INTRODUCTION TO DATA SCIENCE

ASSIGNMENT 2

Pratibha Tewatia
103127766
Master of IT(Professional Computing)
103127766@student.swin.edu.au
8th April 2021

1.1Abstract

For assignment 2 we were asked to compare two classification models on the basis of various parameters like accuracy, precision, recall and confusion matrix. We were supposed to identify a dataset from UCI based on few conditions provided in the task and then perform some data feature engineering techniques which were taught as part of this subject. Also, we were asked to create visualizations and perform descriptive statistics on various columns. And then after cleaning the data, dividing the dataset into trained and test dataset, and creating a model by selecting any of the two classification algorithms. And the final step was to find out the best working model. I have created this report with an intend to clearly lay out the foundation of how I have formulated each and every step and I have tried my best to convey that in the best possible manner.

2.1 Introduction

As part of first task, I identified a dataset which has information about a marketing campaign that was conducted by a Portuguese banking institution. I have explained in detail about it in the next section. After that I acquired the data and converted it into an excel format. Then, I did data exploration for individual columns, and later I explored relationship between columns in pair for categorical columns. For numerical ones I created a scatter matrix. Then I did some feature engineering for all the relevant categorical columns by using label encoder, binary code and created dummy variables for nominal categorical columns with unique values greater than 3. The dataset had 20 columns in the beginning which got increased to 55 (all numerical columns) after doing columns filtering and transformation.

Then I divided the dataset into X and y where X represents the input columns containing numerical columns and y being the output column. After that I split them into three suites with the respective ratio as mentioned in the task guide. I made use of two algorithms – KNN and Random forest. For both the algorithms I performed hyperparameter tuning. After doing that I passed the training and test data of all the three suites one by one and created models for respective algorithms.

Next step was comparing the efficiency of both the algorithms for three test suites with respect to accuracy, precision, recall, F1 and confusion matrix. For this I created a table and compared all the results. And as part of last step I created visualization for comparison of performance metrics. So let us start with the first task.

3.1Task 1(Problem Formulation, Data Acquisition)

3.1.1 Problem Formulation : For this task I visited the UCI repository and selected a Bank Marketing Data Set which is a classification dataset with a goal to predict if a particular client will subscribe to the term deposit or not. The data is related with direct marketing campaigns of a Portuguese banking institution. The marketing campaigns were based on phone calls. So it was required to contact the same client more than once in order to access if the bank term deposit will be subscribed or not. There were total of 4 datasets in this particular data source. I chose the one which consisted of 10% examples which were randomly selected from the main dataset that contained more than 40k rows. For keeping the dataset consistent and for the ease of data derivation I decided to make of dataset with lesser number of rows. So, this dataset contains a total of 20 input variables and 1 output variable. The output variable is ‘y’ (categorical) which gives the result in form of yes or no if the product will be subscribed or not. There are 10 categorical features and 10 numeric features. Attribute information is provided as below :

1) age : (numeric)

2) job : type of job (categorical: 'admin.','blue-collar','entrepreneur','housemaid','management','retired','self-employed','services','student','technician','unemployed','unknown')

3) marital : marital status (categorical: 'divorced','married','single','unknown'; note: 'divorced' means divorced or widowed)

4) education (categorical: 'basic.4y','basic.6y','basic.9y','high.school','illiterate','professional.course','university.degree','unknown')

5) default: has credit in default? (categorical: 'no','yes','unknown')

6) housing: has housing loan? (categorical: 'no','yes','unknown')

7) loan: has personal loan? (categorical: 'no','yes','unknown')

8) contact : contact communication type (categorical: 'cellular','telephone')

9) month: last contact month of year (categorical: 'jan', 'feb', 'mar', ..., 'nov', 'dec')

10) day_of_week: last contact day of the week (categorical: 'mon','tue','wed','thu','fri')

11) duration: last contact duration, in seconds (numeric)

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 that the client was not previously contacted)

