A Project Submitted in Partial Fulfillment of the Requirements of Passing for the Course of CSI 382 - Data Mining and Knowledge Discovery

By

Angraj MD Sargia Showrav

CSE 06607820

Sajib Kotal

CSE 06607811

Mohammad Ahmedullah

CSE 06607837

Department of Computer Science and Engineering
STAMFORD UNIVERSITY BANGLADESH
January 2022

Abstract

In this project we are trying to do some predictions on the Bank Marketing Data set to predict the If a client will subscribe a term deposit or not. Also we will be experiencing relationship between the predictor variables and the response variables. We also will be trying to fit the dataset with different algorithms and finding their corresponding accuracy with the dataset.

The classification goal is to predict if the client will subscribe a term deposit-

1) yes

2) no

Table of Contents

1 Introduction

This Dataset is a big challenge for a Data Scientist. Besides being unbalanced data, further, we will realize that some datasets don't have a solution. And so, take more information and data become crucial to solve the Data Science problem. We are given the data of direct marketing campaigns (phone calls) of a Portuguese banking institution. The classification goal is to predict if the client will subscribe a term deposit (target variable y). This case study is inspired by this research paper where the researchers have used a very similar dataset as the one we will be using throughout this case study for determining the success of Bank Telemarketing

2.1 About the Dataset:

The dataset consists of direct marketing campaigns data of a banking institution. The dataset was picked from UCI Machine Learning Repository . There were four variants of the datasets out of which we chose “ bank-additional-full.csv” which consists of 41188 data points with 20 independent variables out of which 10 are numeric features and 10 are categorical features. The list of features available to us are given below:

2.2 Dataset Attributes:

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)

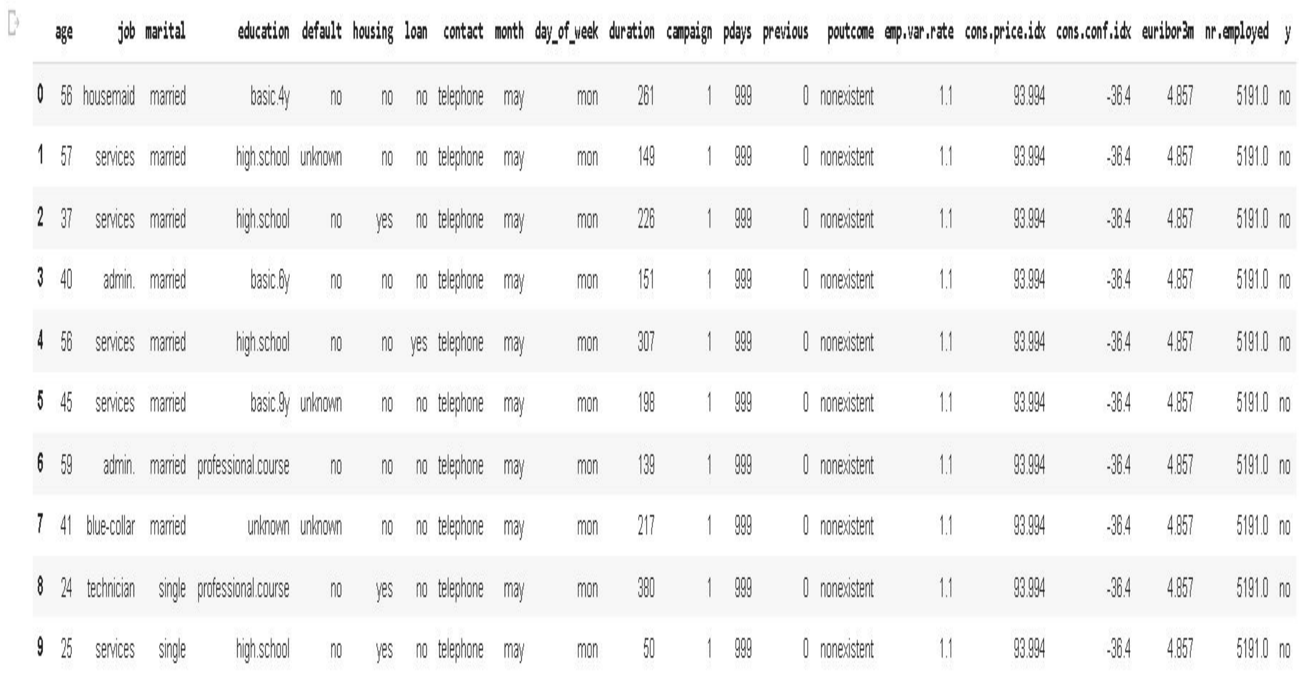
18. cons.conf.idx: consumer confidence index — monthly indicator (numeric)

19. euribor3m: euribor 3 month rate — daily indicator (numeric)

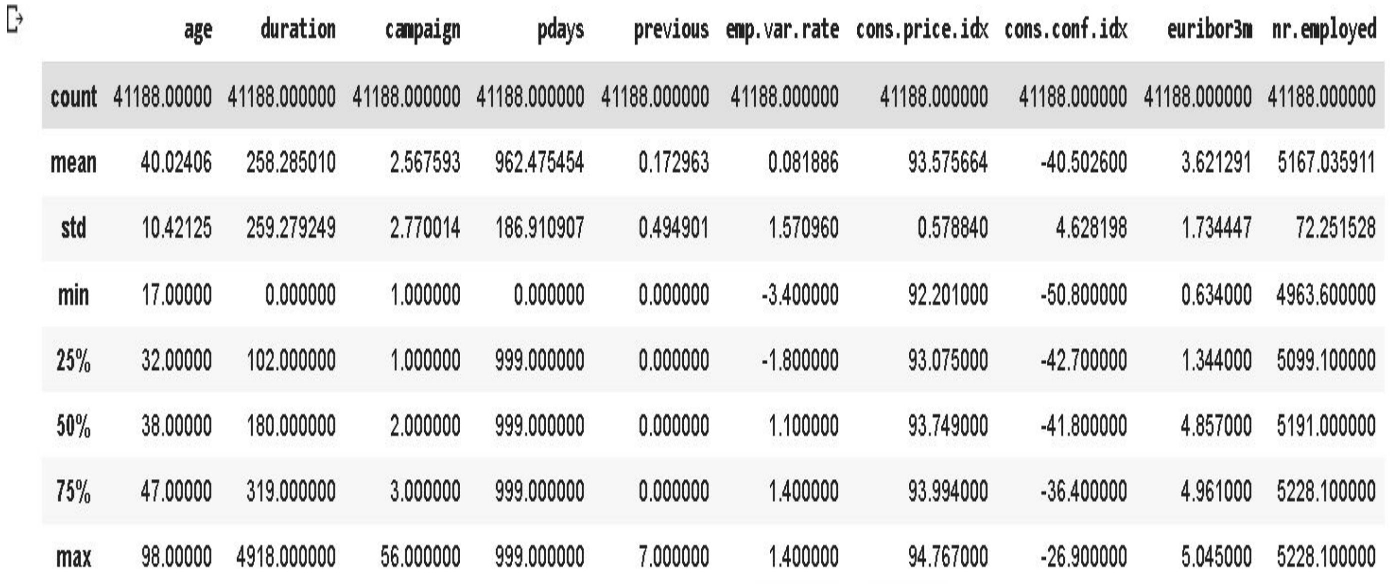
20. nr.employed: number of employees — quarterly indicator (numeric)

2.3 Data set Information

Head(10)



	age	job	marital	education	default	housing	loan	contact	month	day_of_week	duration	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	261	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	149	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
2	37	services	married	high.school	no	yes	no	telephone	may	mon	226	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
3	40	admin.	married	basic.6y	no	no	no	telephone	may	mon	151	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
4	56	services	married	high.school	no	no	yes	telephone	may	mon	307	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
5	45	services	married	basic.9y	unknown	no	no	telephone	may	mon	198	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
6	59	admin.	married	professional.course	no	no	no	telephone	may	mon	139	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
7	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	217	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
8	24	technician	single	professional.course	no	yes	no	telephone	may	mon	380	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
9	25	services	single	high.school	no	yes	no	telephone	may	mon	50	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no



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

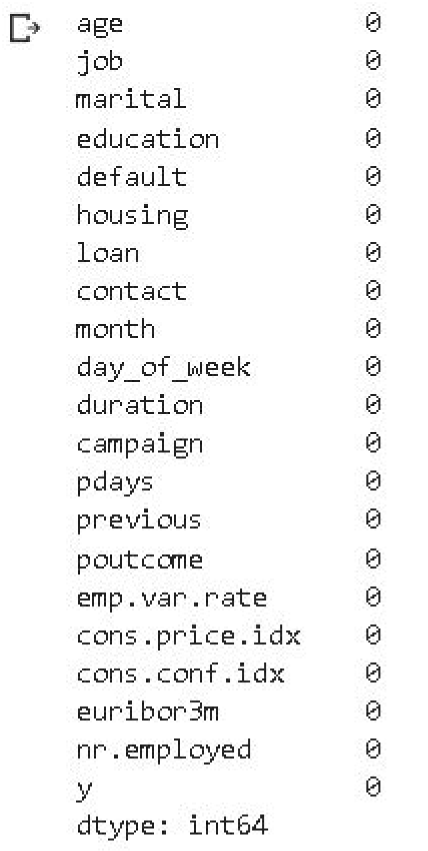
3. Data pre processing

Data preprocessing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process. The phrase "garbage in, garbage out" is particularly appli- cable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out-of-range values (e.g., Income: -100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing val- ues, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis.

3.1 HANDLING MISSING DATA

Missing data is a problem that continues to plague data analysis methods. Even as our analysis methods gain sophistication, we continue to encounter missing values in fields, especially in databases with a large number of fields. The ab- sence of information is rarely beneficial. All things being equal, more data is almost always better. Therefore, we should think carefully about how we handle the thorny issue of missing data

```
df.isnull().sum()
```



```
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
dtype: int64
```

Fortunately Our data set contains no missing or NULL values. So we can proceed as the dataset is given.

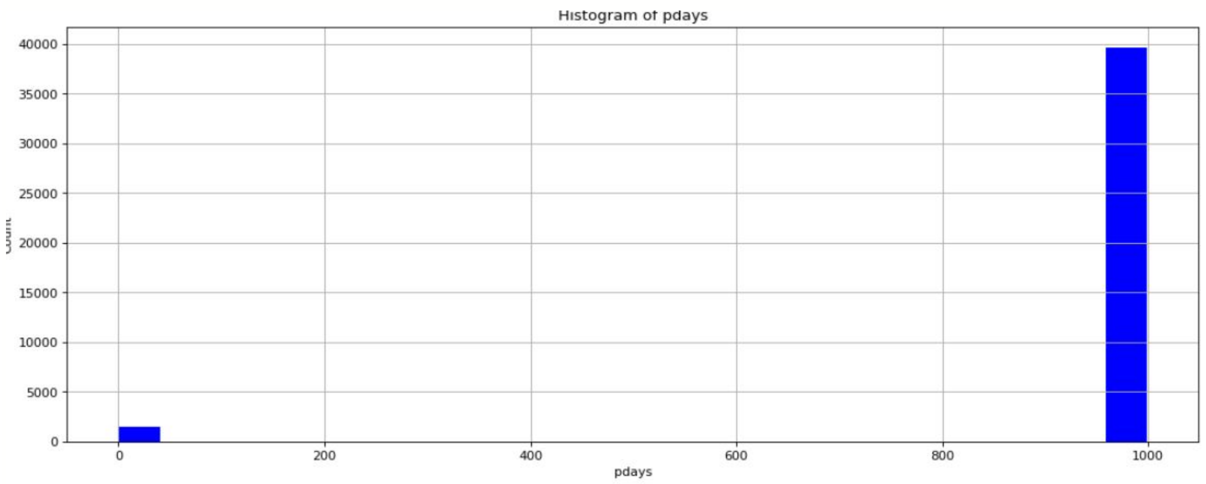
3.2 Finding outliers

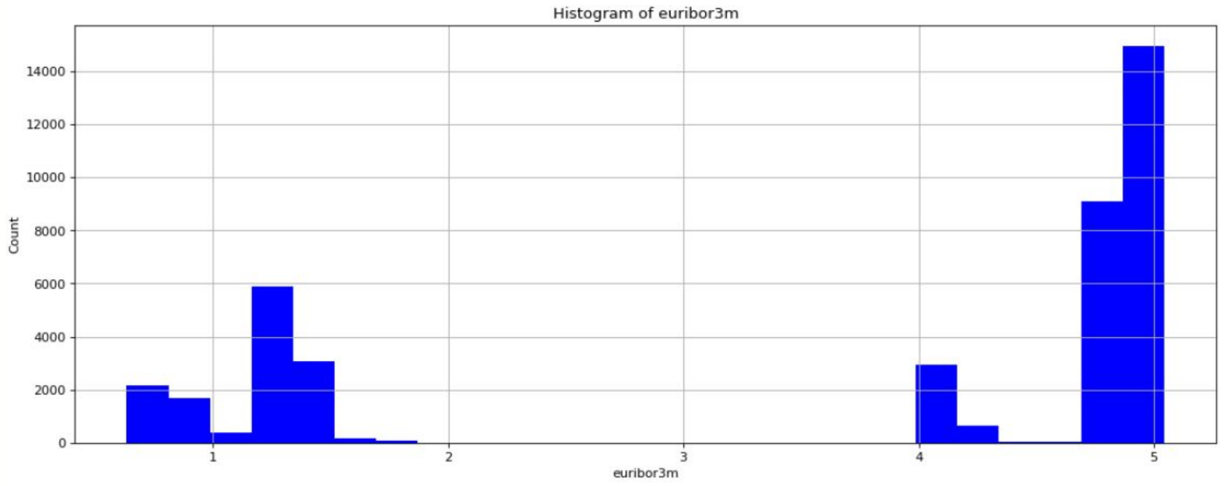
Outliers are data points that are far from other data points . In other words , they are unusual values in a dataset . Outliers are problematic for many stastical analysis because they can cause tests to either miss significant findings or distort real values.

3.2.1 Histograms

One graphical method for identifying outliers for numeric variables is to examine a histogram of the variable. Figure below shows a histogram generated of the vehicle weights from the cars data set. There appears to be one lonely vehicle in th e extreme left tail of the distribution, with a vehicle weight in the hundreds of poun ds rather than in the thousands.

We have tried with every attributes and these attributes shows some distortion in histogram

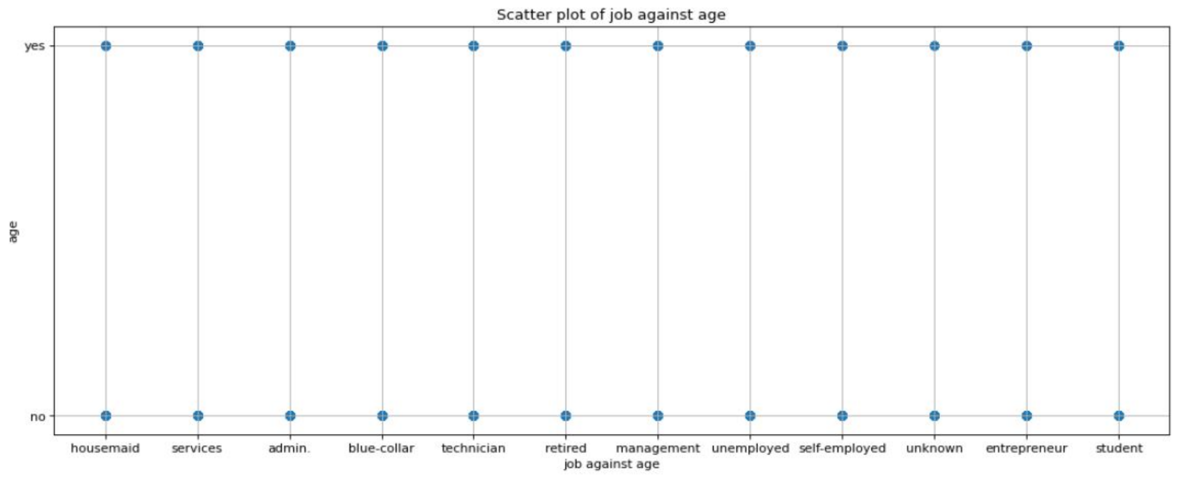


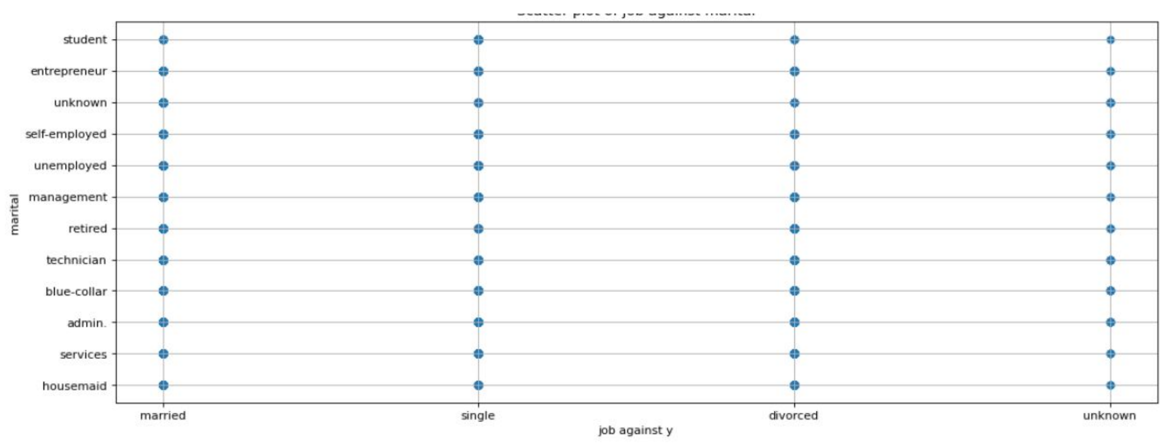


3.2.2 Scatterplots

Sometimes two-dimensional scatter plots can help to reveal outliers in more than one variable.

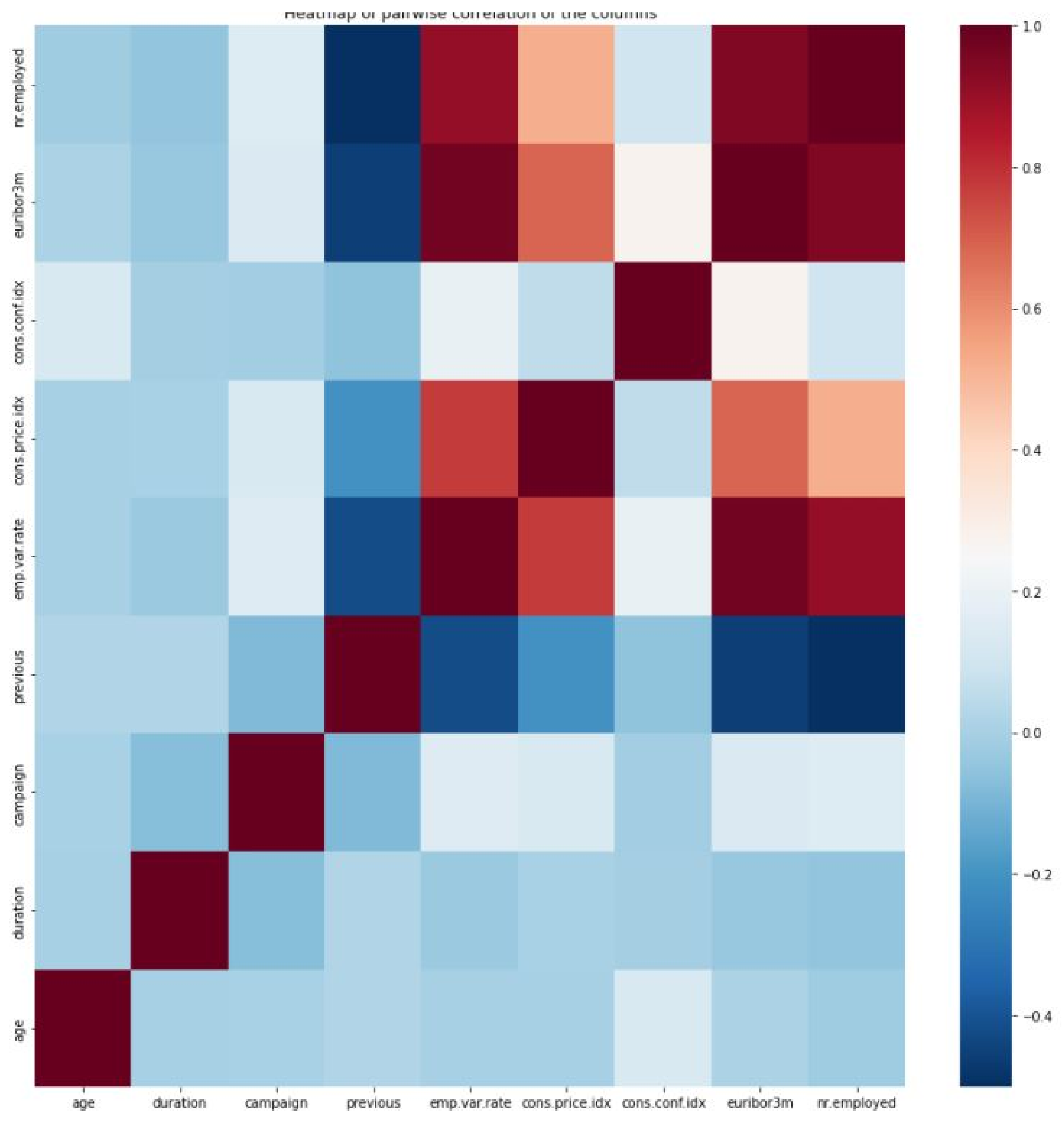
As we have mentioned the dataset is highly imbalanced the data shows disimilar scatterplots that are unable to bear any meanings





3.2.3 NUMERICAL METHODS FOR IDENTIFYING OUTLIERS

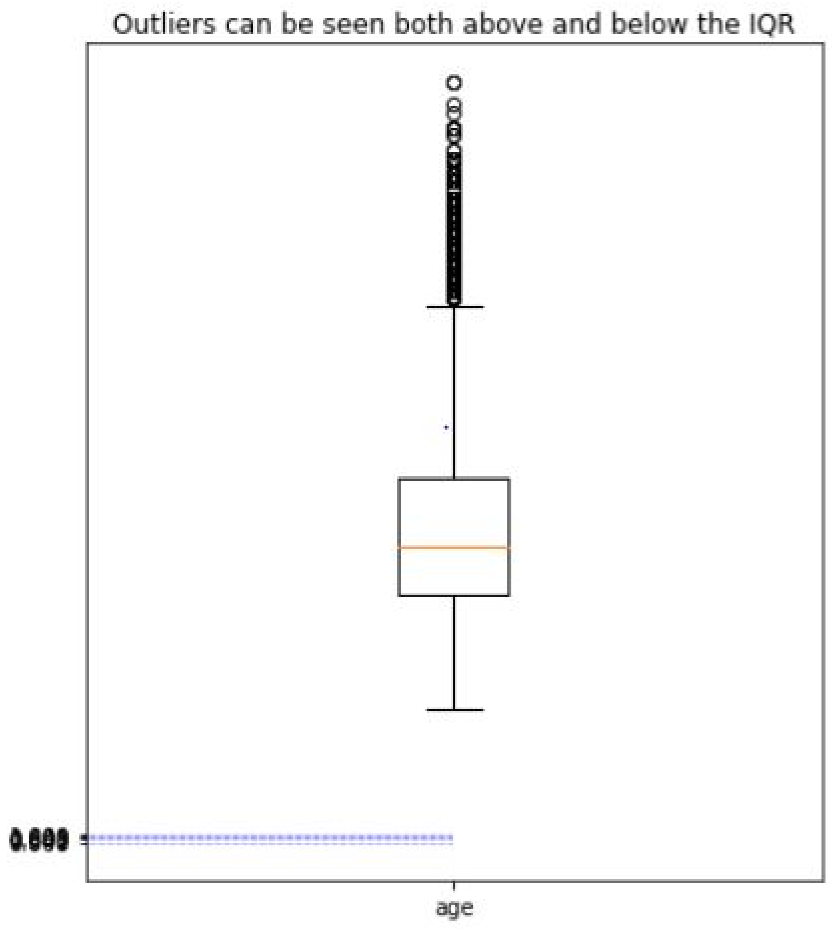
One method of using statistics to identify outliers is to use Z-score standardization. Often, an outlier can be identified because it is much farther than 3 standard deviations from the mean and therefore has a Z-score standardization that is either less than -3 or greater than 3. Field values with Z-scores much beyond this range probably bear further investigation to verify that they do not represent data entry errors or other issues.



