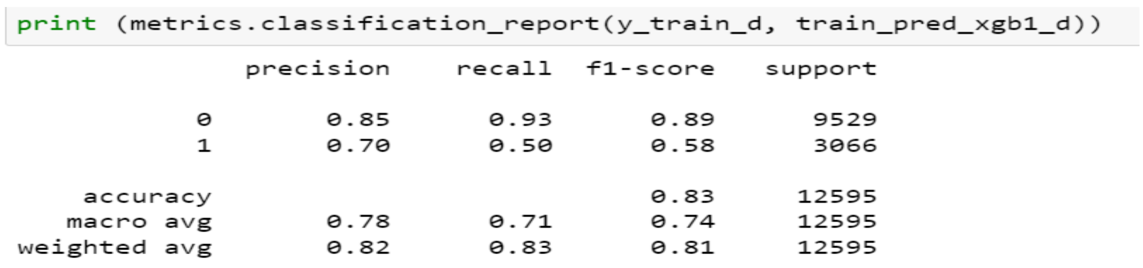
```
print (metrics.classification_report(y_train_d, train_pred_log_d))
              precision    recall  f1-score   support

           0       0.83      0.95      0.89      9529
           1       0.74      0.41      0.53      3066

    accuracy                           0.82     12595
   macro avg       0.79      0.68      0.71     12595
weighted avg       0.81      0.82      0.80     12595
```



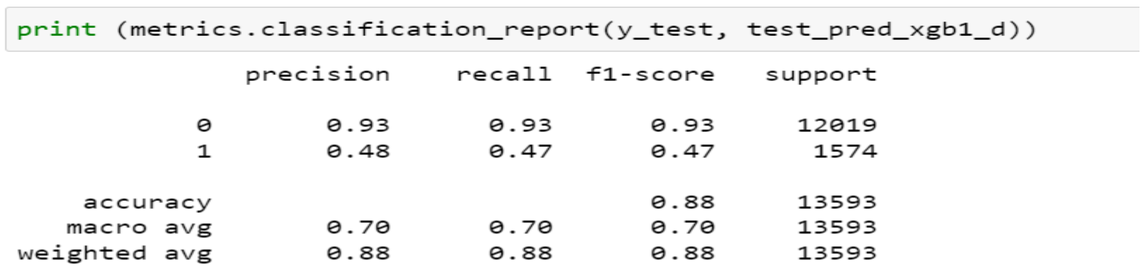
```
print (metrics.classification_report(y_test, test_pred_log_d))
              precision    recall  f1-score   support

           0       0.92      0.95      0.93     12019
           1       0.49      0.38      0.43      1574

    accuracy                           0.88     13593
   macro avg       0.71      0.66      0.68     13593
weighted avg       0.87      0.88      0.88     13593
```

XGB DownSampled

Classification Report

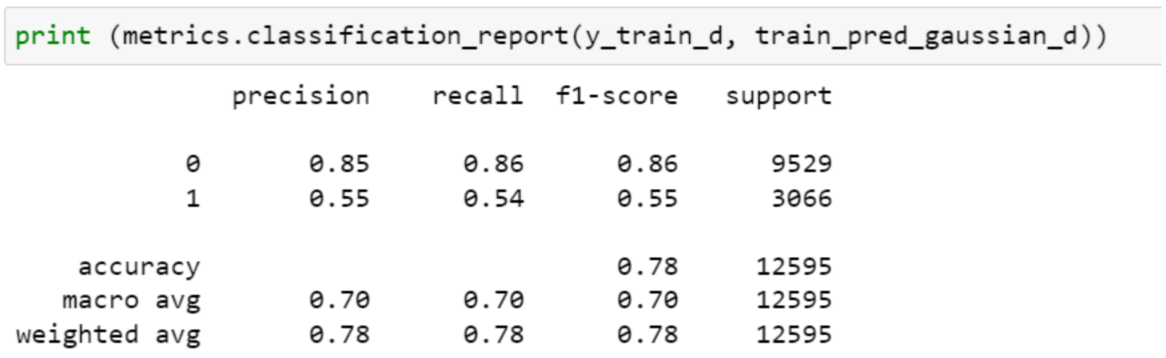


```
print (metrics.classification_report(y_train_d, train_pred_xgb1_d))
              precision    recall  f1-score   support

           0       0.85      0.93      0.89      9529
           1       0.70      0.50      0.58      3066

    accuracy                           0.83     12595
   macro avg       0.78      0.71      0.74     12595
weighted avg       0.82      0.83      0.81     12595
```