14) previous: number of contacts performed before this campaign and for this client (numeric)

15) poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')

16) emp.var.rate: employment variation rate - quarterly indicator (numeric)

17) cons.price.idx: consumer price index - monthly indicator (numeric)

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

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

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

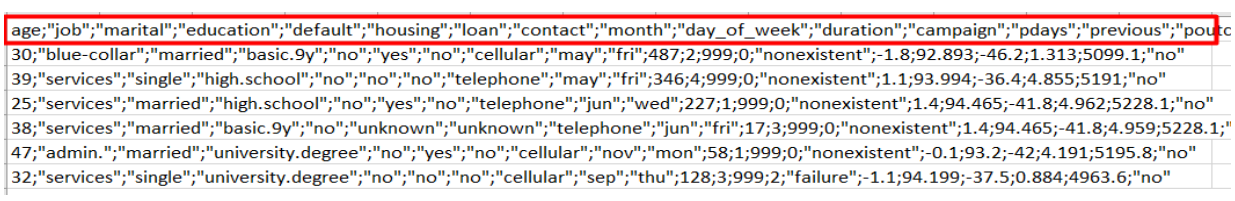
Output variable (desired target):

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

Reason for choosing this dataset :

UCI contains a myriad of datasets hence it becomes very difficult to choose the one which perfectly fits into one's needs. We were asked to search for the dataset that contains atleast 150 rows with atleast 5 columns, one column being a categorical one which should not be the class label. After narrowing down the search I was left with around 37 datasets. I was particularly interested in one dataset which was related to predicting if a user will have a heart disease, but it had 75 columns and it would be very complicated to do data cleaning for every column. Hence, I chose a little less complicated dataset which had only 20 columns. Even though it reached to 55 after doing label encoding, it is a pretty amazing data set. It is a mix of both categorical and numerical columns. I got a chance to put all the knowledge I have gathered so far for creating a model for this data set. In real world, we do not get numerical data. We get raw data in form of text which is generally converted into categories and then this categorical data is converted into numerical ones using label encoders or creating dummy variable. Hence, I used this data set where I could learn how to deal with categorical data in detail (numerical data we have already implemented in the first assignment).

3.1.2 Data Acquisition : After selecting this dataset I downloaded it in my local machine. Since it was in the CSV format, which was not in proper tabular format, I converted it into an excel file using TEXT IMPORT WIZARD. I then uploaded this file into the jupyter notebook folder named "BankStatement.xlsx".



age;"job";"marital";"education";"default";"housing";"loan";"contact";"month";"day_of_week";"duration";"campaign";"pdays";"previous";"poutc
30;"blue-collar";"married";"basic.9y";"no";"yes";"no";"cellular";"may";"fri";487;2;999;0;"nonexistent";-1.8;92.893;-46.2;1.313;5099.1;"no"
39;"services";"single";"high.school";"no";"no";"no";"telephone";"may";"fri";346;4;999;0;"nonexistent";1.1;93.994;-36.4;4.855;5191;"no"
25;"services";"married";"high.school";"no";"yes";"no";"telephone";"jun";"wed";227;1;999;0;"nonexistent";1.4;94.465;-41.8;4.962;5228.1;"no"
38;"services";"married";"basic.9y";"no";"unknown";"unknown";"telephone";"jun";"fri";17;3;999;0;"nonexistent";1.4;94.465;-41.8;4.959;5228.1;"
47;"admin.";"married";"university.degree";"no";"yes";"no";"cellular";"nov";"mon";58;1;999;0;"nonexistent";-0.1;93.2;-42;4.191;5195.8;"no"
32;"services";"single";"university.degree";"no";"no";"no";"cellular";"sep";"thu";128;3;999;2;"failure";-1.1;94.199;-37.5;0.884;4963.6;"no"