The emp.var.rate, cons.price.idx, euribor3m and nr.employed features have very high correlation. With euribor3m and nr.employed having the highest correlation

4.2 Exploring categorical variables

One of the primary reasons for performing exploratory data analysis is to investigate the variables, look at histograms of the numeric variables, examine the distributions of the categorical variables, and explore the relationships among sets of variables



This shows some outliers in age attributes.

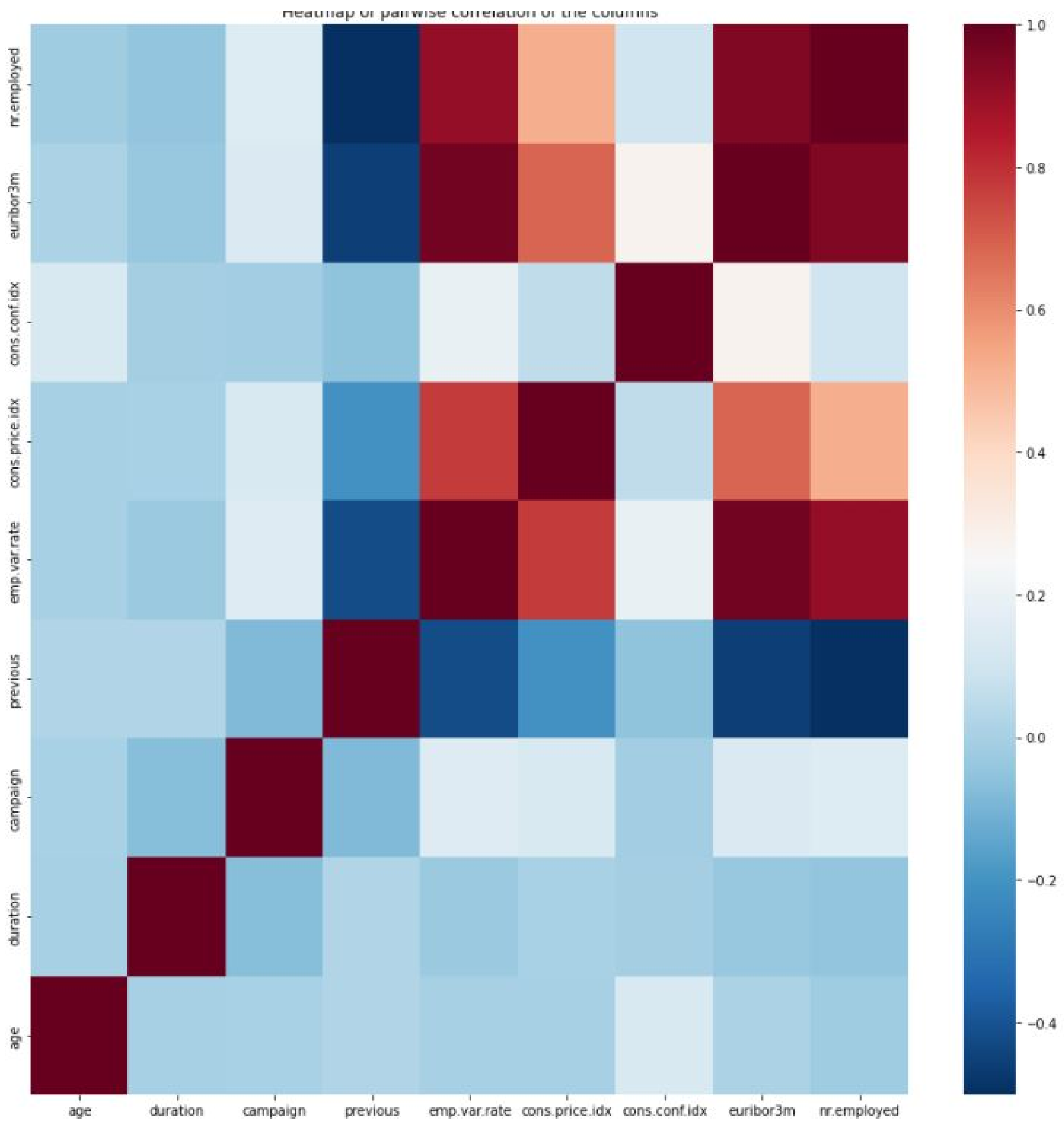
4 Exploraory data analysis

Tukey defined data analysis in 1961 as: "Procedures for analyzing data, techniques for interpreting the results of such procedures, ways of planning the gathering of data to make its analysis easy, i.e., more precise or more accurate, and all the machinery and results of (mathematical) statistics which apply to analyzing data."

A statistical model can be used or not, but primarily EDA is for seeing what the data can tell us beyond the formal modeling or hypothesis testing task.

4.1 Correlated Variables

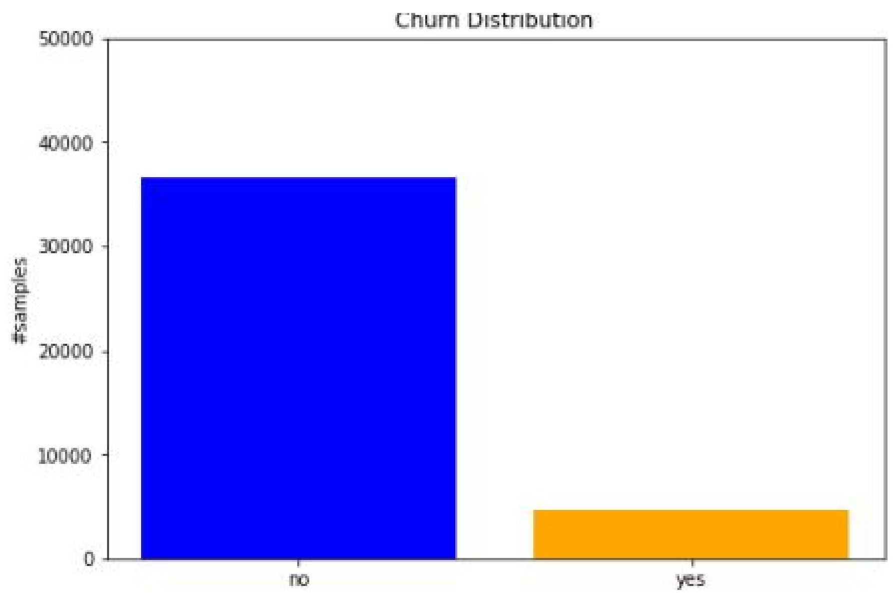
One should take care to avoid feeding correlated variables to one's data mining and statistical models. At best, using correlated variables will overemphasize one data component; at worst, using correlated variables will cause the model to become unstable and deliver unreliable results.



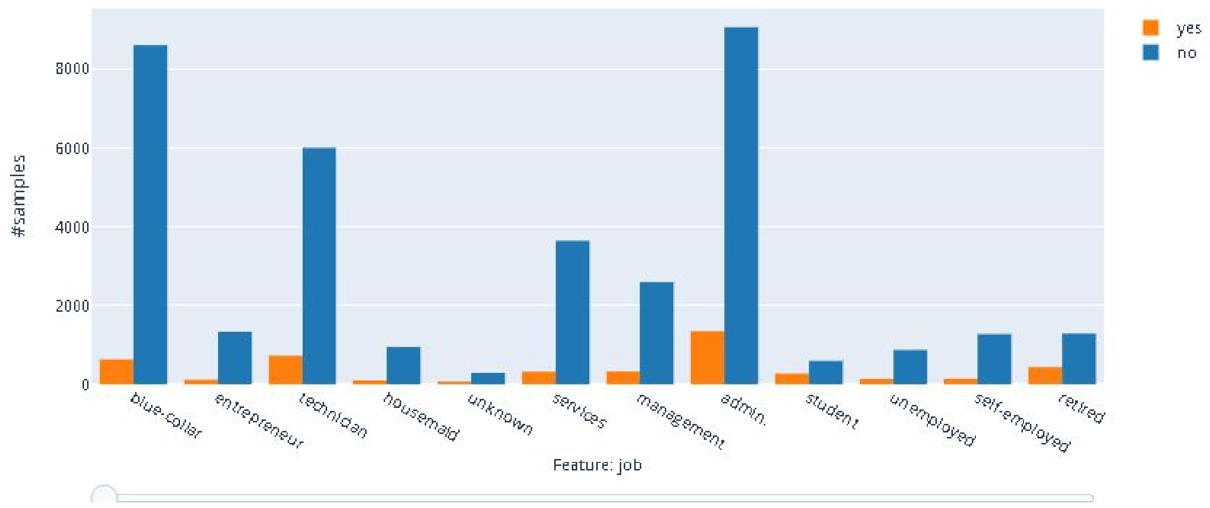
The emp.var.rate, cons.price.idx, euribor3m and nr.employed features have very high correlation. With euribor3m and nr.employed having the highest correlation

4.2 Exploring categorical variables

One of the primary reasons for performing exploratory data analysis is to investigate the variables, look at histograms of the numeric variables, examine the distributions of the categorical variables, and explore the relationships among sets of variables.

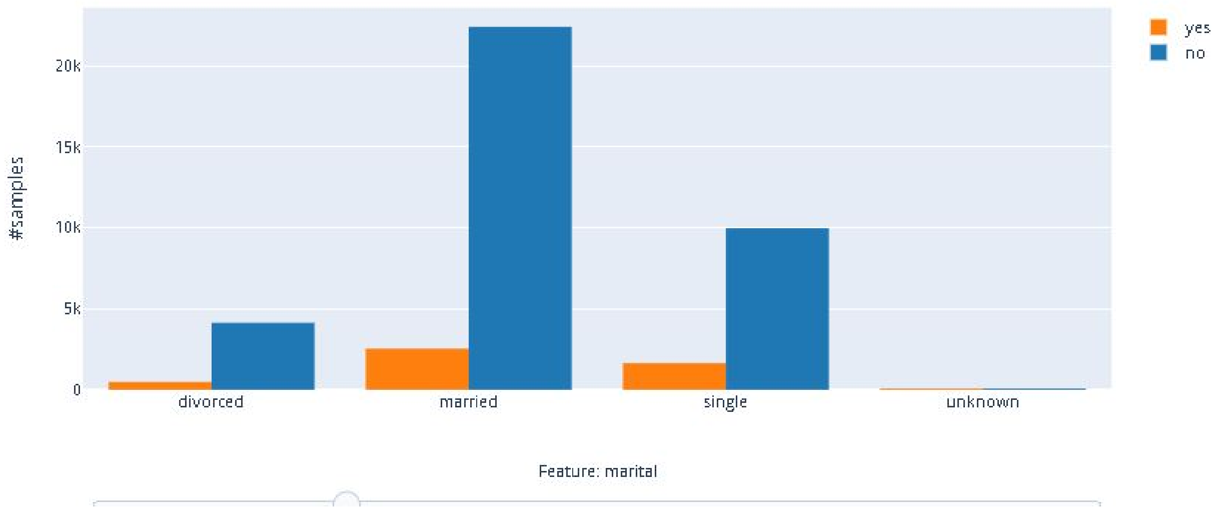


We can see from the above plot that the dataset in imbalanced, where the number of negative class is close to 8 times the number of positive class.

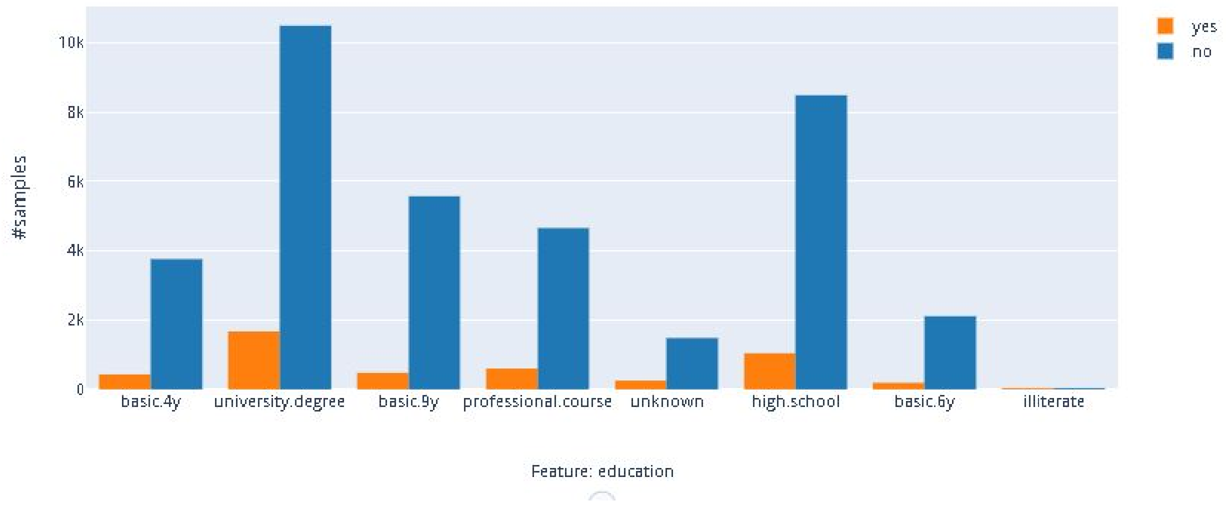


.

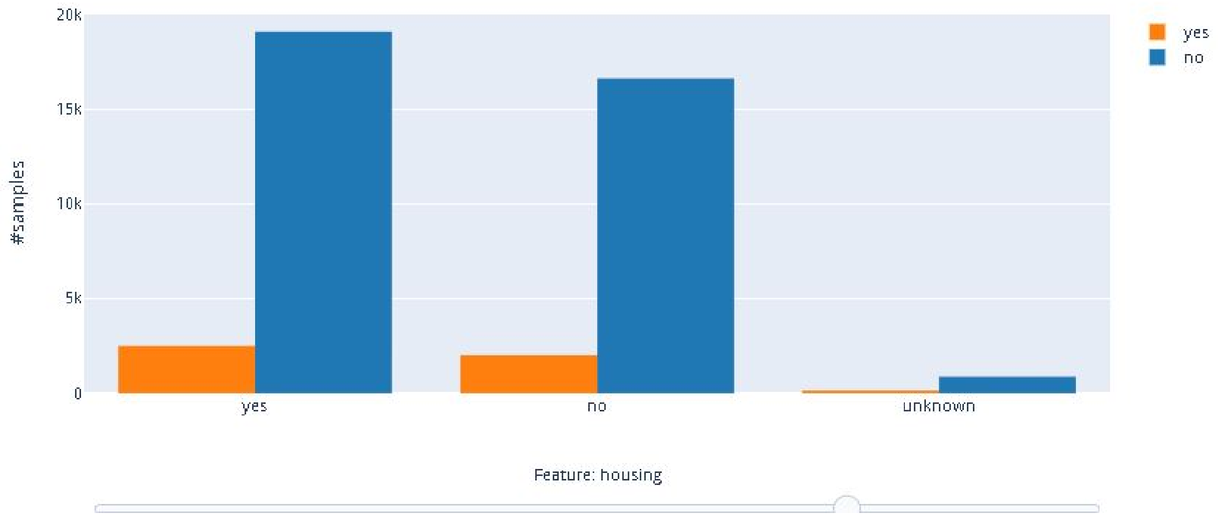
From the above plot, we can see that the customers who have a job of admin have the highest rate of subscribing a term deposit, but they are also the highest when it comes to not subscribing. This is simply because we have more customers working as admin than any other profession.



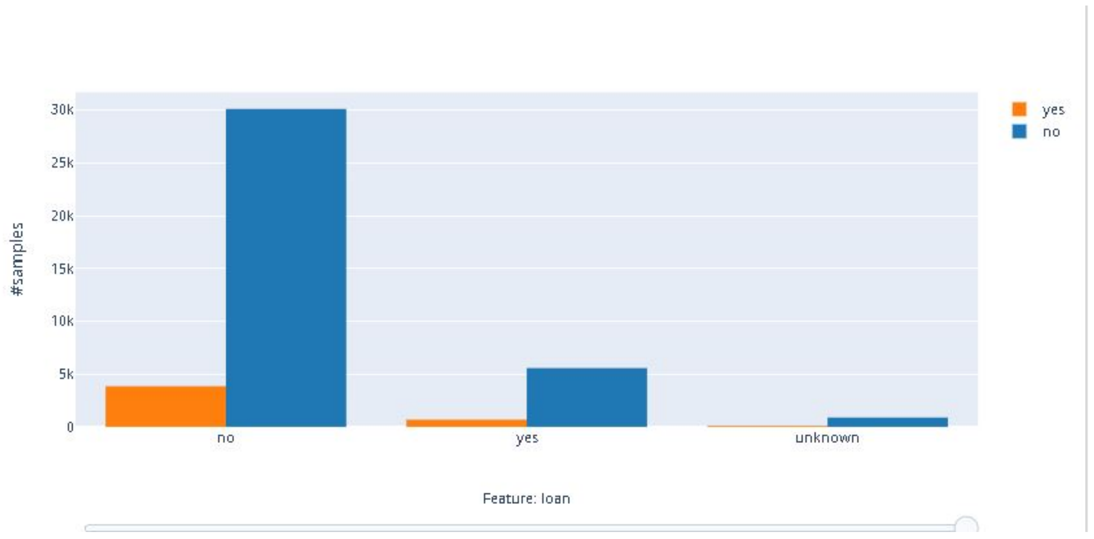
Majority of the customers are married. Followed by Single, divorced and unknown.



we can see that majority of the unversity degree holder clients have a subscription also they have the negative percentage also.