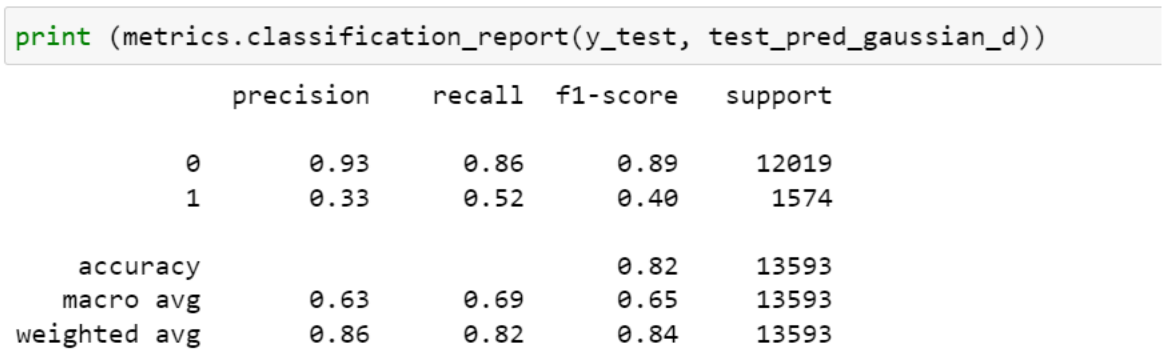
```
print (metrics.classification_report(y_test, test_pred_xgb1_d))
              precision    recall  f1-score   support

           0       0.93      0.93      0.93     12019
           1       0.48      0.47      0.47      1574

    accuracy                           0.88     13593
   macro avg       0.70      0.70      0.70     13593
weighted avg       0.88      0.88      0.88     13593
```

Naïve Bayes – Downsampled



```
print (metrics.classification_report(y_train_d, train_pred_gaussian_d))
```

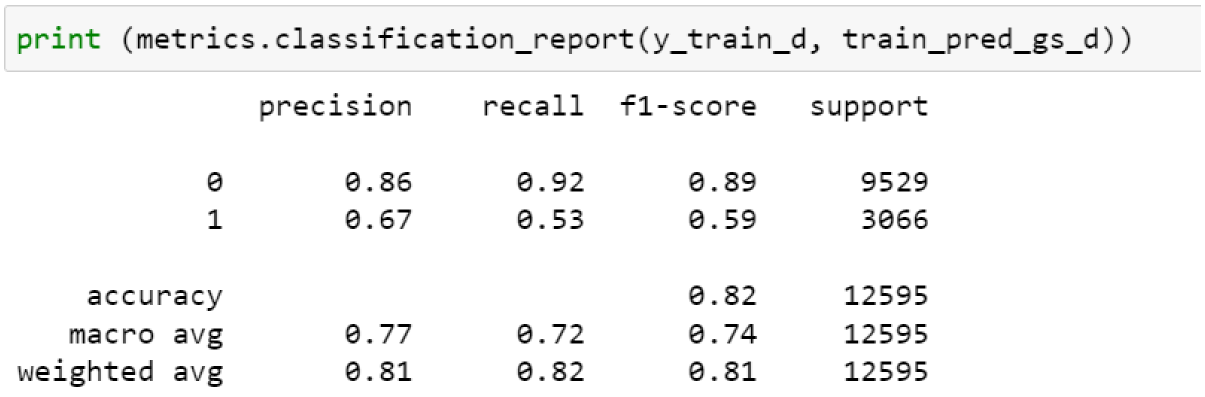
```
              precision    recall  f1-score   support

           0       0.85      0.86      0.86      9529
           1       0.55      0.54      0.55      3066

    accuracy                           0.78     12595
   macro avg       0.70      0.70      0.70     12595
weighted avg       0.78      0.78      0.78     12595
```



```
print (metrics.classification_report(y_test, test_pred_gaussian_d))
```