before the data conversion to excel

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
25	services	married	high.school	no	yes	no	telephone	jun	wed	227	1	999	0	nonexistent	1.4	94.465	-41.8
36	self-employed	single	basic.4y	no	no	no	cellular	jul	thu	148	1	999	0	nonexistent	1.4	93.918	-42.7
38	technician	married	professional.course	no	yes	no	cellular	aug	mon	479	1	999	0	nonexistent	1.4	93.444	-36.1
34	admin.	married	university.degree	no	no	no	cellular	aug	tue	109	1	999	0	nonexistent	1.4	93.444	-36.1
28	blue-collar	single	basic.9y	no	yes	no	telephone	jun	wed	525	1	999	0	nonexistent	1.4	94.465	-41.8

after the data conversion to excel

3.1Task 2 (Data Exploration)

3.2.1 Exploring each column (i.e., attributes) by using appropriate descriptive statistics and/or graphical visualizations.

(Note : As per the task guide this step should be done after data cleaning, however my dataset is dominated by categorical values (Ordinal and Nominal) which will later be required to be transformed/cleaned/encoded in order to make it fit for the modelling because most of the machine learning algorithms understand numerical data. Though there are a few algorithms that make use of categorical data as well, however, the best approach is the encode the variables. After transformation, dummy columns will be created for Nominal categorical data and making visualizations for dummy columns will not insightful. Hence, doing this part first in order to extract important observations and making use of them later while creating the model)

I have created several visualizations and descriptive statistics which are discussed below.

3.2.1.1 **occupation** : The maximum number of people have been enrolled with admin jobs and there are few jobs which are unknown. People with blue collar and technician jobs are also in high number.


```
data['job'].describe()

count       4119
unique        12
top       admin.
freq        1012
Name: job, dtype: object
```

(Note : Since it is a categorical column, while using the describe() method it returned **the count of the values, unique values, highest value, frequency**. But I was also interested in finding the mean and standard deviation of other numerical columns with respect to the job. Hence, I used the groupby method to calculate the mean and standard deviation and then according to the job level I analyzed the columns. The same has been done for all the 10 columns for which I have done the exploration. This information is particularly useful in knowing the data set thoroughly.)

```
data.groupby(['job']).mean()
```

job	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
admin.	38.240119	261.871542	2.642292	944.025692	0.192688	0.061462	93.527008	-40.014723	3.601909	5165.542885
blue-collar	39.265837	261.852941	2.432127	983.270362	0.147059	0.235520	93.669430	-41.571493	3.742370	5174.265611
entrepreneur	42.202703	249.202703	2.216216	992.263514	0.141892	0.263514	93.614682	-40.839865	3.931493	5180.600000
housemaid	45.672727	229.663636	2.463636	980.909091	0.100000	0.426364	93.658536	-39.324545	4.009182	5179.862727
management	42.429012	246.799383	2.432099	953.015432	0.246914	-0.049691	93.483056	-40.591667	3.553451	5165.971914
retired	60.873494	311.789157	2.397590	897.301205	0.325301	-0.580120	93.487506	-39.034337	2.875741	5126.062048
self-employed	40.679245	254.924528	2.817610	967.691824	0.207547	0.086164	93.575283	-40.323899	3.669296	5166.974843
services	38.513995	232.529262	2.844784	978.773537	0.178117	0.067684	93.626059	-41.532061	3.552776	5165.806870
student	26.695122	287.134146	1.951220	902.426829	0.487805	-1.192683	93.448329	-39.665854	2.165354	5092.046341
technician	38.622287	253.286541	2.509407	964.548480	0.169320	0.248770	93.575912	-39.895948	3.800760	5173.511433
unemployed	39.531532	249.801802	2.621622	927.369369	0.234234	-0.181982	93.568532	-40.505405	3.373216	5153.002703
unknown	46.846154	234.410256	2.333333	947.923077	0.179487	0.305128	93.679744	-38.971795	3.934487	5174.471795