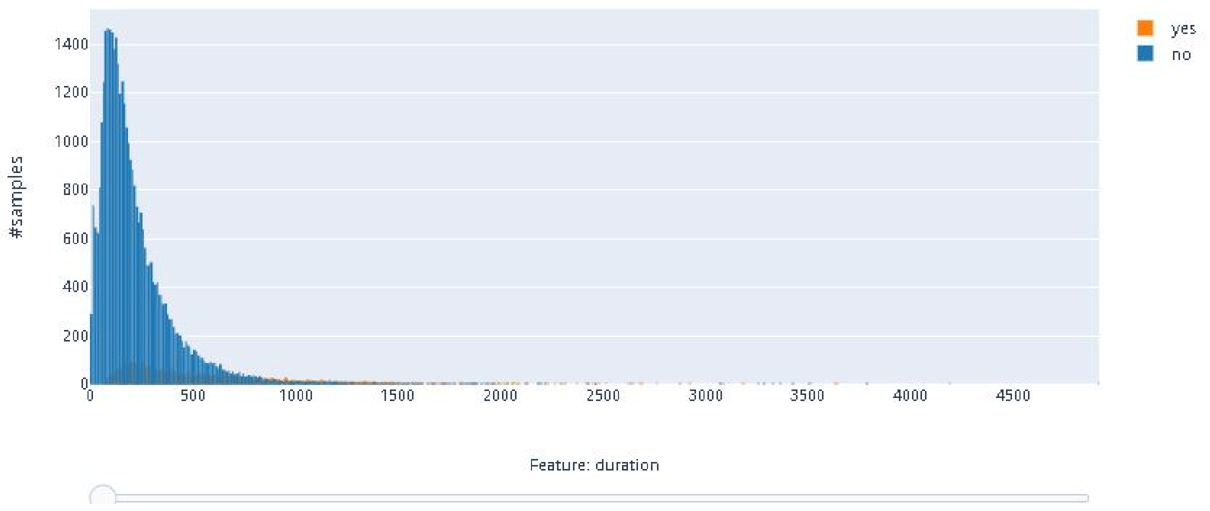
As we can see from the above plot, majority of the customers have a housing loan.



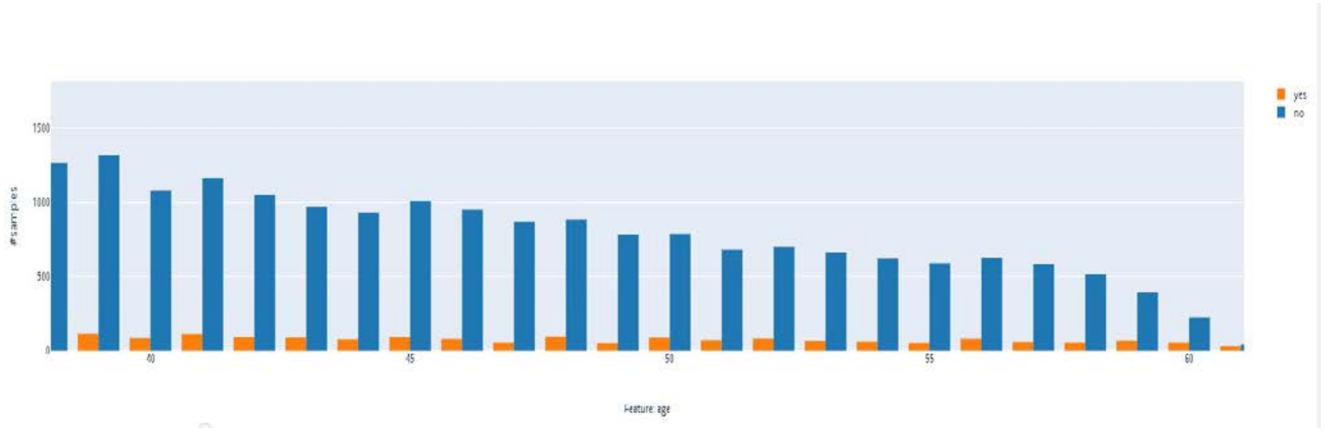
As we can see majority of the clients dont have a housing loan

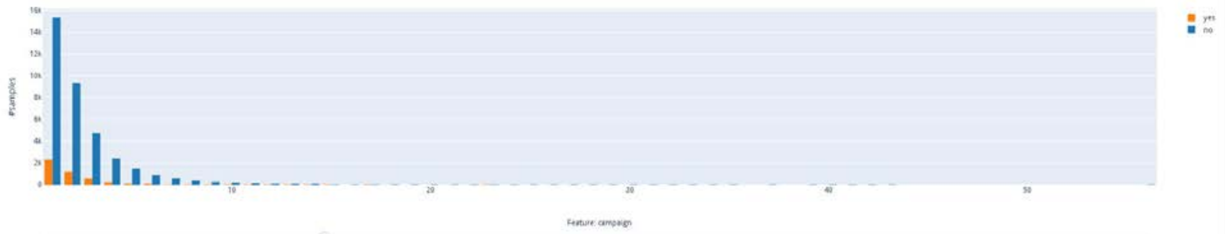
4.3 EXPLORING NUMERICAL VARIABLES

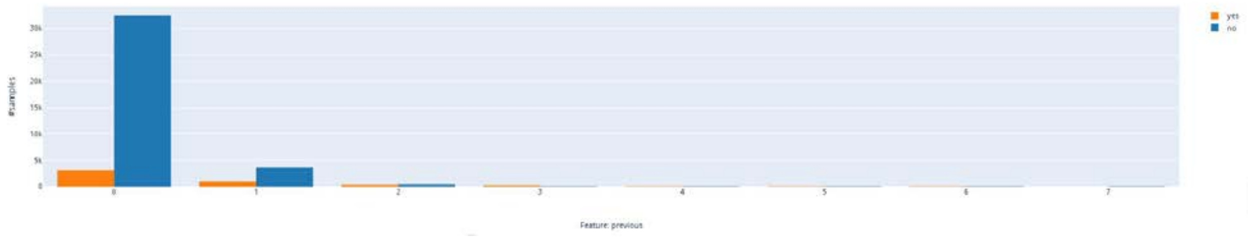
Next, we turn to an exploration of the numerical predictive variables. We begin with numerical summary measures, including minimum and maximum; mea- sures of center, such as mean, median, and mode; and measures of variability, such as standard deviation.

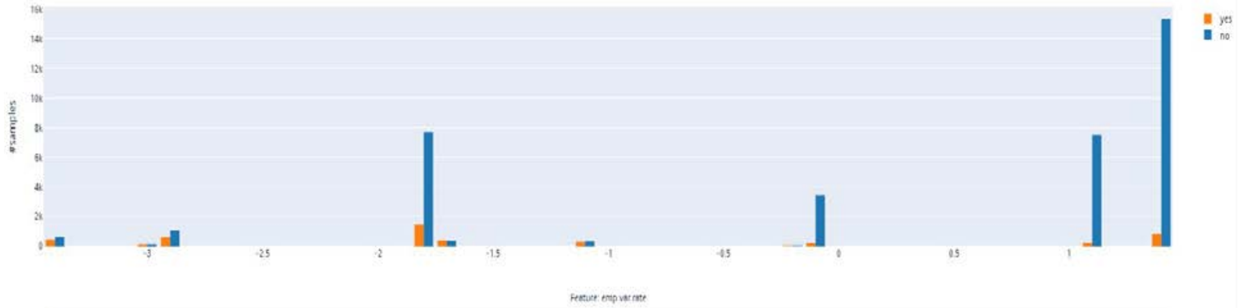


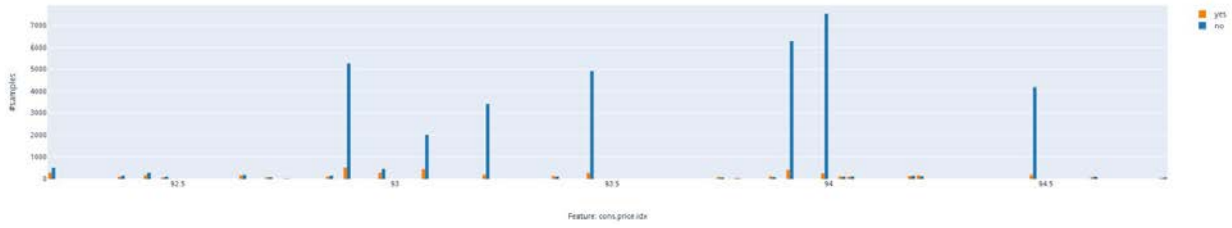
From the above plot it is clear that, the duration (last contact duration) of a customer can be useful for predicting the target variable. It is expected because it is already mentioned in the data overview that this field higheIy affects the target variable and should only be used for benchmark purpose.

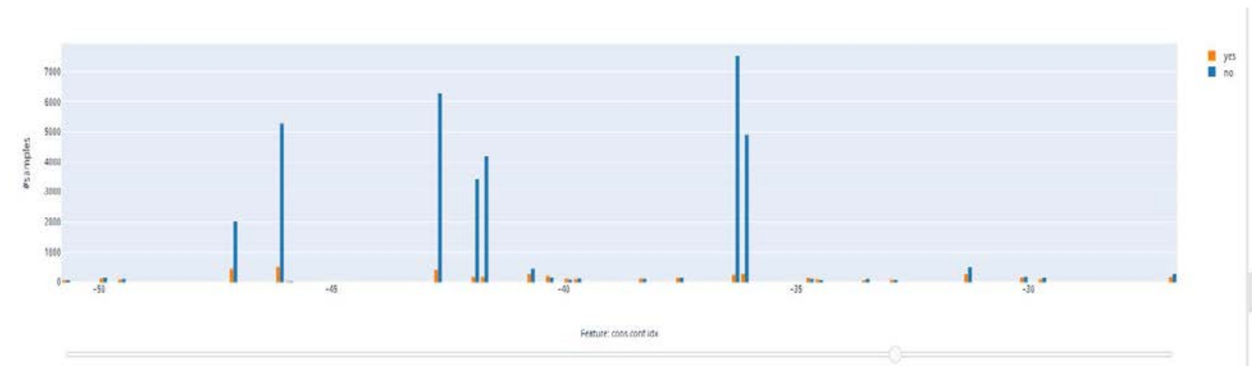



#samples
Feature: campaign
yes
no

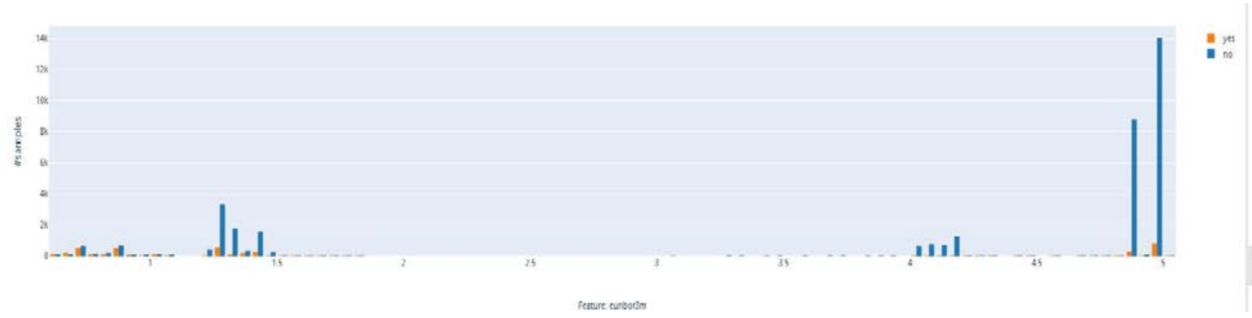

#samples
Feature: previous
yes
no


#samples
Feature: emp.var.rate
yes
no

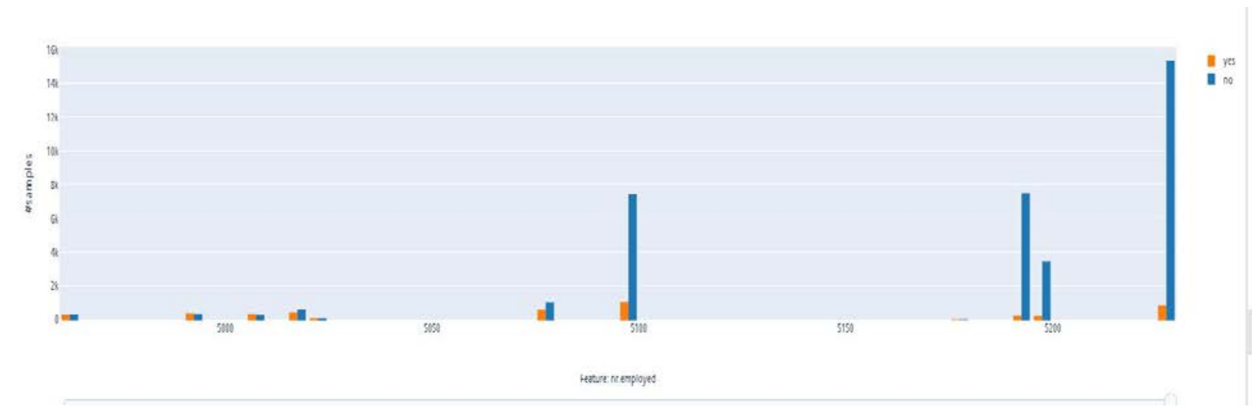

#samples
Feature: cons.price.idx
yes
no



#samples
7000
6000
5000
4000
3000
2000
1000
0
−50
−45
−40
−35
−30
Feature: cons.conf.idx
yes
no

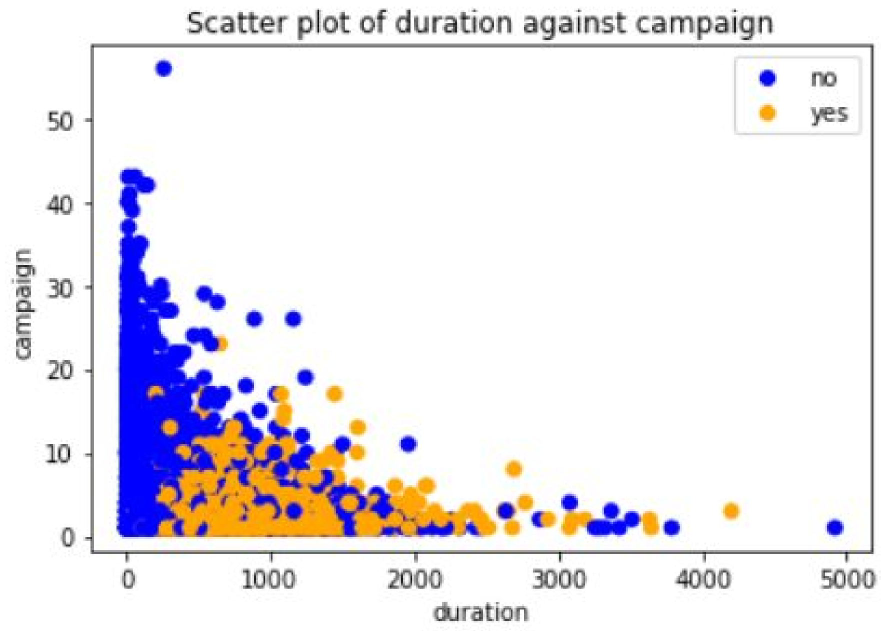


#samples
14k
12k
10k
8k
6k
4k
2k
0
1
1.5
2
2.5
3
3.5
4
4.5
5
Feature: euribor3m
yes
no

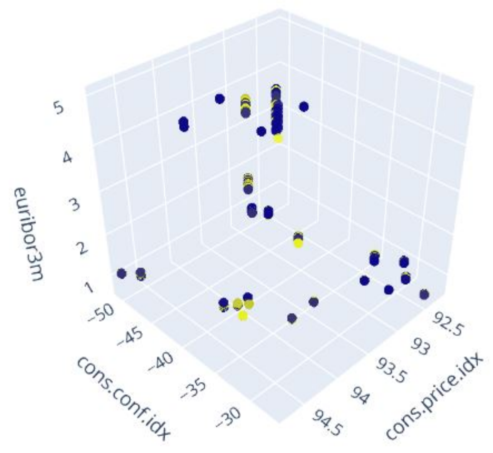


#samples
16k
14k
12k
10k
8k
6k
4k
2k
0
5000
5050
5100
5150
5200
Feature: nr.employed
yes
no

4.4 EXPLORING MULTIVARIATE RELATIONSHIPS



here is a 3d scatterplot between 3 variables



5. **Statistical Approaches to Estimation and** Prediction

If estimation and prediction are considered to be data mining tasks, statistical analysts have been performing data mining for over a century. In this lab we will be examining :

- univariate methods
- statistical estimation
- prediction methods

These methods include point estimation and confidence interval estimation. Next we will consider simple linear regression, where the relationship between two numerical variables is investigated. Finally, we will examine multiple regression, where the relationship between a response variable and a set of predictor variables is modeled linearly.

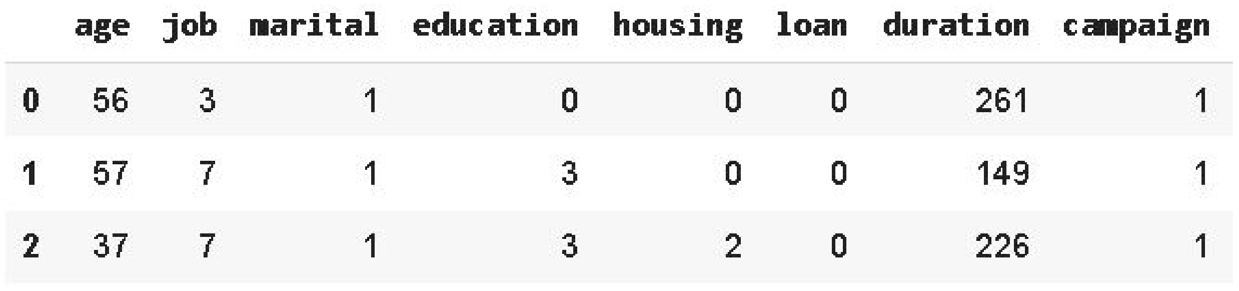
As we have measured some of the attributes ar lesser concern we will be eliminating them as we have a huge dataset to measure. We have to do some pre-processing to those data.

For stastical approach we will be eliminating 5 columngs that are 'contact','month','day_of_week','default','pdays'.

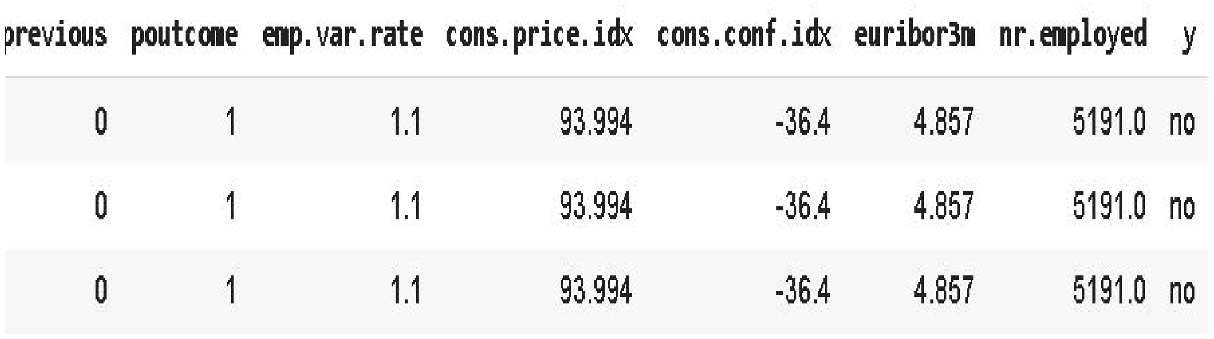
As for stastical approach and our upcoming measurement will be not using these attributes.

Also we will be turning categorical variables to numeric to apply in our measure.

So after the preprocessing the dataset head will be looking like this



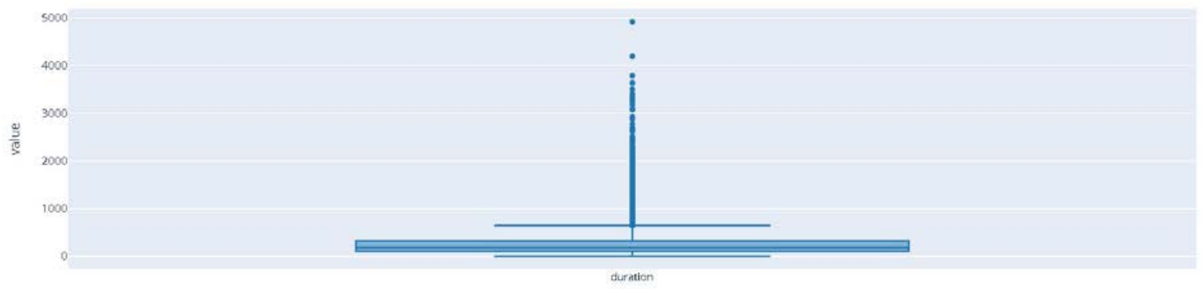
	age	job	marital	education	housing	loan	duration	campaign
0	56	3	1	0	0	0	261	1
1	57	7	1	3	0	0	149	1
2	37	7	1	3	2	0	226	1



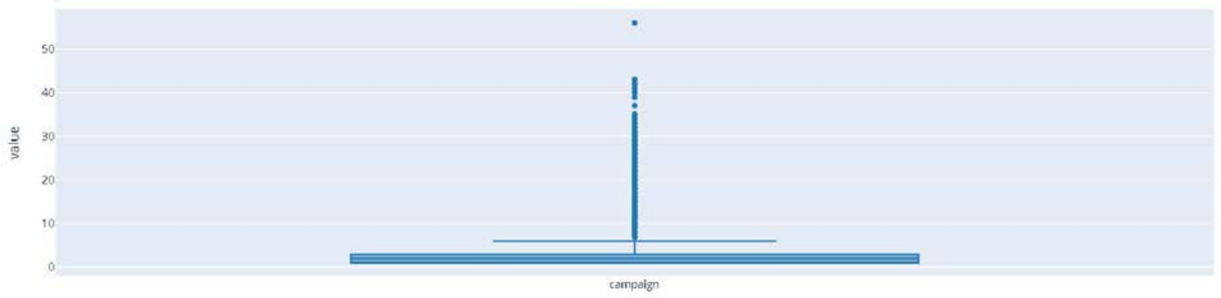
previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	1	1.1	93.994	-36.4	4.857	5191.0	no
0	1	1.1	93.994	-36.4	4.857	5191.0	no
0	1	1.1	93.994	-36.4	4.857	5191.0	no

5.1 UNIVARIATE METHODS: MEASURES OF CENTER AND SPREAD

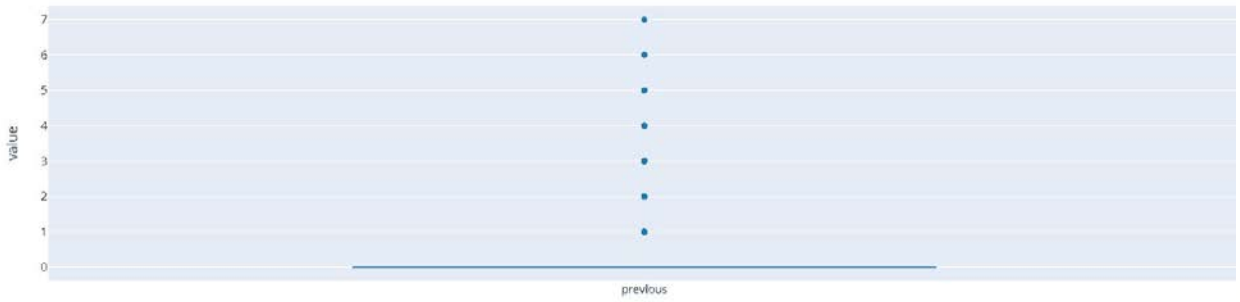
Center describes a typical value of a data point. Two measures of center are **mean and median**. Spread describes the variation of the data. Two measures of spread are range and standard deviation.