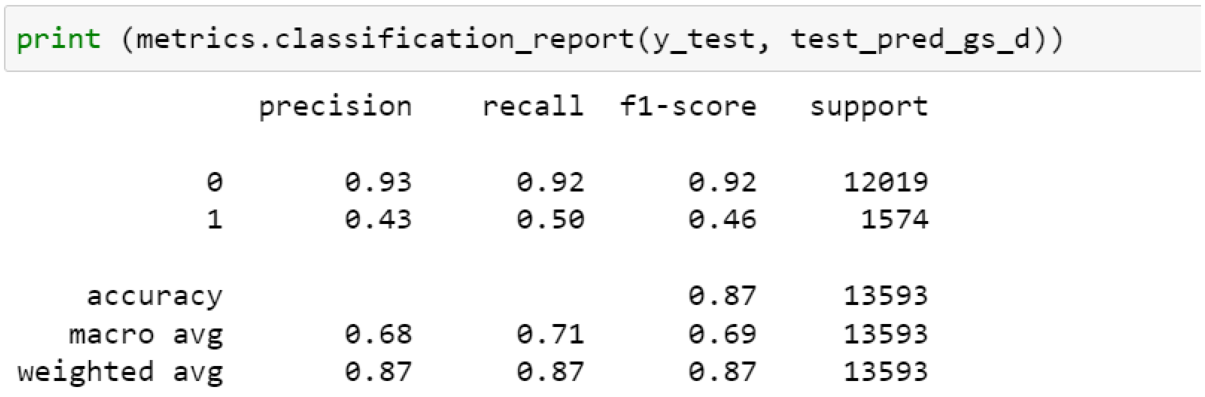
```
              precision    recall  f1-score   support

           0       0.93      0.86      0.89     12019
           1       0.33      0.52      0.40      1574

    accuracy                           0.82     13593
   macro avg       0.63      0.69      0.65     13593
weighted avg       0.86      0.82      0.84     13593
```

Multiple Ensemble – Voting – DownSampled



```
print (metrics.classification_report(y_train_d, train_pred_gs_d))
```

```
              precision    recall  f1-score   support

           0       0.86      0.92      0.89      9529
           1       0.67      0.53      0.59      3066

    accuracy                           0.82     12595
   macro avg       0.77      0.72      0.74     12595
weighted avg       0.81      0.82      0.81     12595
```