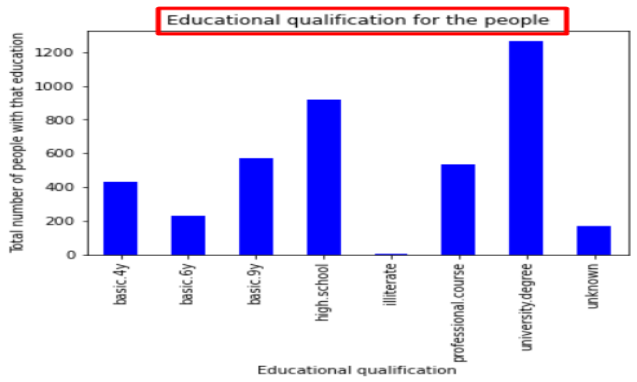
Mean of various columns grouped by job

```
data.groupby(['job']).std()
```

job	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
admin.	8.828773	281.194547	2.762349	227.254380	0.556791	1.609719	0.564158	4.745429	1.759308	76.341192
blue-collar	9.083514	249.304866	2.210068	124.068490	0.416101	1.440046	0.571472	4.100568	1.663464	62.349111
entrepreneur	9.235365	282.236765	2.114223	81.952895	0.387037	1.313361	0.551736	3.966917	1.472563	56.739685
housemaid	11.716145	197.119545	2.325279	133.548984	0.540981	1.515951	0.579479	4.400504	1.594287	75.998330
management	9.295188	208.489644	2.362966	209.035267	0.734482	1.516769	0.572315	4.483795	1.686725	70.142475
retired	9.407393	282.990369	2.277371	301.996558	0.723768	1.815730	0.692803	5.781923	1.943605	95.469753
self-employed	9.946897	265.867211	2.441351	174.302333	0.528617	1.542252	0.548186	4.327122	1.698767	72.547529
services	8.882764	211.917749	3.402255	140.494996	0.483218	1.524678	0.578924	4.191420	1.745116	68.492073
student	4.531615	260.934350	1.562701	295.528154	0.804983	1.652591	0.777807	6.167922	1.771785	89.982944
technician	8.755157	258.004155	2.570369	181.754593	0.506751	1.530741	0.545880	4.577793	1.678652	72.398683
unemployed	9.471315	228.219664	2.611353	258.189604	0.587057	1.661992	0.619620	4.561874	1.810508	85.793369
unknown	11.101300	238.961098	1.752192	222.562042	0.506370	1.709255	0.563961	3.982590	1.697991	70.471483

SD grouped by job

3.2.1.2**Education** : The maximum people have university degree as their highest qualification. The same is shown via this graphical representation. Very few people are illiterate.




```
count                 4119
unique                   8
top       university.degree
freq                  1264
Name: education, dtype: object
```

```
data.groupby(['education']).std()
```

education	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
basic.4y	11.998504	239.910580	2.154453	157.225796	0.443874	1.505288	0.562736	4.470717	1.666075	70.549440
basic.6y	8.304827	258.128285	2.611336	113.858940	0.360548	1.401429	0.567057	3.966704	1.606210	59.152415
basic.9y	9.140190	240.527586	2.326835	136.309935	0.462509	1.455686	0.573845	4.191238	1.655002	63.070573
high.school	9.645409	261.551373	2.954084	197.637103	0.542542	1.577265	0.586147	4.610184	1.764888	73.578428
illiterate	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
professional.course	9.874111	300.646382	2.401494	197.151385	0.511631	1.553704	0.578970	4.702918	1.722437	75.929943
university.degree	9.707178	241.178450	2.581878	219.556469	0.601985	1.621557	0.566992	4.706623	1.763584	78.280306
unknown	12.233660	234.796448	2.396656	235.674130	0.777729	1.676053	0.624201	4.602961	1.878283	84.991680