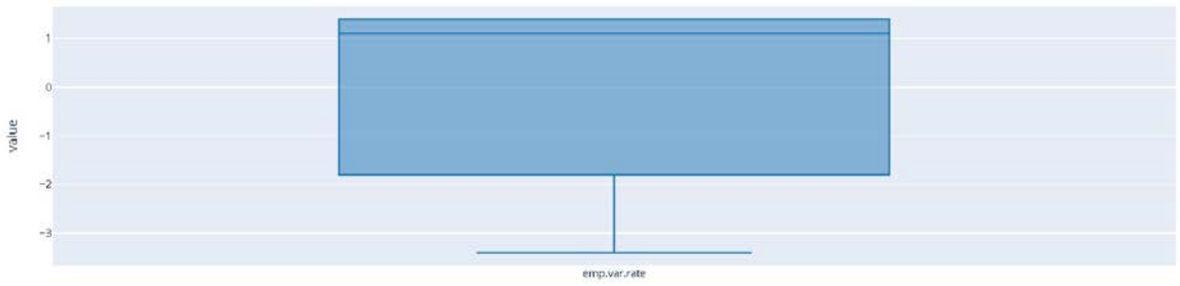


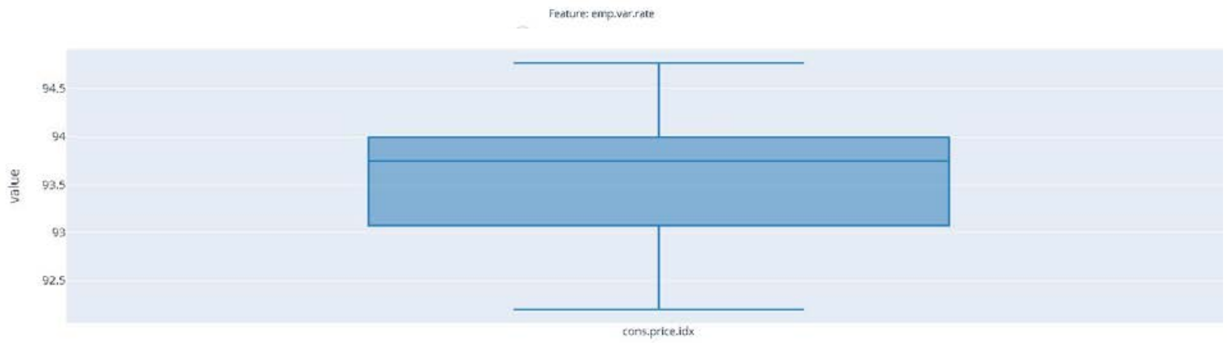
Feature: duration

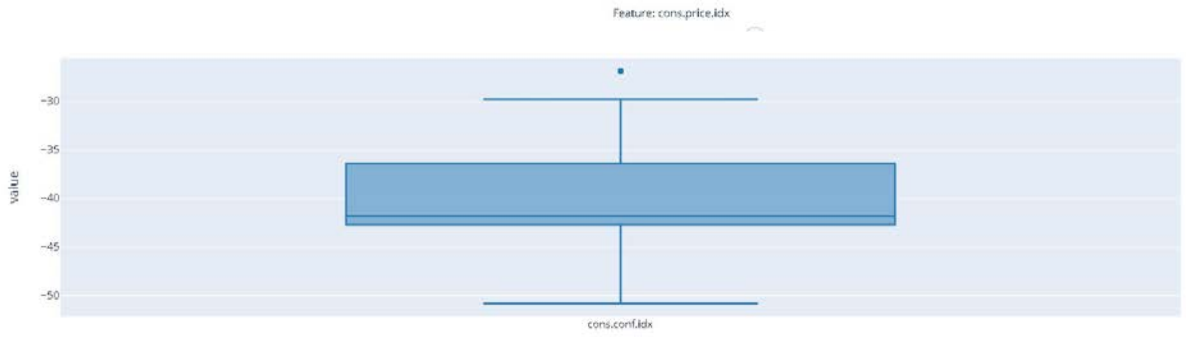


Feature: campaign









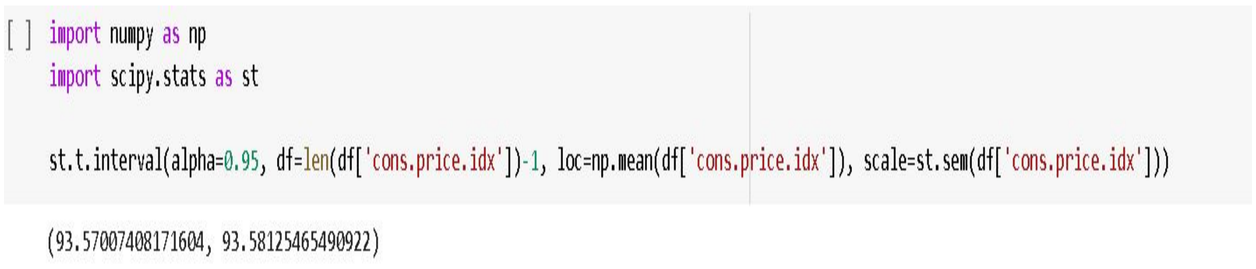
Here we can see there are lot more variations in the duration, campaign and previous attribute

5.2 Estimation

Confidence Interval Estimate

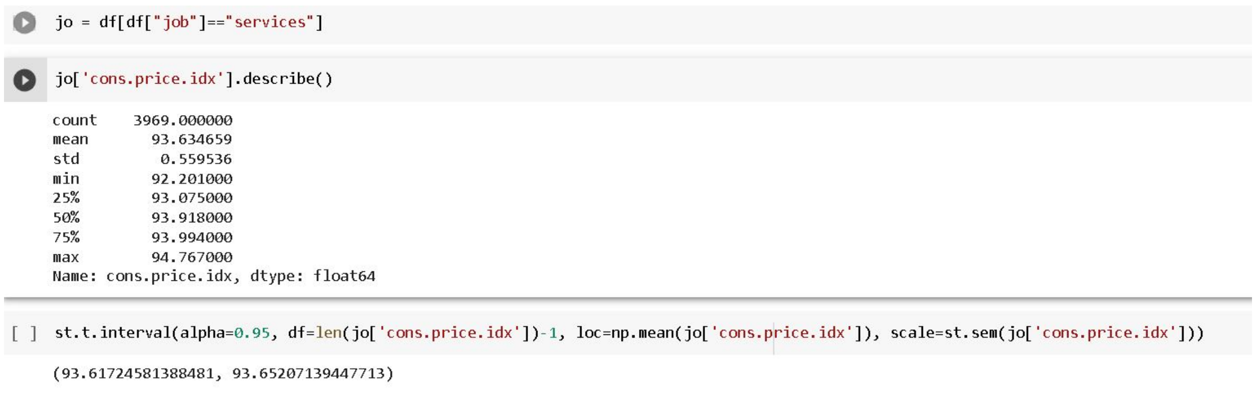
When we run studies we want to be confident in the results from our sample. Confidence intervals show us the likely range of values of our population mean. When we calculate the mean we just have one estimate of our metric; confidence intervals give us richer data and show the likely values of the true population mean.

Here we r calculating cons.idx.price with a confidence of 95%. A 95% confidence interval means that if we were to take 100 different samples and compute a 95% confidence interval for each sample, then approximately 95 of the 100 confidence intervals will contain the true mean value (**μ**)



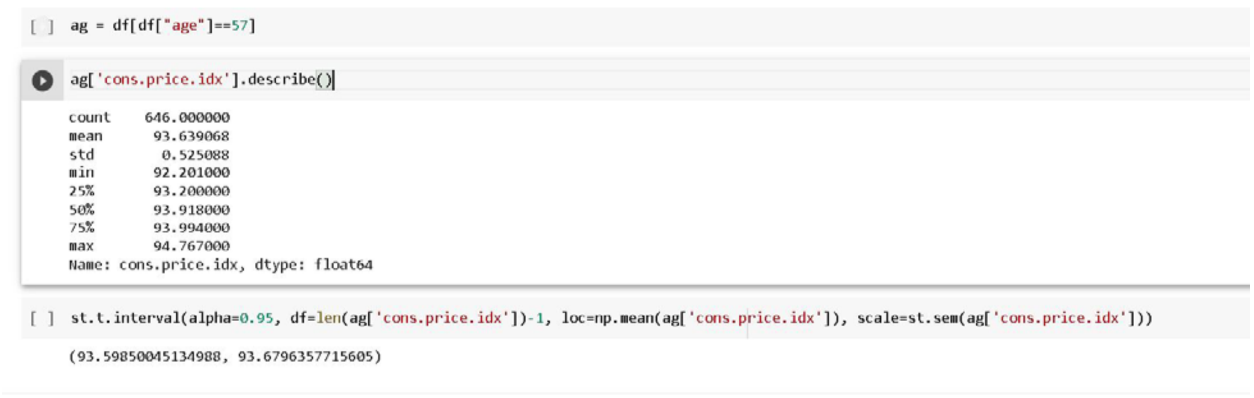
```
[ ] import numpy as np
    import scipy.stats as st

    st.t.interval(alpha=0.95, df=len(df['cons.price.idx'])-1, loc=np.mean(df['cons.price.idx']), scale=st.sem(df['cons.price.idx']))

    (93.57007408171604, 93.58125465490922)
```

we are 95% confident that the population mean number of cons.price.idx for all data falls between 93.93.57007408171604 and 93.58125465490922



```
jo = df[df["job"]=="services"]

jo['cons.price.idx'].describe()

count    3969.000000
mean       93.634659
std         0.559536
min        92.201000
25%        93.075000
50%        93.918000
75%        93.994000
max        94.767000
Name: cons.price.idx, dtype: float64

[ ] st.t.interval(alpha=0.95, df=len(jo['cons.price.idx'])-1, loc=np.mean(jo['cons.price.idx']), scale=st.sem(jo['cons.price.idx']))

(93.61724581388481, 93.65207139447713)
```

We are 95% confident that the mean number of cons.idx.price for all services job falls between 93.61724581388481 and 93.65207139447713



```
[ ]  ag = df[df["age"]==57]

     ag['cons.price.idx'].describe()

     count    646.000000
     mean      93.639068
     std        0.525088
     min       92.201000
     25%       93.200000
     50%       93.918000
     75%       93.994000
     max       94.767000
     Name: cons.price.idx, dtype: float64

[ ]  st.t.interval(alpha=0.95, df=len(ag['cons.price.idx'])-1, loc=np.mean(ag['cons.price.idx']), scale=st.sem(ag['cons.price.idx']))

     (93.59850045134988, 93.6796357715605)
```

we are 95 % sure that the mean number of 57 age falls between 93.59850045134988 and 93.6796357715605

5.3 BIVARIATE METHODS

Regression - Calculation

To calculate b_0 and b_1 we need to calculate the $\bar{x}$, $\bar{y}$, $\sum x^2$, $\sum xy$, σ and co-variance of x and y. In the given scenario, $\bar{x}=6.92$, $\bar{y}=42.67$, $\sigma=4.44$, $\sum x^2=5191$, $\sum xy=19135.91$.

Thus we can calculate the variance of x as:

$\sigma(x)=\sum x^2-n\times\bar{x}=(\sigma)^2=19.76$

Now we can find the co-variance of x and y by using:

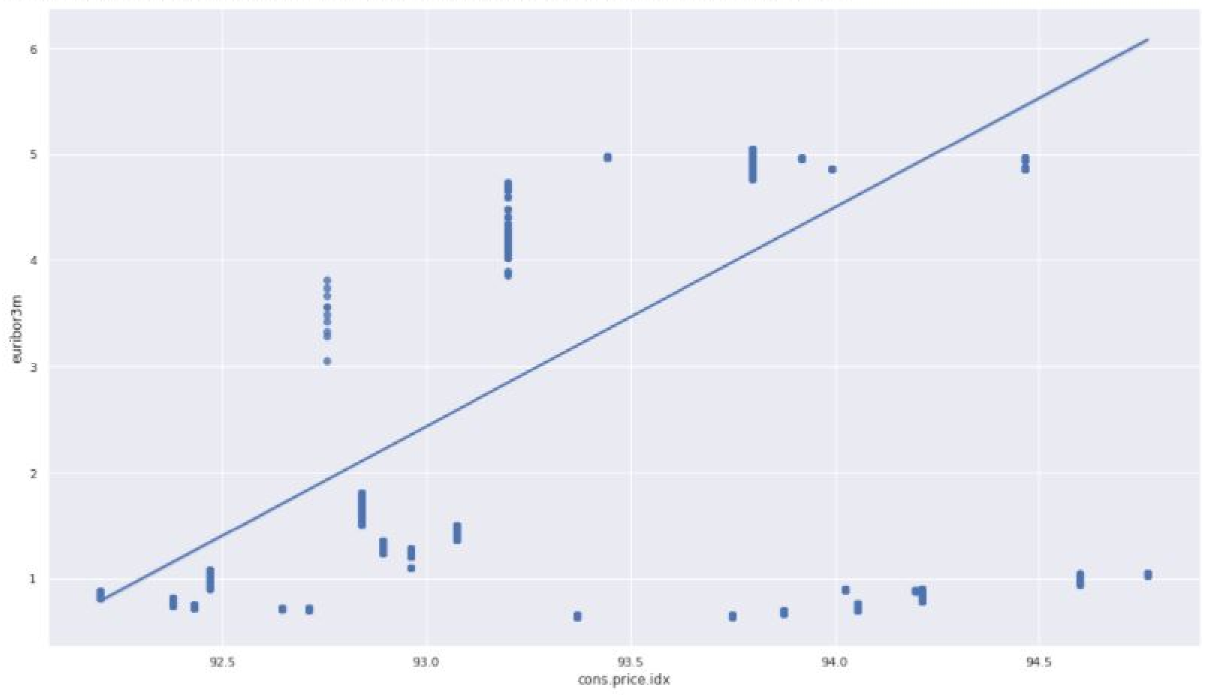
Cov(x,y) $=\sum xy-n\times\bar{x}\times\bar{y}\, n-1=-47.43$

slope of the regression line is, $b_1=Cov(x,y)\sigma^2(x)=-47.43 19.76=-2.42$

intercept is, $b_0=\bar{y}-b_1\times\bar{x}=59.4$

regression equation, $\hat{y}=59.4-2.42(x)$

```
<matplotlib.axes._subplots.AxesSubplot at 0x7f1458086e90>
```



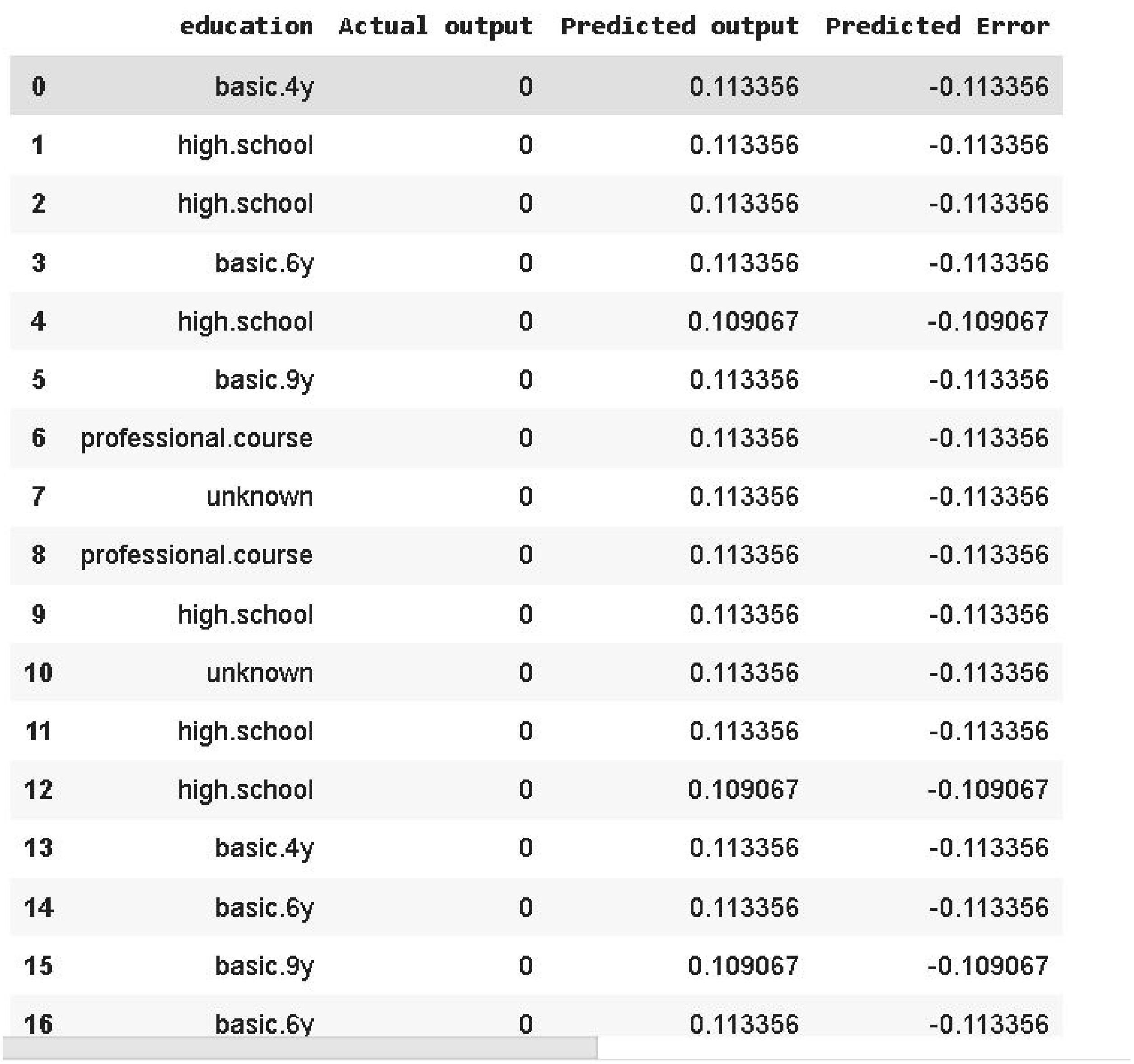
```
[ ] pred = regression_equation(pre_df1['cons.price.idx'], pre_df1['y'])
    print(pred)
```

```
Regression equation is: 7.075 -0.074 (x)
0        0.081530
1        0.081530
2        0.081530
3        0.081530
4        0.081530
           ...
41183    0.024017
41184    0.024017
41185    0.024017
41186    0.024017
41187    0.024017
Name: cons.price.idx, Length: 41188, dtype: float64
```

```
[ ] pred = regression_equation(pre_df1['loan'], pre_df1['y'])
    print(pred)
```