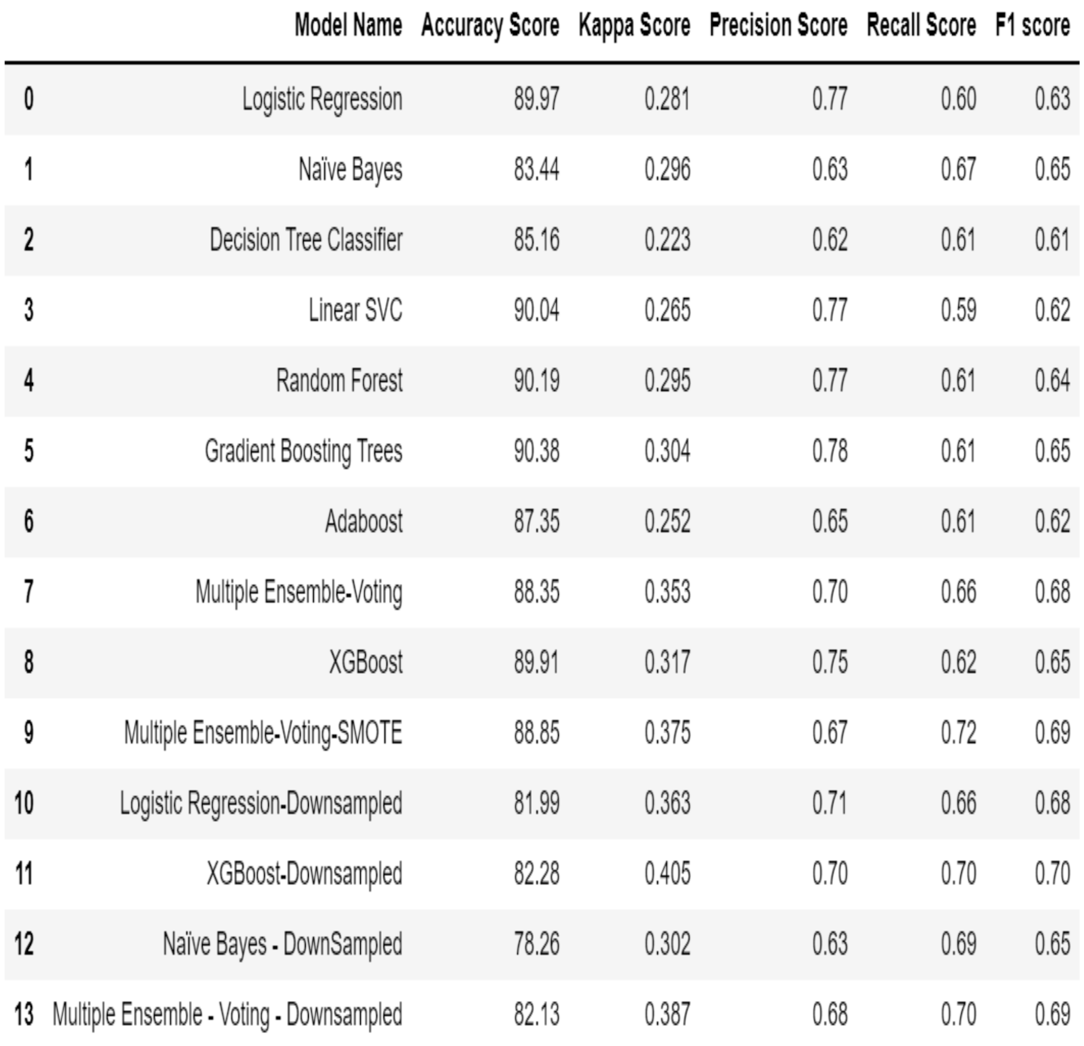


```
print (metrics.classification_report(y_test, test_pred_gs_d))
```

```
              precision    recall  f1-score   support

           0       0.93      0.92      0.92     12019
           1       0.43      0.50      0.46      1574

    accuracy                           0.87     13593
   macro avg       0.68      0.71      0.69     13593
weighted avg       0.87      0.87      0.87     13593
```

Comparison of Metrics for all Algorithms

We have tabulated all the metrics from different algorithms and a dataframe out of it.