Standard deviation of all columns grouped by qualification

```
data.groupby(['education']).mean()
```

education	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
basic.4y	47.657343	255.682984	2.421911	973.524476	0.142191	0.291841	93.666193	-40.465501	3.828089	5174.093939
basic.6y	40.144737	259.000000	2.649123	985.881579	0.140351	0.271053	93.689114	-41.539912	3.810469	5176.600000
basic.9y	39.231707	250.538328	2.348432	979.963415	0.158537	0.183275	93.640805	-41.443554	3.722737	5173.182578
high.school	38.097720	258.534202	2.630836	958.022801	0.206298	-0.002497	93.564314	-40.995765	3.511732	5163.212595
illiterate	42.000000	146.000000	4.000000	999.000000	0.000000	-2.900000	92.201000	-31.400000	0.834000	5076.200000
professional.course	40.207477	278.816822	2.512150	958.211215	0.194393	0.163925	93.599630	-40.127664	3.701426	5167.595140
university.degree	39.017405	247.707278	2.583070	947.900316	0.207278	-0.009731	93.499109	-39.830063	3.547132	5163.023180
unknown	42.826347	267.281437	2.538922	939.700599	0.263473	-0.074251	93.637455	-39.487425	3.410174	5151.260479

Mean grouped by qualification

3.2.1.3**Housing** : Based on housing people have been categorized as yes/no/unknown. Maximum people have houses.


```
count      4119
unique        3
top         yes
freq       2175
Name: housing, dtype: object
```

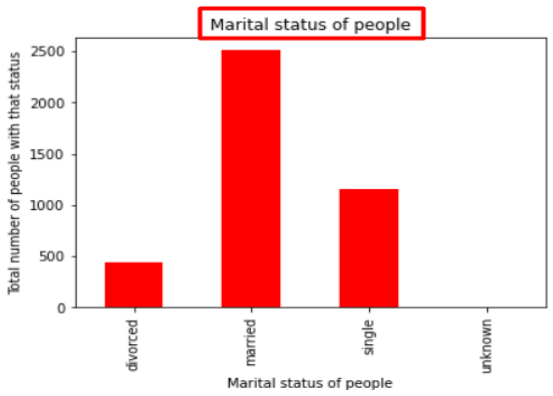
housing	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	40.213159	260.585644	2.588363	964.982599	0.172376	0.176998	93.636321	-40.358728	3.719115	5168.835454
unknown	39.523810	243.923810	2.580952	961.190476	0.285714	0.228571	93.634800	-40.207619	3.743962	5171.364762
yes	40.057931	254.198161	2.491954	956.529195	0.200920	0.000230	93.529174	-40.631862	3.532779	5164.255816

Mean of numerical columns grouped by housing

housing	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	10.348194	263.060251	2.683058	180.664933	0.509089	1.535665	0.576427	4.468868	1.707474	73.253679
unknown	9.117919	209.279565	2.209163	190.902064	0.716754	1.528745	0.550635	4.237531	1.717736	73.736166
yes	10.341072	249.513559	2.483766	200.998526	0.558121	1.583422	0.578747	4.712248	1.752154	73.973839

Standard Deviation of numerical columns grouped by housing

3.2.1.4 **Marital status :** As per marital status people have been categorized as married/single/divorced and unknown. Maximum people from the data set are married. Only 8 unknown entries are present.




```
count      4119
unique        4
top     married
freq       2509
Name: marital, dtype: object
```

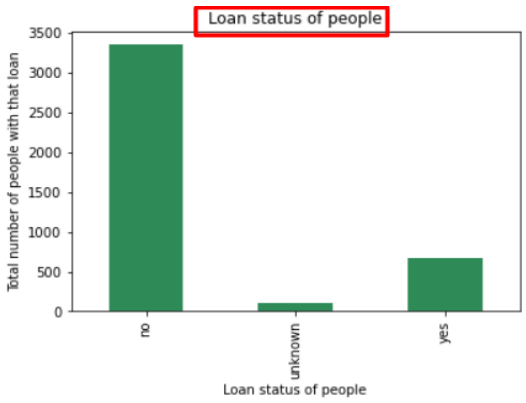
marital	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
divorced	45.107623	263.163677	2.542601	972.255605	0.174888	0.143274	93.600547	-40.621973	3.707177	5170.145964
married	42.389398	256.289757	2.527700	964.162216	0.173774	0.172260	93.599039	-40.357114	3.728554	5170.724472
single	33.209020	254.647875	2.555941	947.338248	0.233304	-0.128448	93.529971	-40.768257	3.352657	5155.756114
unknown	42.272727	336.272727	2.545455	999.000000	0.090909	0.181818	93.537545	-39.690909	3.855182	5174.409091