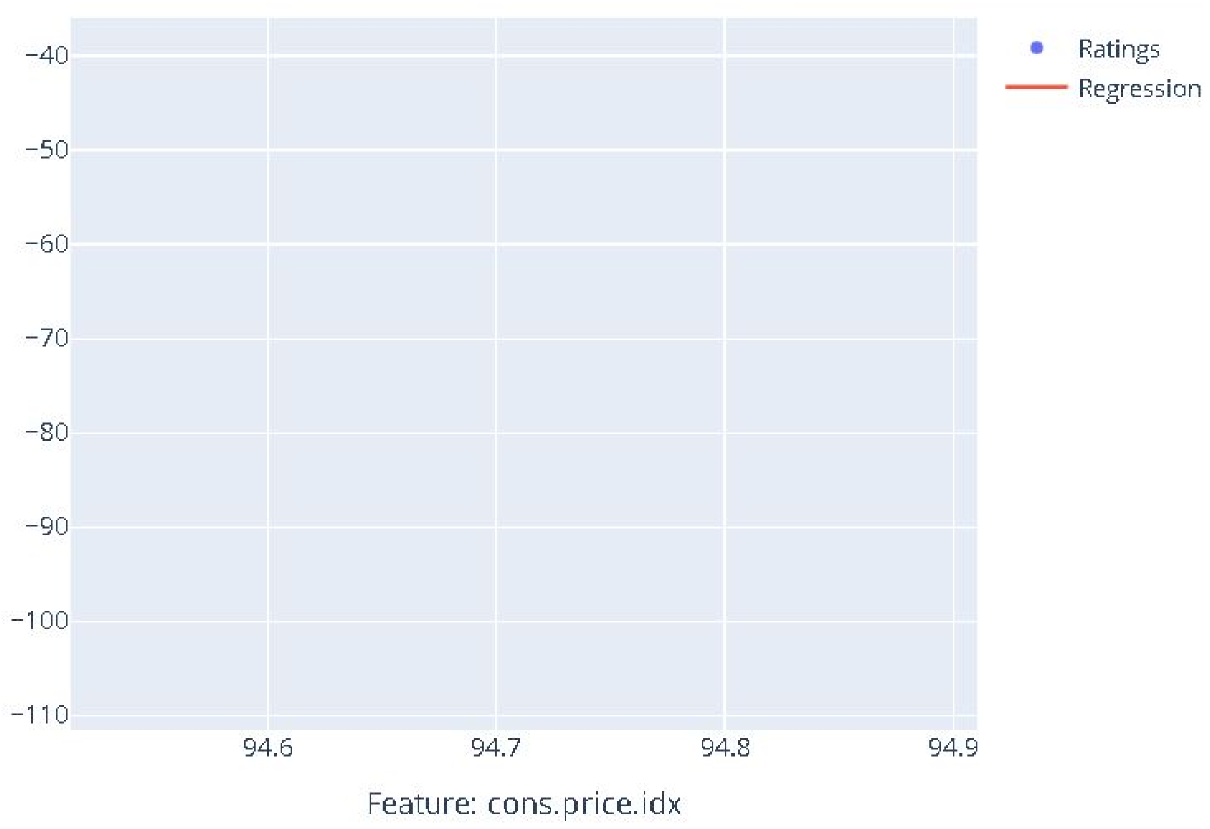
```
Regression equation is: 0.113 -0.002 (x)
0        0.113356
1        0.113356
2        0.113356
3        0.113356
4        0.109067
           ...
41183    0.113356
41184    0.113356
41185    0.113356
41186    0.113356
41187    0.113356
Name: loan, Length: 41188, dtype: float64
```

Residuals



	education	Actual output	Predicted output	Predicted Error
0	basic.4y	0	0.113356	-0.113356
1	high.school	0	0.113356	-0.113356
2	high.school	0	0.113356	-0.113356
3	basic.6y	0	0.113356	-0.113356
4	high.school	0	0.109067	-0.109067
5	basic.9y	0	0.113356	-0.113356
6	professional.course	0	0.113356	-0.113356
7	unknown	0	0.113356	-0.113356
8	professional.course	0	0.113356	-0.113356
9	high.school	0	0.113356	-0.113356
10	unknown	0	0.113356	-0.113356
11	high.school	0	0.113356	-0.113356
12	high.school	0	0.109067	-0.109067
13	basic.4y	0	0.113356	-0.113356
14	basic.6y	0	0.113356	-0.113356
15	basic.9y	0	0.109067	-0.109067
16	basic.6y	0	0.113356	-0.113356

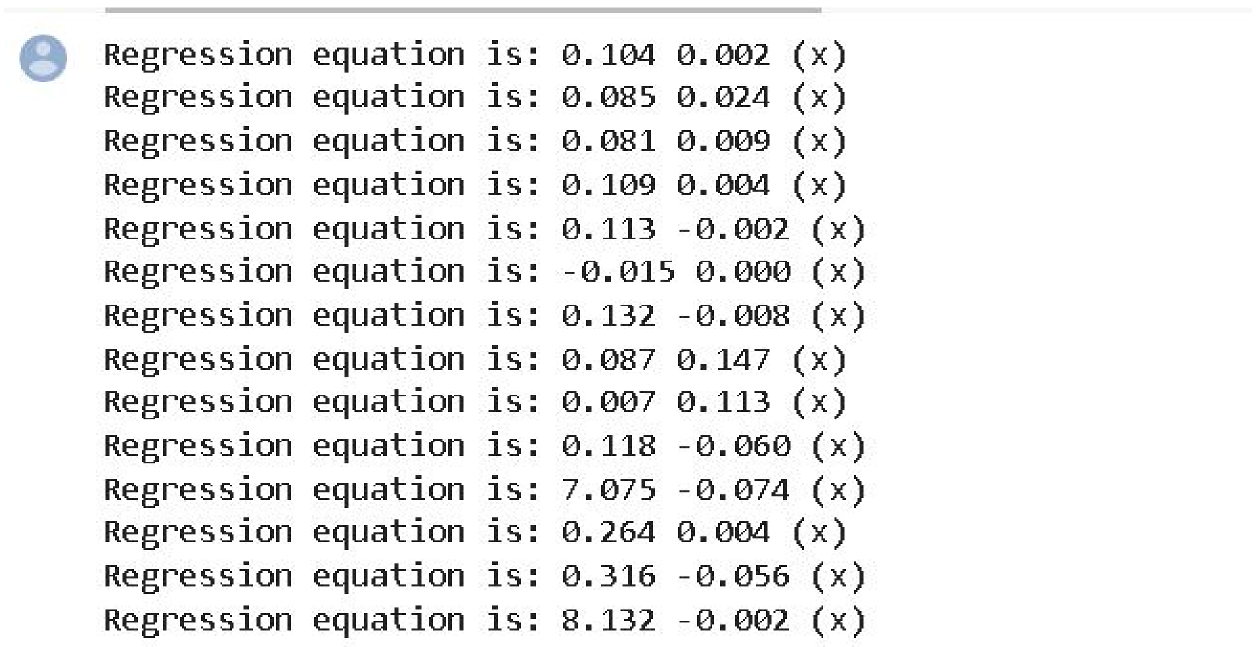
New Prediction



5.4 MULTIPLE REGRESSION

Most data mining applications enjoy a wealth (indeed, a superfluity) of data, with some data sets including hundreds of variables, many of which may have a linear relationship with the target (response) variable.

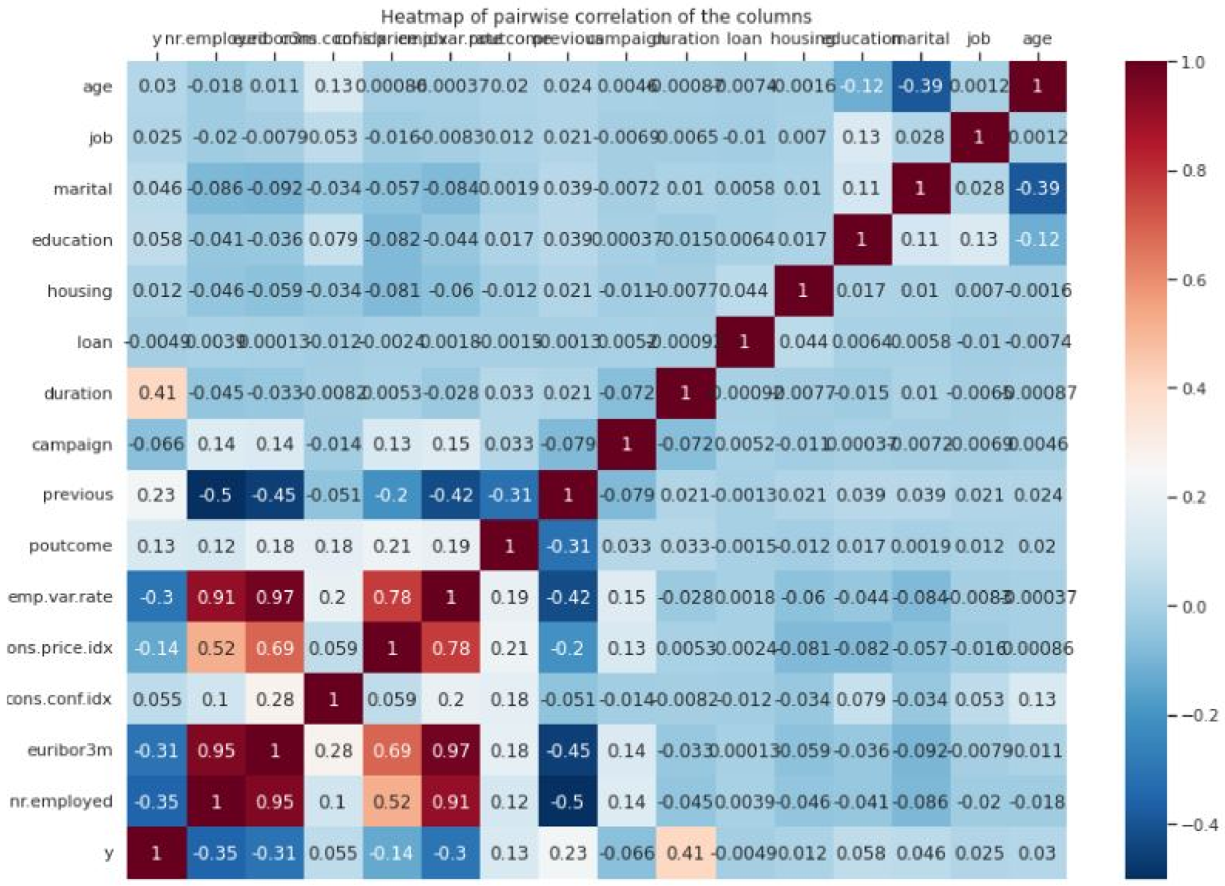
Multiple regression modeling provides an elegant method of describing such re- lationships. Multiple regression models provide improved precision for estima- tion and prediction, analogous to the improved precision of regression estimates over univariate estimates.



```
Regression equation is: 0.104 0.002 (x)
Regression equation is: 0.085 0.024 (x)
Regression equation is: 0.081 0.009 (x)
Regression equation is: 0.109 0.004 (x)
Regression equation is: 0.113 -0.002 (x)
Regression equation is: -0.015 0.000 (x)
Regression equation is: 0.132 -0.008 (x)
Regression equation is: 0.087 0.147 (x)
Regression equation is: 0.007 0.113 (x)
Regression equation is: 0.118 -0.060 (x)
Regression equation is: 7.075 -0.074 (x)
Regression equation is: 0.264 0.004 (x)
Regression equation is: 0.316 -0.056 (x)
Regression equation is: 8.132 -0.002 (x)
```

5.5 Correlation Coefficients

We can verify these graphical findings with the correlation coefficients for all the variables

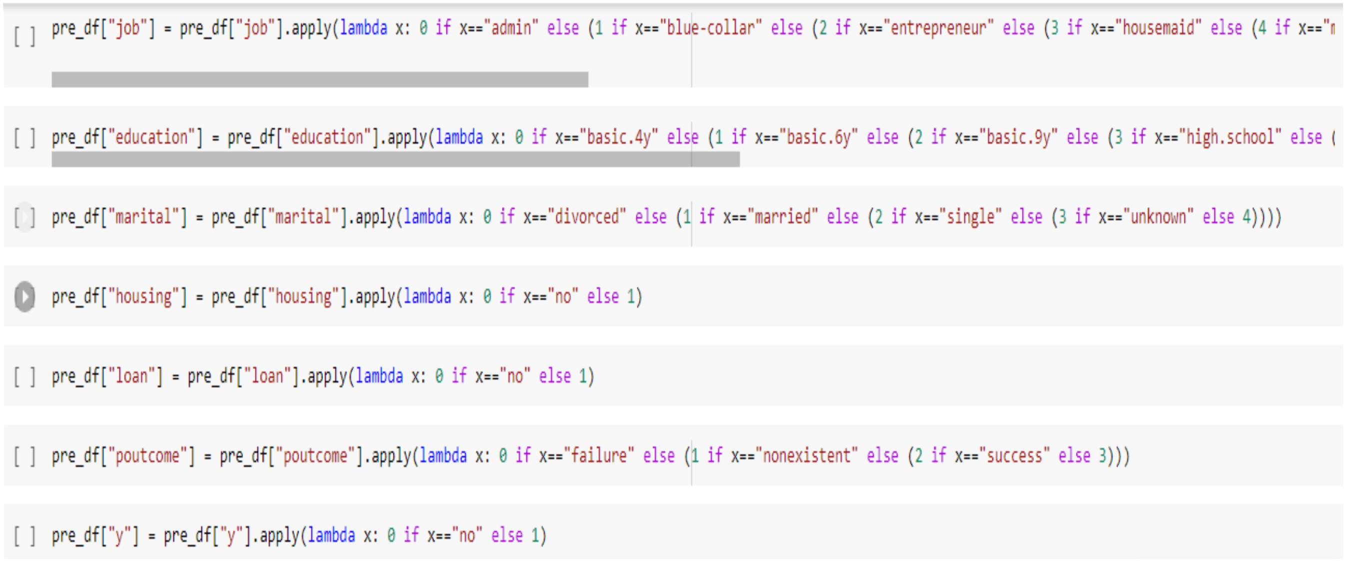


6. Model Building

6.1 k-Nearest Neighbor Algorithm

k-nearest neighbor algorithm, which is most often used for classification, although it can also be used for estimation and prediction. k-Nearest neighbor is an example of instance-based learning, in which the training data set is stored, so that a classification for a new unclassified record may be found simply by comparing it to the most similar records in the training set.

Dataset Preprocessing
We need to transform all categorical data to numerical ones. That's why we are applying some Lambda functions to our dataset columns.

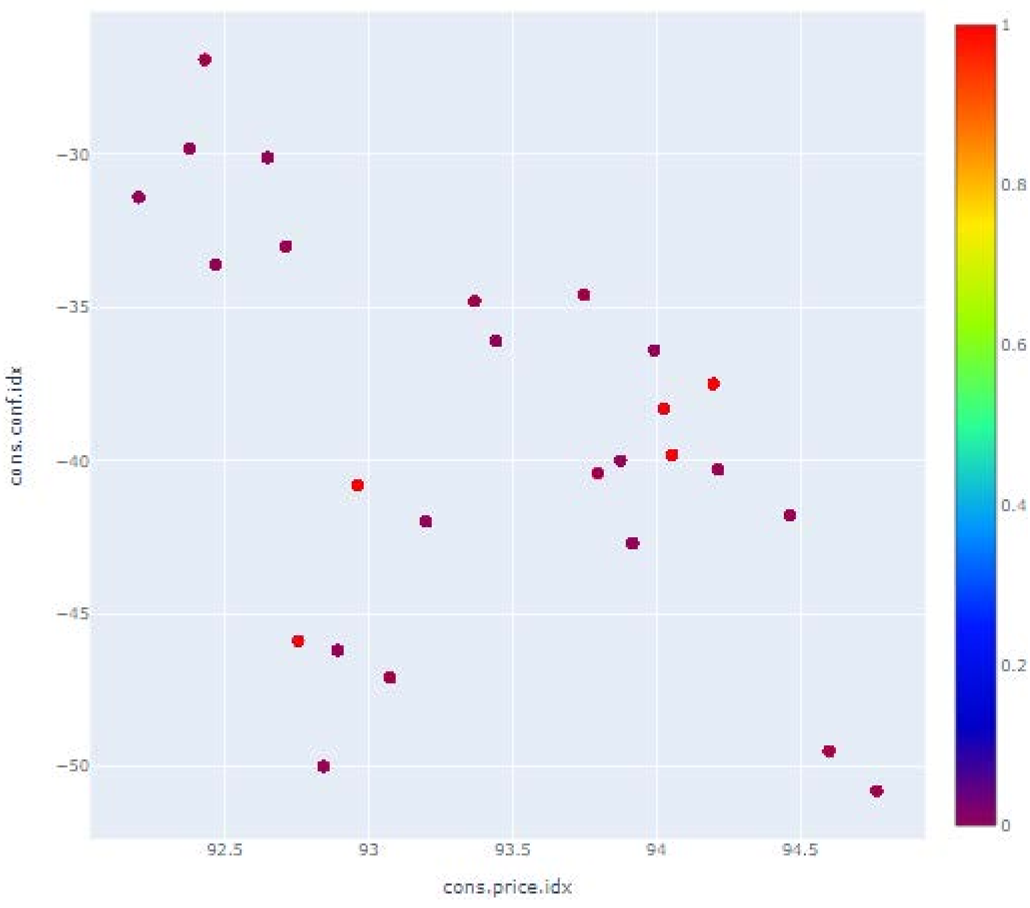


```
[ ] pre_df["job"] = pre_df["job"].apply(lambda x: 0 if x=="admin" else (1 if x=="blue-collar" else (2 if x=="entrepreneur" else (3 if x=="housemaid" else (4 if x=="

[ ] pre_df["education"] = pre_df["education"].apply(lambda x: 0 if x=="basic.4y" else (1 if x=="basic.6y" else (2 if x=="basic.9y" else (3 if x=="high.school" else (

[ ] pre_df["marital"] = pre_df["marital"].apply(lambda x: 0 if x=="divorced" else (1 if x=="married" else (2 if x=="single" else (3 if x=="unknown" else 4))))

pre_df["housing"] = pre_df["housing"].apply(lambda x: 0 if x=="no" else 1)

[ ] pre_df["loan"] = pre_df["loan"].apply(lambda x: 0 if x=="no" else 1)

[ ] pre_df["poutcome"] = pre_df["poutcome"].apply(lambda x: 0 if x=="failure" else (1 if x=="nonexistent" else (2 if x=="success" else 3)))

[ ] pre_df["y"] = pre_df["y"].apply(lambda x: 0 if x=="no" else 1)
```

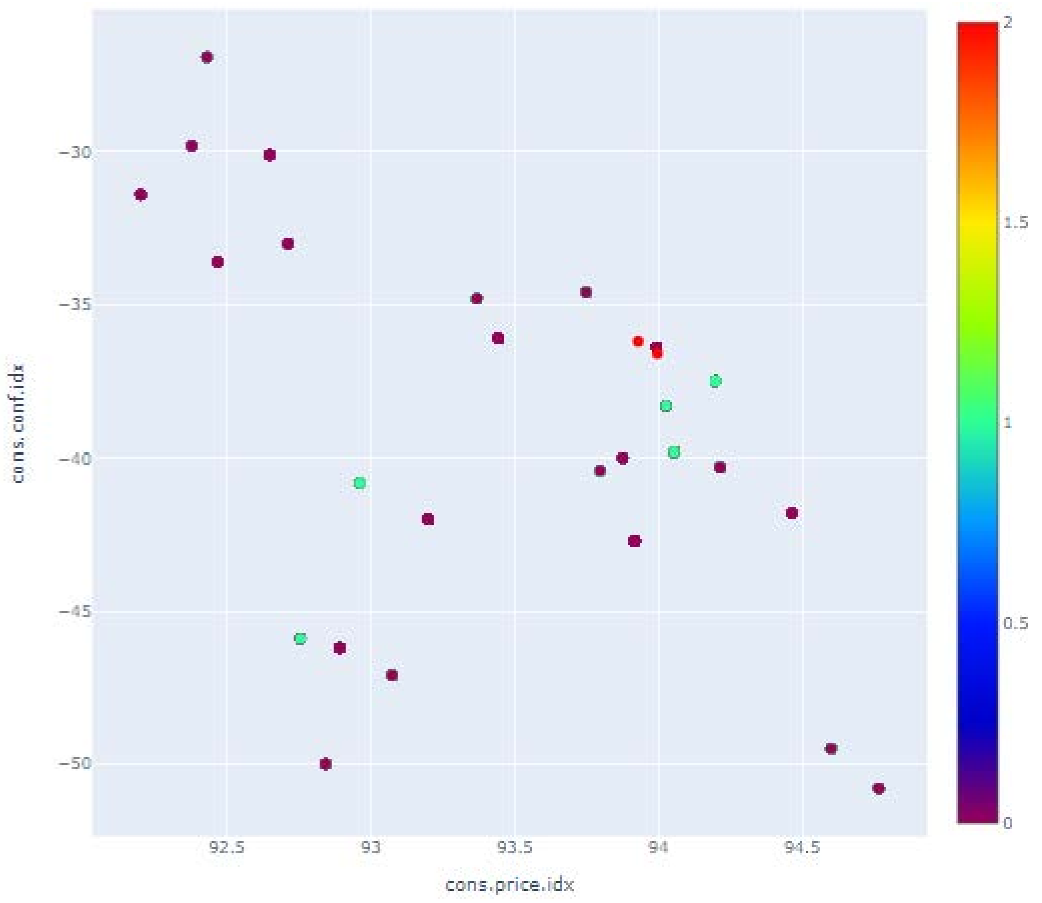
In the dataset ,the column of job, educational, marital, housing, loan, poutcome and the target variable of Bank Marketing dataset is categorical. So, this is how we transform the numeric data using the lambda function.

k-nearest neighbor using scatter plot

In the Bank Marketing dataset field, suppose that we are classifying the type of y. Figure below is a scatter plot of Bank Marketing data set column age against job for a sample of 10000 data set. The particular y is symbolized by the shade of the points



−30
−35
−40
−45
−50
cons.conf.idx
92.5
93
93.5
94
94.5
cons.price.idx
1
0.8
0.6
0.4
0.2
0



Now suppose that we have a new two record, without a Y classification, and would like to classify which Y should be similar attributes

Preparing dataset to be fed into Model

The target/response variable in our dataset is **Y**. So we are putting the drug labels in our target variable y.
The other variables/predictors are the columns like as age ,job marital etc and should be put in our training variable X

```
[ ]  y = pre_df['y']
     X = pre_df.drop(columns=['y'])

     print("Data shape: ", X.shape)
     print("Labels shape: ", y.shape)
```

```
     Data shape:  (41188, 15)
     Labels shape:  (41188,)
```