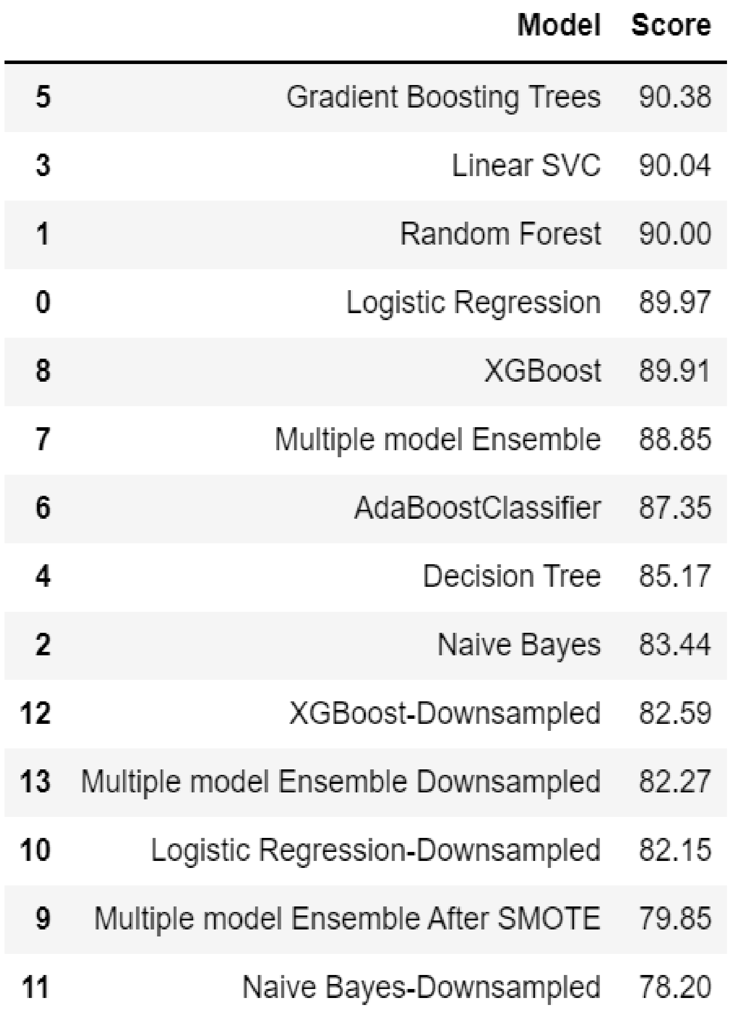
	Model Name	Accuracy Score	Kappa Score	Precision Score	Recall Score	F1 score
0	Logistic Regression	89.97	0.281	0.77	0.60	0.63
1	Naïve Bayes	83.44	0.296	0.63	0.67	0.65
2	Decision Tree Classifier	85.16	0.223	0.62	0.61	0.61
3	Linear SVC	90.04	0.265	0.77	0.59	0.62
4	Random Forest	90.19	0.295	0.77	0.61	0.64
5	Gradient Boosting Trees	90.38	0.304	0.78	0.61	0.65
6	Adaboost	87.35	0.252	0.65	0.61	0.62
7	Multiple Ensemble-Voting	88.35	0.353	0.70	0.66	0.68
8	XGBoost	89.91	0.317	0.75	0.62	0.65
9	Multiple Ensemble-Voting-SMOTE	88.85	0.375	0.67	0.72	0.69
10	Logistic Regression-Downsampled	81.99	0.363	0.71	0.66	0.68
11	XGBoost-Downsampled	82.28	0.405	0.70	0.70	0.70
12	Naïve Bayes - DownSampled	78.26	0.302	0.63	0.69	0.65
13	Multiple Ensemble - Voting - Downsampled	82.13	0.387	0.68	0.70	0.69

Score Card

We have collated all the results from the various algorithms and made two score cards