Mean of various columns with respect to their marital status

marital	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
divorced	10.081336	293.313205	2.358199	161.019148	0.537072	1.518718	0.574055	4.424042	1.691403	70.536890
married	9.993669	251.230116	2.486272	182.766367	0.509180	1.520955	0.570921	4.498845	1.684501	70.369896
single	7.318678	246.753203	2.806961	220.595186	0.608277	1.649639	0.598017	4.852961	1.826598	80.682019
unknown	12.092071	182.948676	3.236159	0.000000	0.301511	1.589225	0.454871	4.266018	1.636611	55.852206

Standard Deviation of various columns with respect to their marital status

3.2.1.5 **Loan Status :** Maximum people do not have any loan on their name.




```
count     4119
unique       3
top         no
freq      3349
Name: loan, dtype: object
```

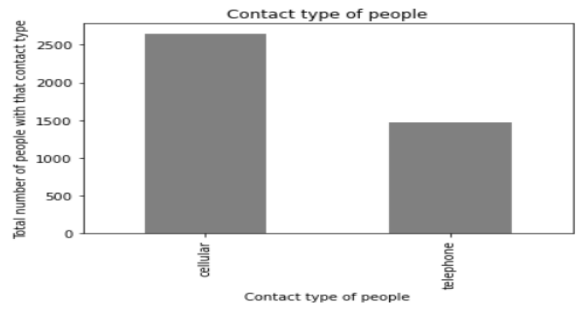
loan	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	40.163332	259.236787	2.564049	959.856972	0.190206	0.081188	93.582737	-40.512213	3.612484	5165.858107
unknown	39.523810	243.923810	2.580952	961.190476	0.285714	0.228571	93.634800	-40.207619	3.743962	5171.364762
yes	39.956391	246.487218	2.395489	963.147368	0.175940	0.081353	93.555732	-40.479098	3.646675	5168.851128

Mean of various columns with respect to their loan status

loan	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	10.324971	256.673702	2.657776	193.268611	0.534876	1.559058	0.579216	4.587464	1.736964	74.242504
unknown	9.117919	209.279565	2.209163	190.902064	0.716754	1.528745	0.550635	4.237531	1.717736	73.736166
yes	10.441441	251.253508	2.120961	185.427890	0.543880	1.589920	0.584303	4.688527	1.720617	70.723724

Standard Deviation of various columns with respect to their marital status

3.2.1.6 **Contact :** This visualization demonstrates that around 2500 people have a cellular mode of communication while over 1500 have telephone.




```
count        4119
unique          2
top      cellular
freq         2652
Name: contact, dtype: object
```

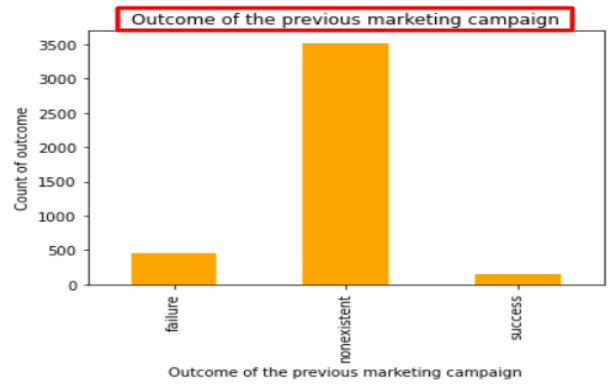
contact	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
cellular	40.016214	262.126697	2.373680	943.199472	0.277149	-0.360709	93.332208	-41.360256	3.118162	5152.155882
telephone	40.289707	247.137014	2.832993	991.556919	0.033401	0.890661	94.027121	-38.942331	4.531015	5192.379482