Training & Testing Set

```
[ ]  # Splittng train:test in 90:10 ratio

     import numpy as np
     from sklearn.model_selection import train_test_split

     X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.10, random_state=42)
```

```
[ ]  print(X_train.shape)
     print(y_train.shape)
     print(X_test.shape)
     print(y_test.shape)
```

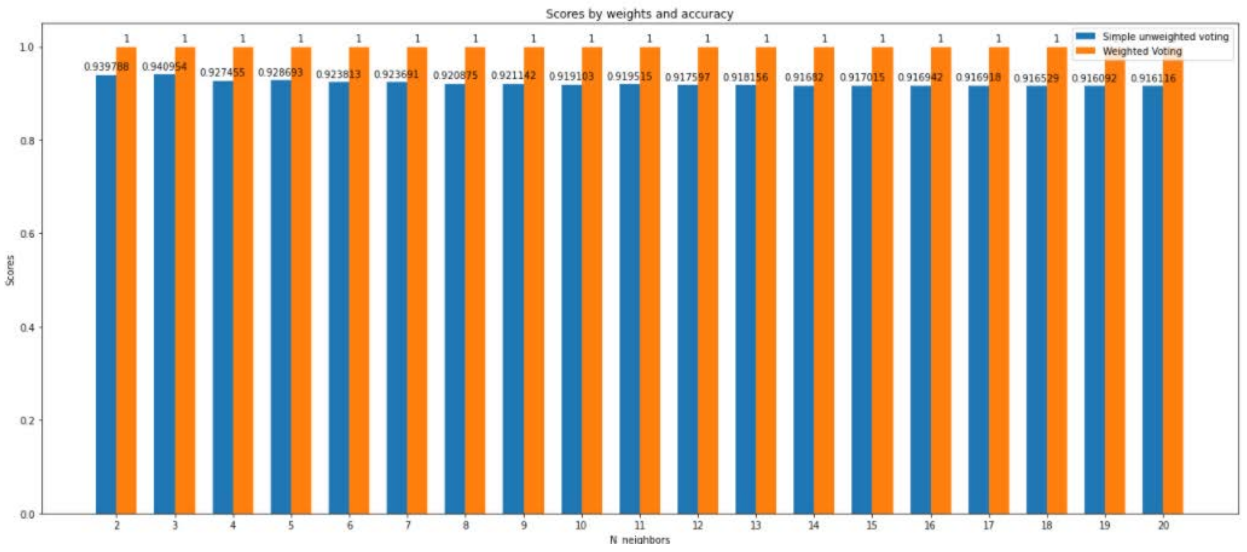
```
     (37069, 15)
     (37069,)
     (4119, 15)
     (4119,)
```

Running the model

We will run our model in different scenarios. The scenarios are as follows:

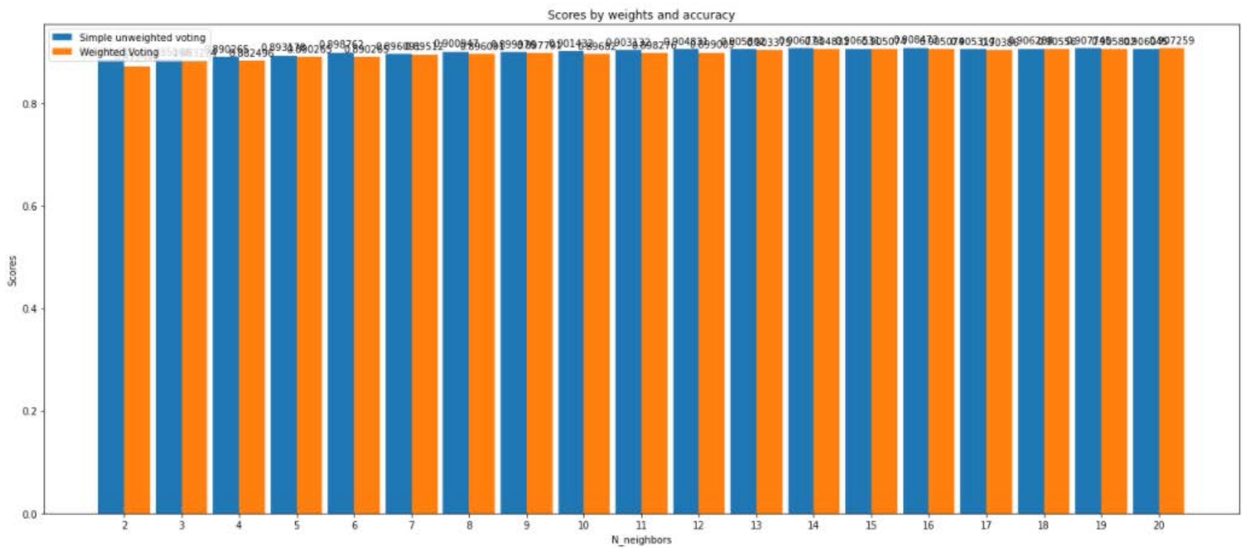
Scenario 1 –

We will train our model with all available data and calculate it for different k's. We will also calculate the accuracy for each configuration and plot it in Matplotlib with all available data.

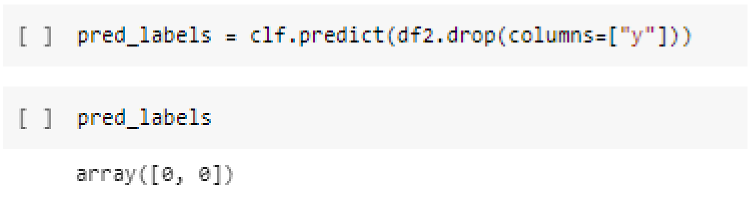


Scenario 2 -

We will train our model with training dataset and calculate it for different k's. We will also calculate the accuracy for each configuration and plot it in Matplotlib with our testing data.



Checking for individual predictions that our model has made
we check which Y our model recommends to the 2 new data in our dataset.



```
[ ]  pred_labels = clf.predict(df2.drop(columns=["y"]))
```

```
[ ]  pred_labels

     array([0, 0])
```

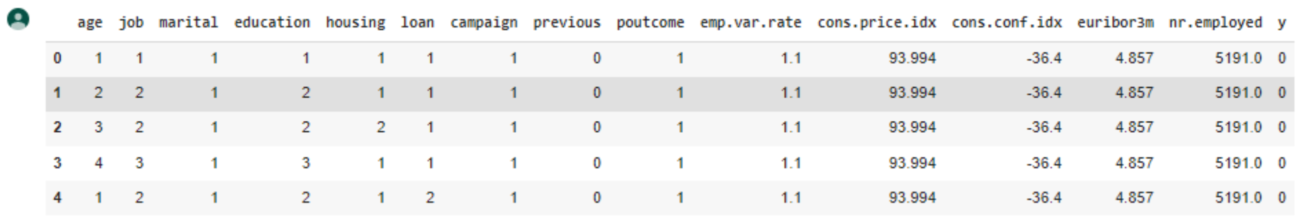
We have classified Two new data and no found new data are under the Yes class.

6.2 Decision Tree

One attractive classification method involves the construction of a decision tree, a collection of decision nodes, connected by branches, extending downward from the root node until terminating in leaf nodes. Beginning at the root node, which by convention is placed at the top of the decision tree diagram, attributes are tested at the decision nodes, with each possible outcome resulting in a branch. Each branch then leads either to another decision node or to a terminating leaf node.

Dataset Preprocessing

We need to transform all categorical data to numerical ones. we are
checking null values of the dataset. we find that our data set has no null values.

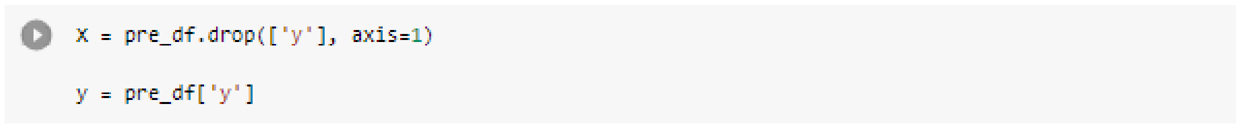


	age	job	marital	education	housing	loan	campaign	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	1	1	1	1	1	1	1	0	1	1.1	93.994	-36.4	4.857	5191.0	0
1	2	2	1	2	1	1	1	0	1	1.1	93.994	-36.4	4.857	5191.0	0
2	3	2	1	2	2	1	1	0	1	1.1	93.994	-36.4	4.857	5191.0	0
3	4	3	1	3	1	1	1	0	1	1.1	93.994	-36.4	4.857	5191.0	0
4	1	2	1	2	1	2	1	0	1	1.1	93.994	-36.4	4.857	5191.0	0

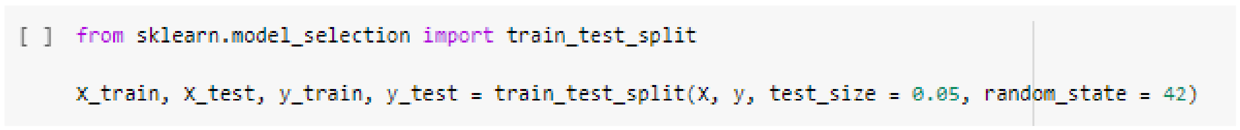
Preparing dataset to be fed into Model

The target/response variable in our dataset is **Y**. So, we are putting the class labels in our target variable y.

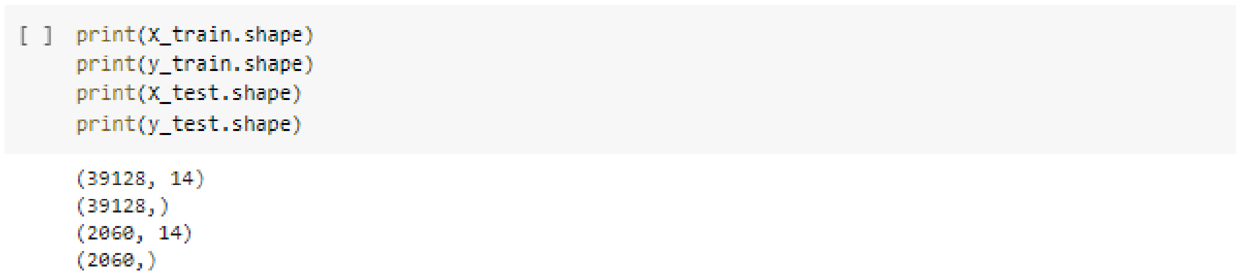
The other variables/predictors are the columns like as age,job, marital ertc and should be put in our training variable X. We have taken 5% of our data set for data testing.



```
X = pre_df.drop(['y'], axis=1)

y = pre_df['y']
```



```
[ ] from sklearn.model_selection import train_test_split

    X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.05, random_state = 42)
```



```
[ ] print(X_train.shape)
    print(y_train.shape)
    print(X_test.shape)
    print(y_test.shape)

    (39128, 14)
    (39128,)
    (2060, 14)
    (2060,)
```

Decision Tree – CART

We will now build our model of Decision Tree Classifier.

The classification and regression trees(CART) method was suggested by Bramante al. [1] in 1984. The decision trees produced by CART are strictly binary, containing exactly two branches for each decision node. CART recursively partitions the records in the training data set into subsets of records with similar values for the target attribute. The CART algorithm grows the tree by conducting for each decision node, an exhaustive search of all available variables and all possible splitting values, selecting the optimal split according to the following criteria (from Kennedy et al. [2]).

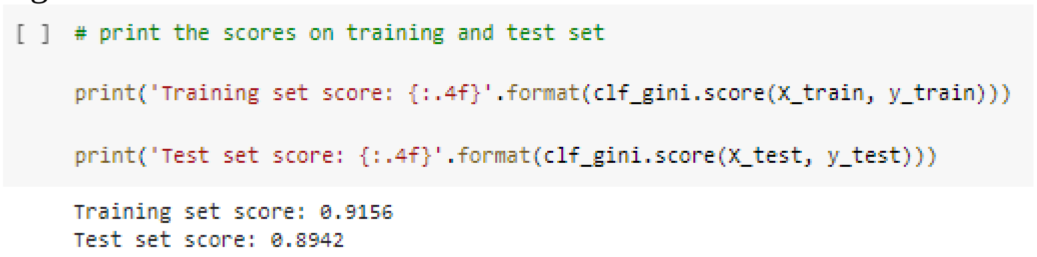
CART (Classification and Regression Trees) is very similar to C4.5, but it differs in that it supports numerical target variables (regression) and does not compute rule sets. CART constructs binary trees using the feature and threshold that yield the largest information gain at each node.

Measures for selecting the Best Split
The measures developed for selecting the best split are often based on the degree of impurity of the child nodes. The smaller the degree of impurity, the more skewed the class distribution. For example, a node with class distribution (0,1) has zero impurity where's a node with uniform class distribution has the highest impurity. Examples of impurity measures include:

- Entropy(t) = −∑c−1i=0p(i|t)log2p(i|t)
- Gini(t) = 1−∑c−1i=0[p(i|t)]2
- Classification Error(t) = 1−maxi[p(i|t)]

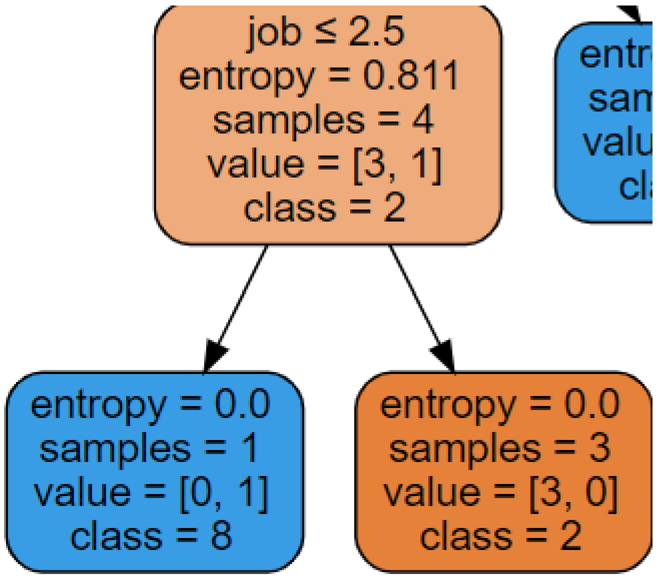
where, c is the number of classes and 0log20=0 in entropy calculations.

The accuracy of the training and testing set is coming by applying the CART algorithm to our data set. 0.9156 and 0.8942



```
[ ]  # print the scores on training and test set

     print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))

     print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))

     Training set score: 0.9156
     Test set score: 0.8942
```

We have drawn a tree in our data set using Graph Viz based on the stab attribute. In the tree we see that our training variable has got a stable and unstable value. Here,1st step we get Gini's value 0, we've got 2 pure leaf nodes.



Calculating a Confusion Matrix of CART algorithm

Below is the process for calculating a confusion Matrix.

1. You need a test dataset or a validation dataset with expected outcome values.
2. Make a prediction for each row in your test dataset.
3. From the expected outcomes and predictions count:
    - The number of correct predictions for each class.
    - The number of incorrect predictions for each class, organized by the class that was predicted.

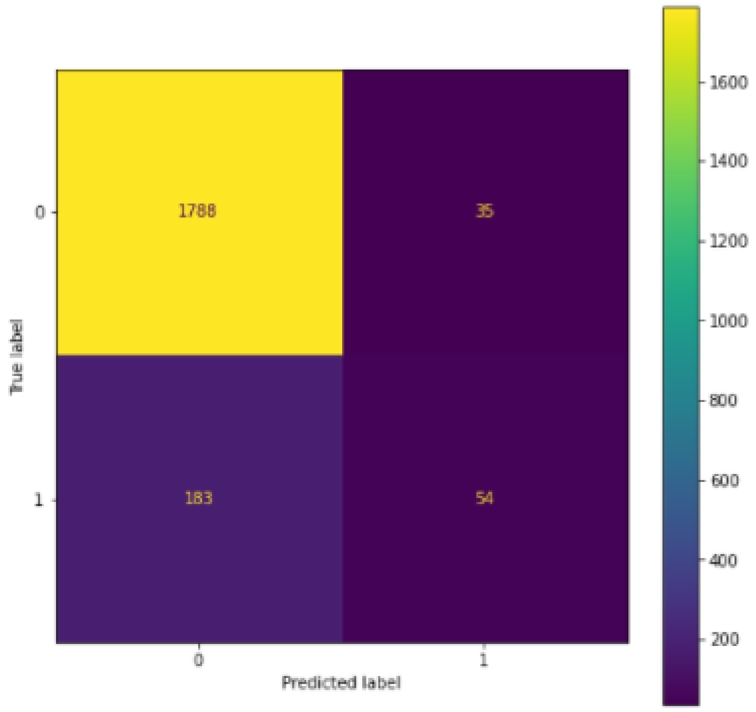
These numbers are then organized into a table, or a matrix as follows:

- Expected down the side: Each row of the matrix corresponds to a predicted class.
- Predicted across the top: Each column of the matrix corresponds to an actual class.

The counts of correct and incorrect classification are then filled into the table. The total number of correct predictions for a class go into the expected row for that class value and the predicted column for that class value.

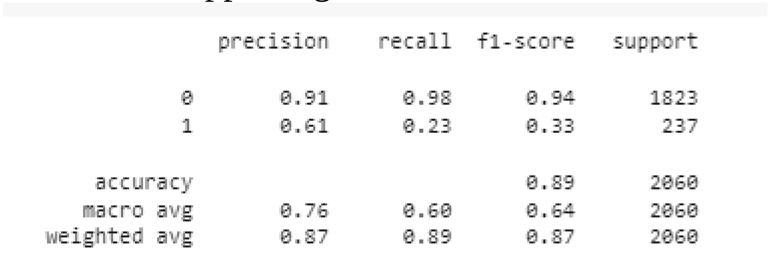
In the same way, the total number of incorrect predictions for a class go into the expected row for that class value and the predicted column for that class value.

Using the Confusion Matrix, we see in our data set that the value of stable is recognized as no or "0" 1788 times and the value of Yes or "1"is recognized 54 times. And we can see there that misclassification is happening in our data set or it is very medium.



Support and Confidence of CART algorithm
The **support** of the decision rule refers to the proportion of records in the dataset that rest in that particular terminal leaf node. The confidence of the rule refers to the proportion of records in the leaf node for which the decision rule is true.

By support here we mean the no or ‘0’ class is supporting 1823 data and the Yes or ‘1’ class is supporting 237 data.

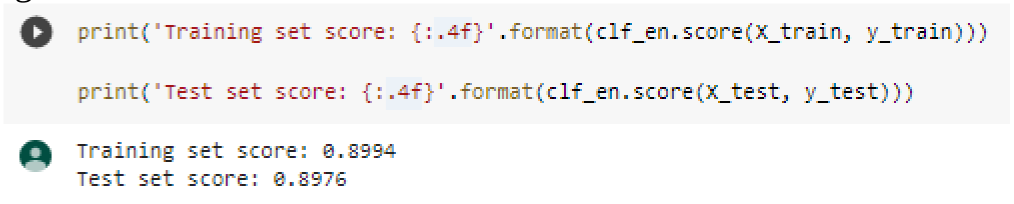


```
              precision    recall  f1-score   support

           0       0.91      0.98      0.94      1823
           1       0.61      0.23      0.33       237

    accuracy                           0.89      2060
   macro avg       0.76      0.60      0.64      2060
weighted avg       0.87      0.89      0.87      2060
```

Decision Tree - C4.5
The C4.5 algorithm is Quinlan’s extension of his own ID3 algorithm for generating decision trees \cite{10.5555/152181}. Just as with CART, the C4.5 algorithm recursively visits each decision node, selecting the optimal split, until no further splits are possible. However, there are interesting differences between CART and C4.5:

- Unlike CART, the C4.5 algorithm is not restricted to binary splits. Whereas CART always produces a binary tree, C4.5 produces a tree of more variable shape.
- For categorical attributes, C4.5 by default produces a separate branch for each value of the categorical attribute. This may result in more “congested” than desired, since some values may have low frequency or may naturally be associated with other values.