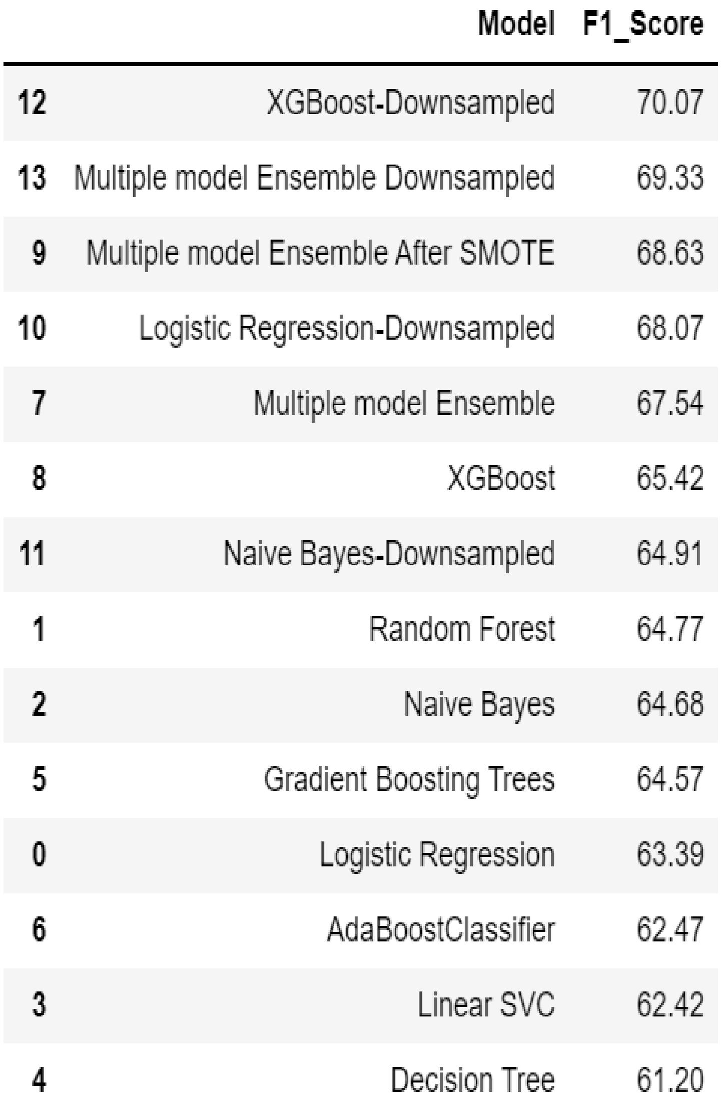
One with accuracy as the metric and the other with F1 score as the metric

Score Card of Accuracy



	Model	Score
5	Gradient Boosting Trees	90.38
3	Linear SVC	90.04
1	Random Forest	90.00
0	Logistic Regression	89.97
8	XGBoost	89.91
7	Multiple model Ensemble	88.85
6	AdaBoostClassifier	87.35
4	Decision Tree	85.17
2	Naive Bayes	83.44
12	XGBoost-Downsampled	82.59
13	Multiple model Ensemble Downsampled	82.27
10	Logistic Regression-Downsampled	82.15
9	Multiple model Ensemble After SMOTE	79.85
11	Naive Bayes-Downsampled	78.20

Score Card of F1-Score



	Model	F1_Score
12	XGBoost-Downsampled	70.07
13	Multiple model Ensemble Downsampled	69.33
9	Multiple model Ensemble After SMOTE	68.63
10	Logistic Regression-Downsampled	68.07
7	Multiple model Ensemble	67.54
8	XGBoost	65.42
11	Naive Bayes-Downsampled	64.91
1	Random Forest	64.77
2	Naive Bayes	64.68
5	Gradient Boosting Trees	64.57
0	Logistic Regression	63.39
6	AdaBoostClassifier	62.47
3	Linear SVC	62.42
4	Decision Tree	61.20

Conclusions

- It was a great learning experience working on a financial dataset.

- Our dataset consist of categorical and numerical features.

- Duration was an important parameter but it cannot be used for prediction as it is not available before making the call

- When visualized age in groups, it is found that clients with age less than 30 and greater than 60 are less contacted through the campaign but have a higher success rate.

- The biggest issue with the dataset was a imbalanced one with proportion of 89:11

- The number of datapoints is also around 48000 which made the balancing difficult

- Different machine learning models are trained and tested on the dataset.

- Different models are summarized in table above.

- If accuracy was taken as the metric, Gradient boosting trees and Random Forest are showing best performances.

- If F1 score was taken as the metric, XGBoost with Down samples train and Multiple Ensemble model after SMOTE are showing Best performances

- The Methods like SMOTE and Down sampling influenced on improving the F1 score.

- Based on the Subject Matter Expert we can decide on which metric to be used Precision or Recall and try to improve that single metric to improve the model performance in further studies

- The Trade-off between Precision and Recall is inevitable in these kind of imbalanced problems