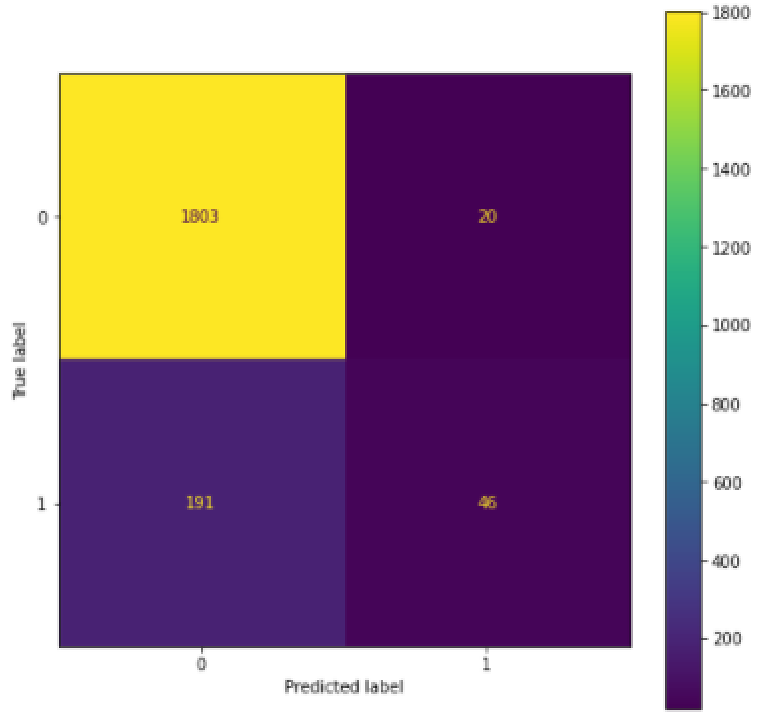
The accuracy of the training and testing set is coming by applying the C4.5 algorithm to our data set. 0.8994 and 0.8976



```
print('Training set score: {:.4f}'.format(clf_en.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

```
Training set score: 0.8994
Test set score: 0.8976
```

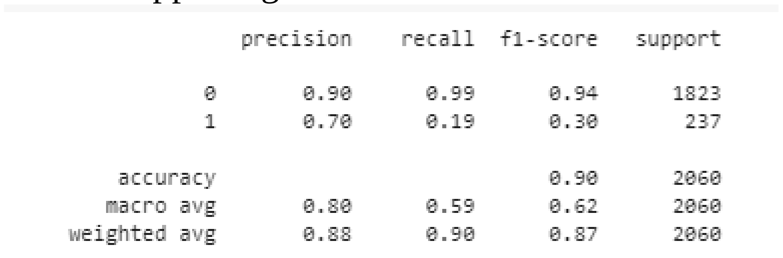
Calculating a Confusion Matrix of C4.5 algorithm



Using the Confusion Matrix, we see in our data set that the value of stable is recognized as no or “0” 1803 times and the value of Yes or “1”is recognized 46 times. And we can see there that misclassification is happening in our data set or it is very medium.

Support and Confidence of C4.5algorithm

By support here we mean the stable class is supporting 1823 data and the unstable class is supporting 237 data.

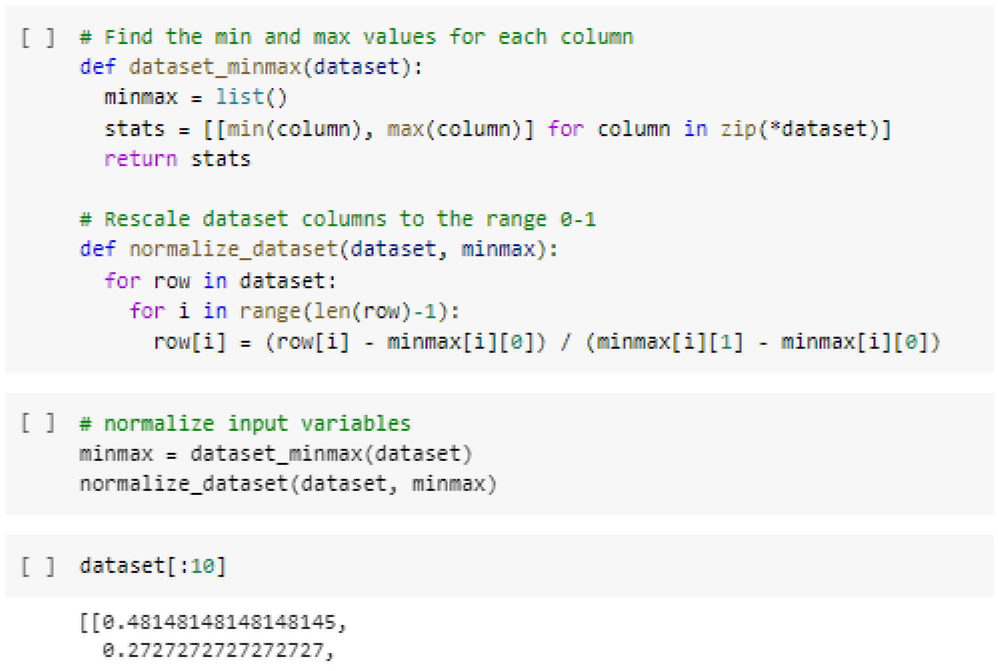


```
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      1823
           1       0.70      0.19      0.30       237

    accuracy                           0.90      2060
   macro avg       0.80      0.59      0.62      2060
weighted avg       0.88      0.90      0.87      2060
```

So, after applying CART and C4.5 algorithms, we see that two algorithms are good for our data set

6.3 Neural Networks

Neural networks are used for effective data mining in order to turn raw data into useful information. Neural networks look for patterns in large batches of data.
Frist, we normalize our data set for this neural model



```
[ ]  # Find the min and max values for each column
     def dataset_minmax(dataset):
       minmax = list()
       stats = [[min(column), max(column)] for column in zip(*dataset)]
       return stats

     # Rescale dataset columns to the range 0-1
     def normalize_dataset(dataset, minmax):
       for row in dataset:
         for i in range(len(row)-1):
           row[i] = (row[i] - minmax[i][0]) / (minmax[i][1] - minmax[i][0])
```

```
[ ]  # normalize input variables
     minmax = dataset_minmax(dataset)
     normalize_dataset(dataset, minmax)
```

```
[ ]  dataset[:10]
```

```
     [[0.48148148148148145,
       0.2727272727272727,
```

A Neural Network is made up of 3 components:

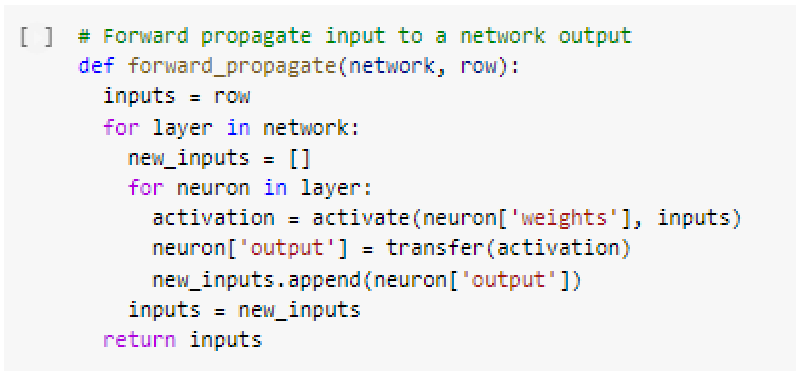
- Input Layer.
- Hidden Layers.
- Output Layer.

Sigmoid Activation Function

A weighted sum of inputs is passed through an activation function and this output serves as an input to the next layer.
Activation functions are a critical part of the design of a neural network. The choice of activation function in the hidden layer will control how well the network model learns the training dataset. The choice of activation function in the output layer will define the type of predictions the model can make

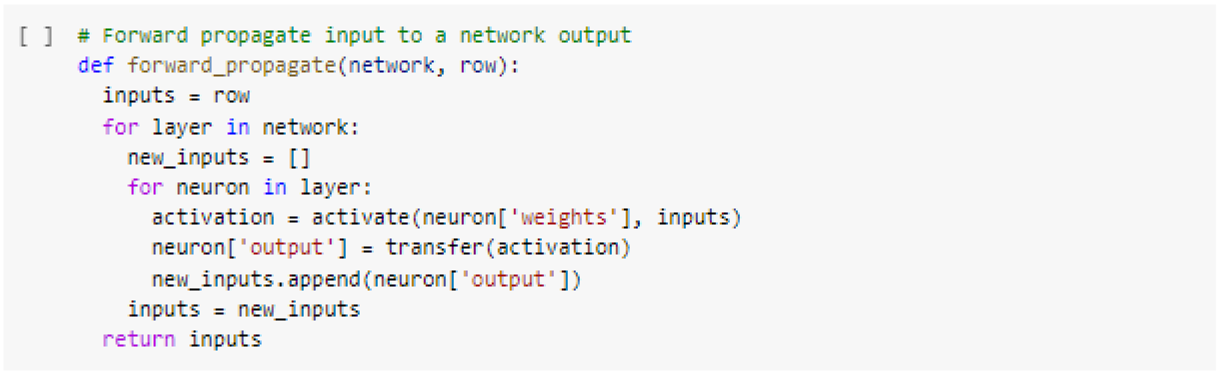
Now we calculate an input for neuron activation.

$y=1(1+e-a)$

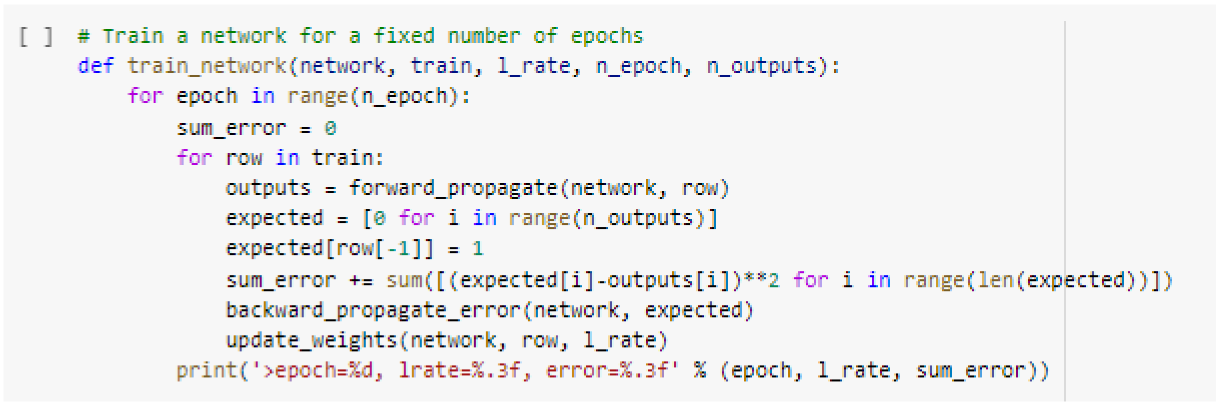


```
[ ] # Forward propagate input to a network output
    def forward_propagate(network, row):
      inputs = row
      for layer in network:
        new_inputs = []
        for neuron in layer:
          activation = activate(neuron['weights'], inputs)
          neuron['output'] = transfer(activation)
          new_inputs.append(neuron['output'])
        inputs = new_inputs
      return inputs
```

Backward propagation is an important mathematical tool for improving the accuracy of predictions in data mining.
To measure how well the output predictions fit the actual target values, most neural network models use the sum of squared errors.
Frist we calculate error rating then we calculate backward



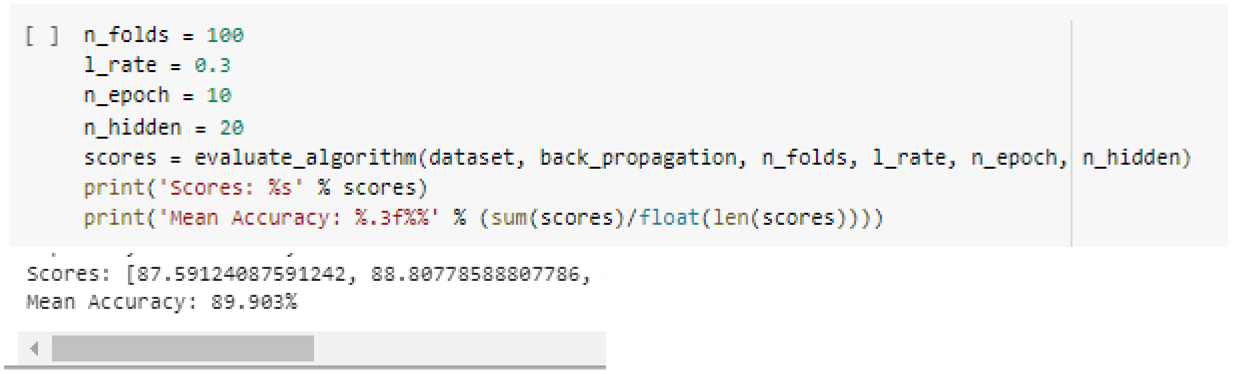
```
[ ] # Forward propagate input to a network output
    def forward_propagate(network, row):
      inputs = row
      for layer in network:
        new_inputs = []
        for neuron in layer:
          activation = activate(neuron['weights'], inputs)
          neuron['output'] = transfer(activation)
          new_inputs.append(neuron['output'])
        inputs = new_inputs
      return inputs
```



```
[ ] # Train a network for a fixed number of epochs
    def train_network(network, train, l_rate, n_epoch, n_outputs):
        for epoch in range(n_epoch):
            sum_error = 0
            for row in train:
                outputs = forward_propagate(network, row)
                expected = [0 for i in range(n_outputs)]
                expected[row[-1]] = 1
                sum_error += sum([(expected[i]-outputs[i])**2 for i in range(len(expected))])
                backward_propagate_error(network, expected)
                update_weights(network, row, l_rate)
            print('>epoch=%d, lrate=%.3f, error=%.3f' % (epoch, l_rate, sum_error))
```

Let’s we find our mean accuracy.
Fold: Divided our data set in some fold ,some fold for training and 1 fold for testing. Fold change randomly.
Learning rate: The learning rate is a hyper parameter that controls how much to change our model in response to the estimated error each time the model weights are updated. The learning rate the most important hyper parameter when configuring your neural network.

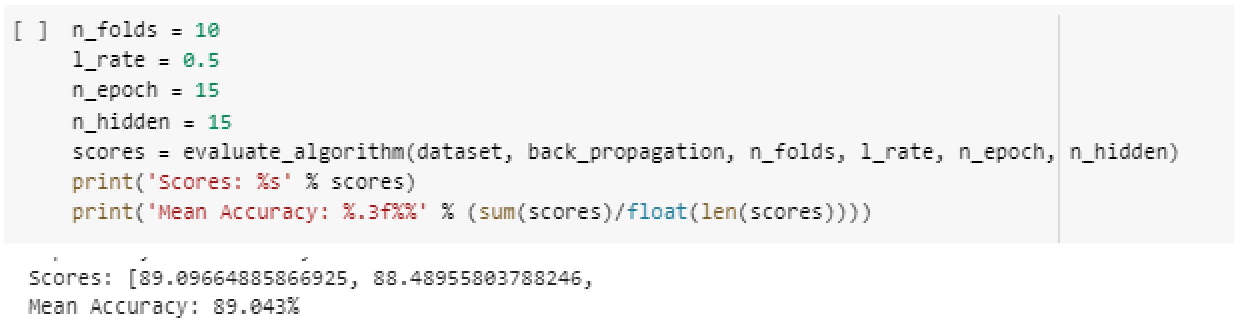
Epoch: An epoch means training the neural network with all the training data for one cycle. A forward pass and a backward pass together are counted as one pass. We use a part of the dataset to train the neural network.
Hidden layer: A hidden layer neural network is a layer in between input layers and output layers, where artificial neurons take in a set of weighted inputs and produce an output through an activation function.



```
[ ]  n_folds = 100
     l_rate = 0.3
     n_epoch = 10
     n_hidden = 20
     scores = evaluate_algorithm(dataset, back_propagation, n_folds, l_rate, n_epoch, n_hidden)
     print('Scores: %s' % scores)
     print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

```
Scores: [87.59124087591242, 88.80778588807786,
Mean Accuracy: 89.903%
```

When fold=100 ,learning rate=0.3 ,epoch=10 and hidden layer=20 we get mean accuracy 89.903%



```
[ ]  n_folds = 10
     l_rate = 0.5
     n_epoch = 15
     n_hidden = 15
     scores = evaluate_algorithm(dataset, back_propagation, n_folds, l_rate, n_epoch, n_hidden)
     print('Scores: %s' % scores)
     print('Mean Accuracy: %.3f%%' % (sum(scores)/float(len(scores))))
```

```
Scores: [89.09664885866925, 88.48955803788246,
Mean Accuracy: 89.043%
```

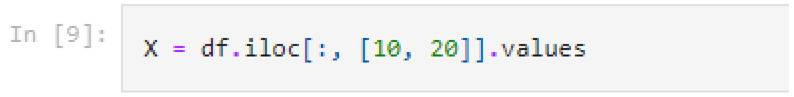
When fold=10 ,learning rate=0.5 ,epoch=15 and hidden layer=15 we get mean accuracy 89.043%

6.4 k-means Clustering

Clustering refers to the grouping of records, observations, or cases into classes of similar objects. A cluster is a collection of records that are similar to one another and dissimilar to records in other clusters.
Instead, clustering algorithms seek to segment the entire data set into relatively homogeneous subgroups or clusters, where the similarity of the records within the cluster is maximized, and the similarity to records outside this cluster is minimized.

Selecting Data for clustering analysis

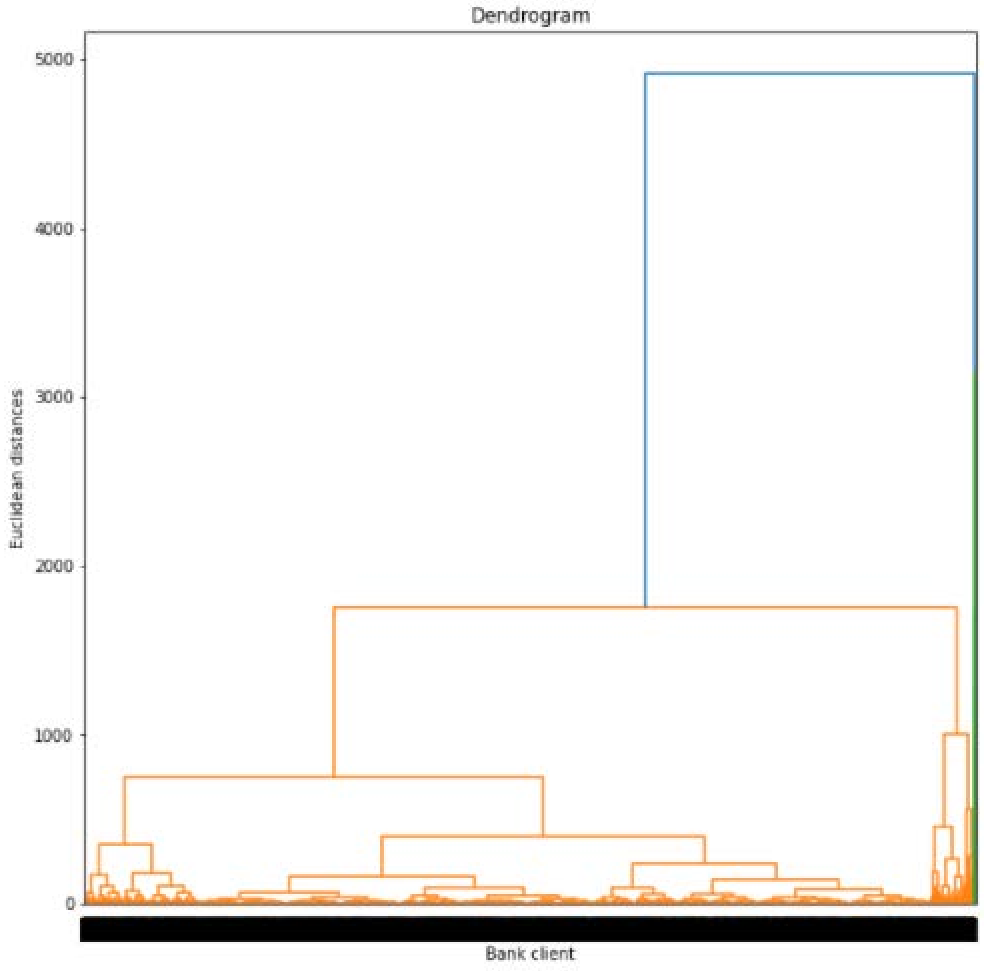
Here we are selecting data attributes for clustering analysis.



```
In [9]:  X = df.iloc[:, [10, 20]].values
```

Hierarchical Clustering Methods

In hierarchical clustering, a treelike cluster structure (dendrogram) is created through recursive partitioning (divisive methods) or combining (agglomerative) of existing clusters.

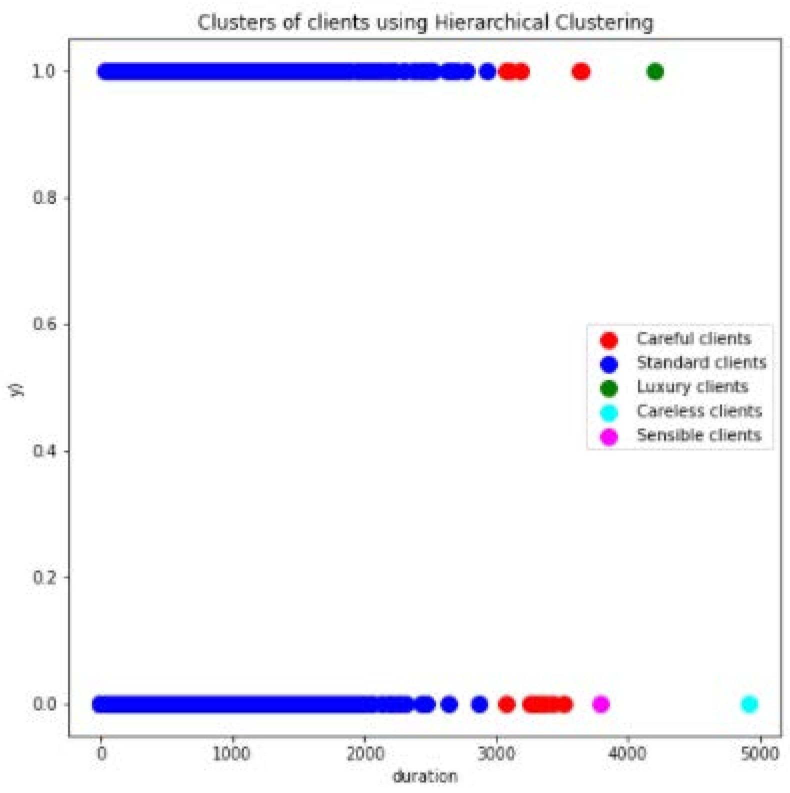


Agglomerative Clustering

Agglomerative clustering methods [1] initialize each observation to be a tiny cluster of its own. Then, in succeeding steps, the two closest clusters are aggregated into a new combined cluster. In this way, the number of clusters in the data set is reduced by one at each step. Eventually, all records are combined into a single huge cluster.
Divisive clustering methods begin with all the records in one big cluster, with the most dissimilar records being split off recursively, into a separate cluster, until each record represents its own cluster.

Single linkage

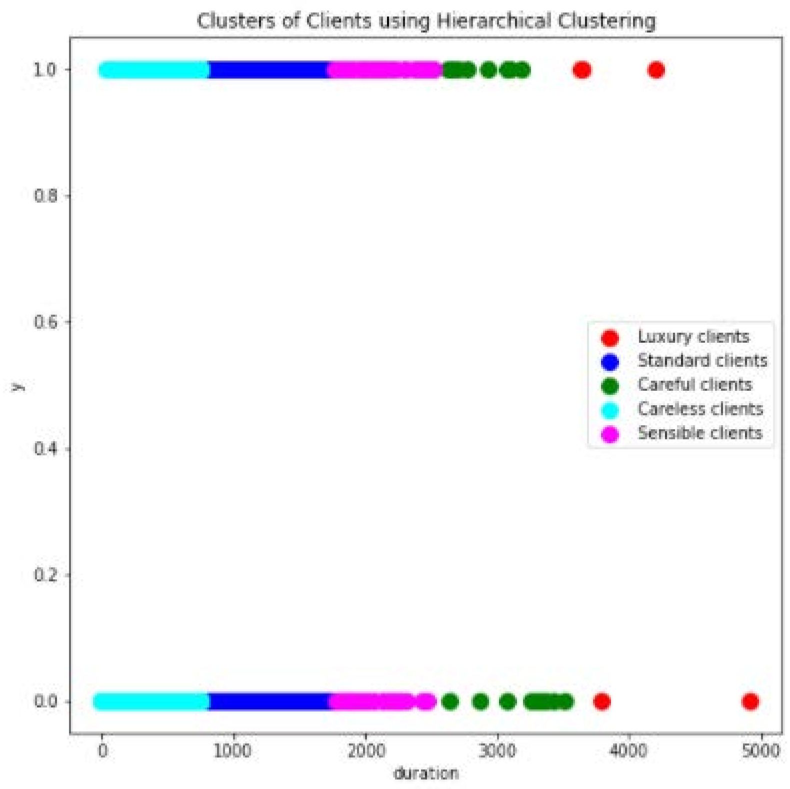
Single linkage, sometimes termed the nearest-neighbor approach, is based on the minimum distance between any record in cluster A and any record in cluster B.



Bank Marketing data set we have applied single linkage on duration & Y column and we have seen the relation of our target variable as in the picture above.

Complete linkage

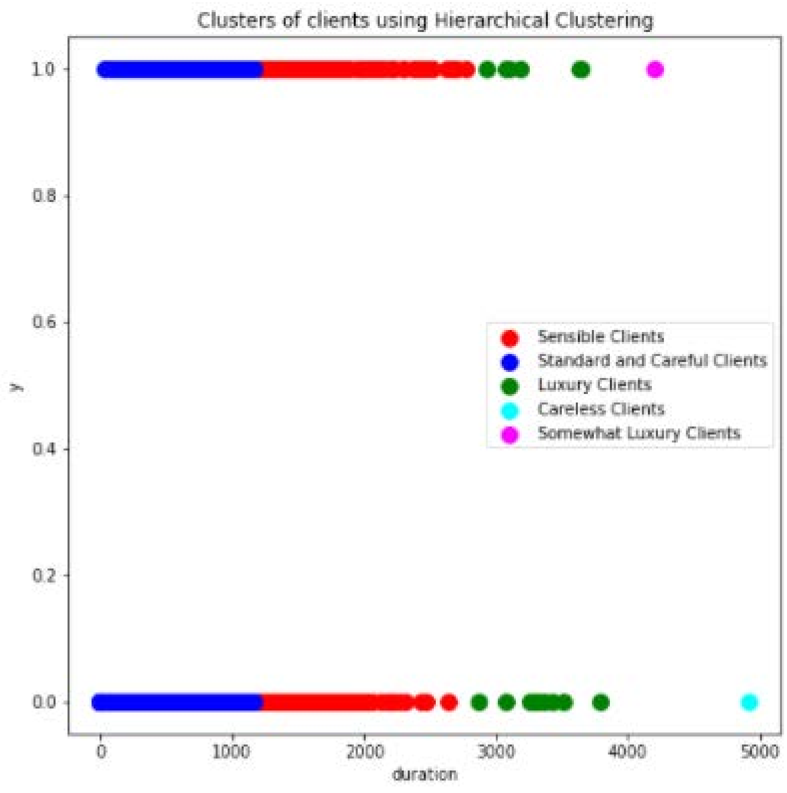
Complete linkage, sometimes termed the farthest-neighbor approach, is based on the maximum distance between any record in cluster A and any record in cluster B.



Bank Marketing data set we have applied single linkage on duration & Y column and we have seen the relation of our target variable as in the picture above.

Average linkage

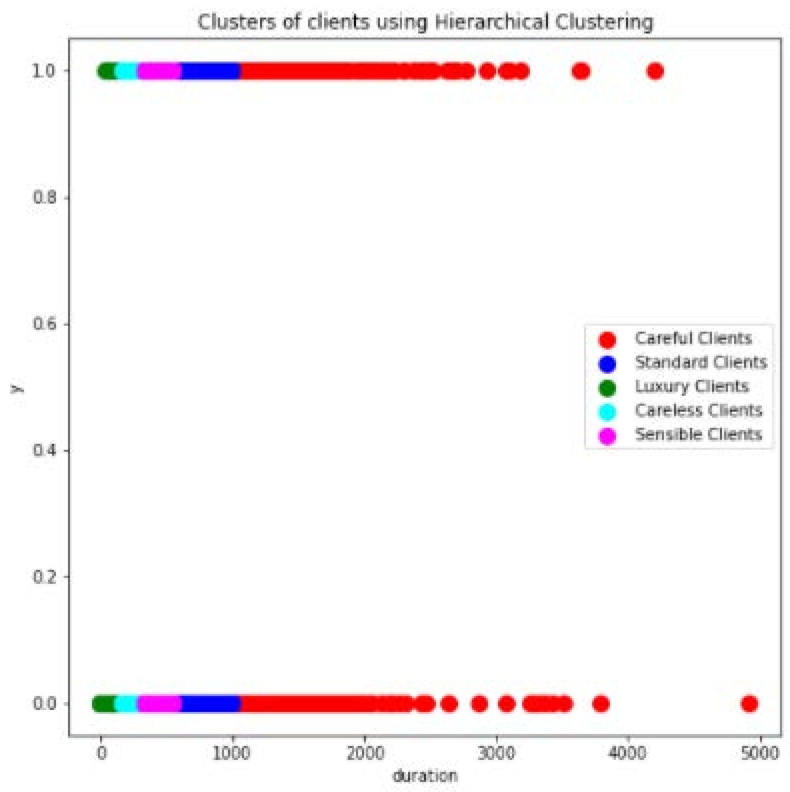
average linkage, the criterion is the average distance of all the records in cluster A from all the records in cluster B. The resulting clusters tend to have approximately equal within-cluster variability.



Bank Marketing data set we have applied single linkage on duration & Y column and we have seen the relation of our target variable as in the picture above.

Ward linkage

Ward´s linkage is a method for hierarchical cluster analysis. The idea has much in common with analysis of variance (ANOVA). The linkage function specifying the distance between two clusters is computed as the increase in the "error sum of squares" (ESS) after fusing two clusters into a single cluster.

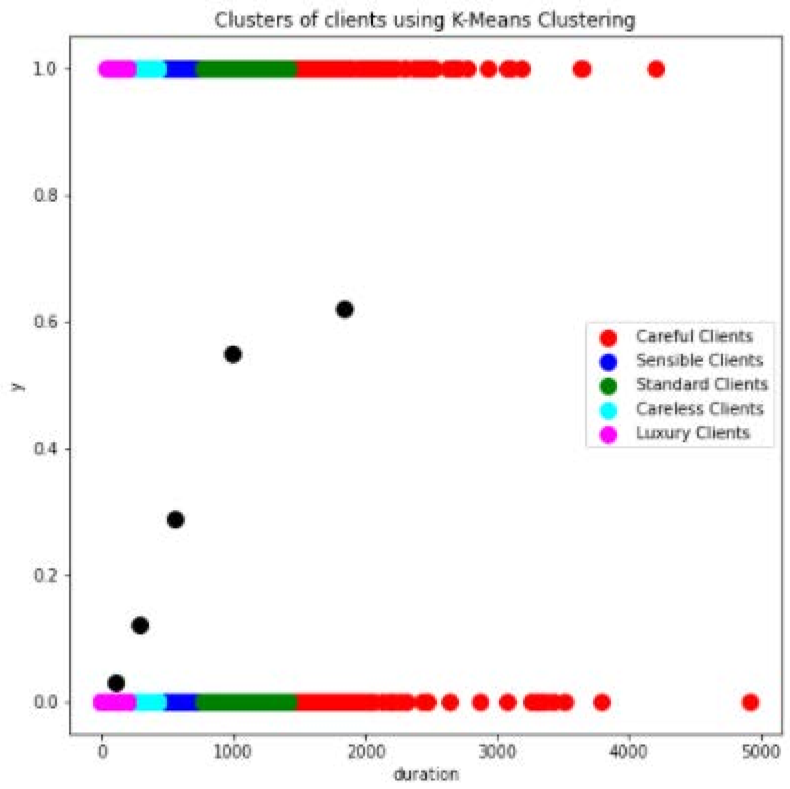


Bank Marketing data set we have applied single linkage on duration & Y column and we have seen the relation of our target variable as in the picture above.

k-means Clustering

The k-means clustering algorithm [2] is a straightforward and effective algorithm for finding clusters in data. The algorithm proceeds as follows.

- Step 1: Ask the user how many clusters k the data set should be partitioned into.
- Step 2: Randomly assign k records to be the initial cluster center locations.
- Step 3: For each record, find the nearest cluster center. Thus, in a sense, each cluster center "owns" a subset of the records, thereby representing a partition of the data set. We therefore have k clusters, $C_1, C_2, \ldots, C_k$.
- Step 4: For each of the k clusters, find the cluster centroid, and update the location of each cluster center to the new value of the centroid.
- Step 5: Repeat steps 3 to 5 until convergence or termination.



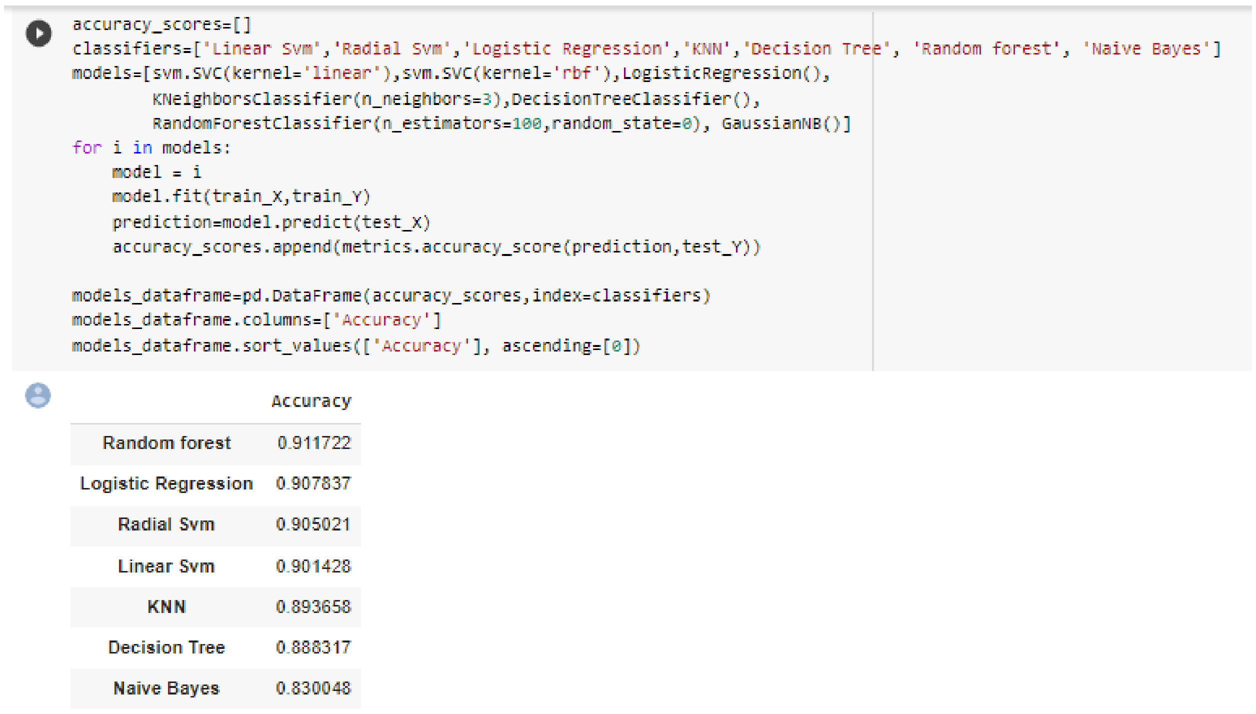
The name of clusters is given based on their products for example, when referring to a product with stable and unstable, we have used cyan color.

Model evaluation plays a crucial role while developing a predictive machine learning model. Building just a predictive model without checking does not count as a fit model but a model which gives maximum accuracy surely does count a good one. For this, we need to check on the metrics and make improvements accordingly until we get our desired accuracy rate.

7. Model evaluation

7.1 Evaluation Metrics

Accuracy

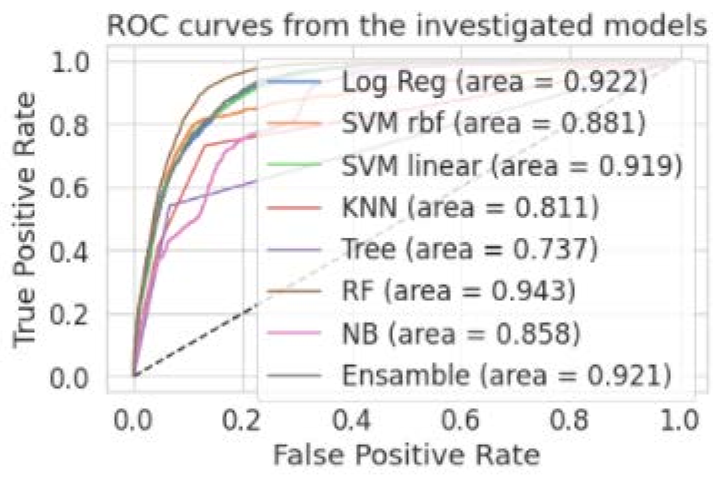


```python
accuracy_scores=[]
classifiers=['Linear Svm','Radial Svm','Logistic Regression','KNN','Decision Tree', 'Random forest', 'Naive Bayes']
models=[svm.SVC(kernel='linear'),svm.SVC(kernel='rbf'),LogisticRegression(),
        KNeighborsClassifier(n_neighbors=3),DecisionTreeClassifier(),
        RandomForestClassifier(n_estimators=100,random_state=0), GaussianNB()]
for i in models:
    model = i
    model.fit(train_X,train_Y)
    prediction=model.predict(test_X)
    accuracy_scores.append(metrics.accuracy_score(prediction,test_Y))

models_dataframe=pd.DataFrame(accuracy_scores,index=classifiers)
models_dataframe.columns=['Accuracy']
models_dataframe.sort_values(['Accuracy'], ascending=[0])
```

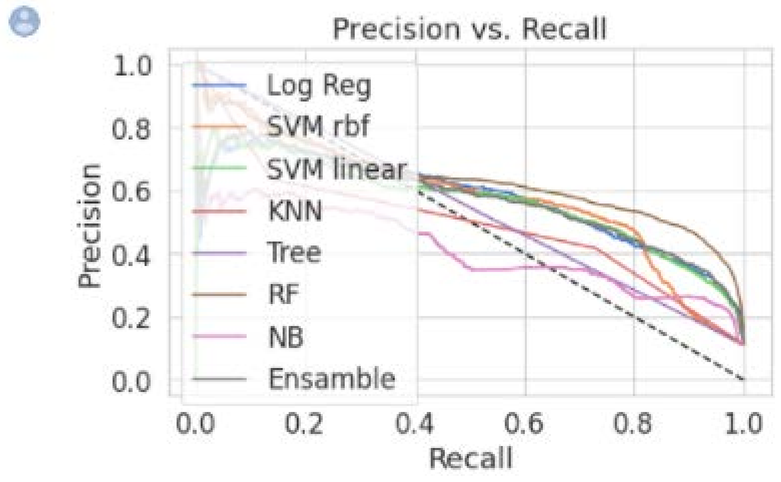
	Accuracy
Random forest	0.911722
Logistic Regression	0.907837
Radial Svm	0.905021
Linear Svm	0.901428
KNN	0.893658
Decision Tree	0.888317
Naive Bayes	0.830048

We see that Random Forest had better accuracy 91% accuracy. Logistic Regrassion90%, Radial Svm accuracy of 90% and Linear Svm similerly, KNN 89%, Decision Tree 88%. Next, , Naïve Bayes 83%. Let us look at the feature importance in random forest.

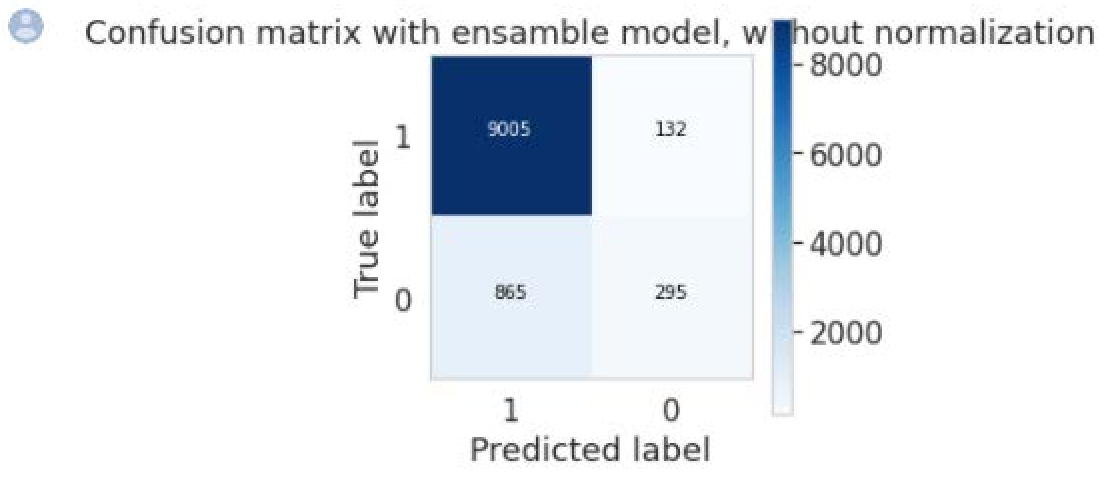
7.2 ROC Curve



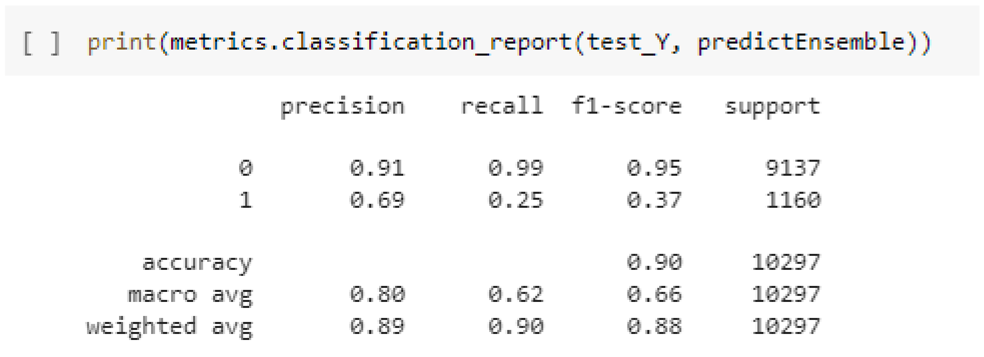
7.3 Precision Recall Curve



7.4 Confusion Matrix



Here 0 is no deposit, and 1 is deposit. In this confusion matrix we can easily see that diposit data predicted itself as 9005, where it is misclassified on 865 data. On the other hand, no deposit data predicted itself as 295 and misclassified on 1132 amounts of data.



```
[ ]  print(metrics.classification_report(test_Y, predictEnsemble))

              precision    recall  f1-score   support

           0       0.91      0.99      0.95      9137
           1       0.69      0.25      0.37      1160

    accuracy                           0.90     10297
   macro avg       0.80      0.62      0.66     10297
weighted avg       0.89      0.90      0.88     10297
```

Looking at the recall scores for 0 (no) and 1 (yes), the model did well for predicting stable (correct 82% of the time) while did not do that well for predicting unstable (correct 93% of the time).

8. Conclusion

As we have mentioned before this is highly Imbalanced dataset. Thats why accuracy of the models were very low. Still there were well enough of accuracy to measure for us. It was a diifficult dataset to work on.But we have managed to gain some confidence by cracking it.

8.1 Limitations:

As the dataset contains millions of data . We were unable to measure the perfect combinations for neural network and tweak the network. we dont have that kind of device that can hold much pressure as we want.

References

1. https://archive.ics.uci.edu/ml/datasets/Bank%2BMarketing

2. https://towardsdatascience.com/machine-learning-case-study-a-data-driven-approach-to-predict-the-success-of-bank-telemarketing-20e37d46c31c