Mean of various columns with respect to their communication connectivity

contact	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
cellular	10.714010	254.854038	2.343292	228.750046	0.638656	1.640892	0.499067	4.935696	1.814915	80.696456
telephone	9.547536	254.234116	2.908810	85.661673	0.220633	0.992703	0.425799	3.388074	1.095196	49.257442

Standard Deviation of various columns with respect to their communication connectivity

3.2.1.7This figure demonstrates the **outcome of previous marketing campaign.** A very meagre amount of the campaign was successful and most of the data is non-existent which is why it is difficult to predict the outcome with this column. This will not serve as a good input feature.




```
count        4119
unique          2
top      cellular
freq         2652
Name: contact, dtype: object
```

contact	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
cellular	40.016214	262.126697	2.373680	943.199472	0.277149	-0.360709	93.332208	-41.360256	3.118162	5152.155882
telephone	40.289707	247.137014	2.832993	991.556919	0.033401	0.890661	94.027121	-38.942331	4.531015	5192.379482

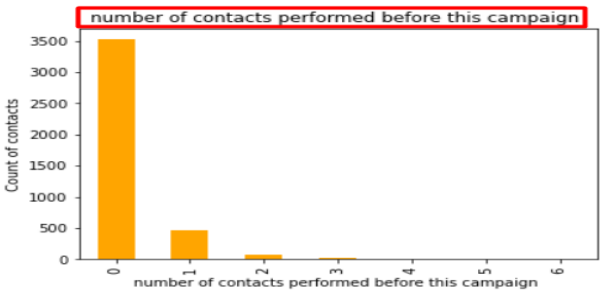
Mean of various columns with respect to their outcome of previous marketing campaign

contact	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
cellular	10.714010	254.854038	2.343292	228.750046	0.638656	1.640892	0.499067	4.935696	1.814915	80.696456
telephone	9.547536	254.234116	2.908810	85.661673	0.220633	0.992703	0.425799	3.388074	1.095196	49.257442

Standard deviation of various columns with respect to their outcome of previous marketing campaign

3.2.1.8 **previous** : This column tells us about the number of contacts performed before this campaign and for this client. As given below the mean value for this numerical column is 0.19 and standard deviation is 0.5. The minimum value of 0 indicates that in certain cases no contact was made with the client and the maximum value for contact made is 6.

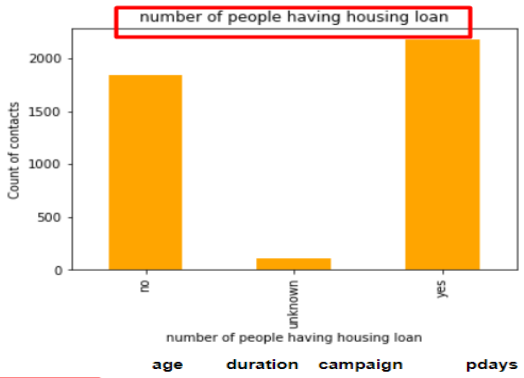
```
count    4119.000000
mean        0.190337
std         0.541788
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         6.000000
Name: previous, dtype: float64
```



3.2.1.9 **age :** The minimum age of people in this dataset is 18 years while the maximum age is 88. The average age is 40 and standard deviation is 10 which is pretty high which means thar the data is more dispersed for this feature.

```
count    4119.000000
mean       40.113620
std        10.313362
min        18.000000
25%        32.000000
50%        38.000000
75%        47.000000
max        88.000000
Name: age, dtype: float64
```

3.2.1.10 **housing :** This displays the number of people having housing loans. Maximum number of people from this dataset have it.




```
count     4119
unique       3
top        yes
freq      2175
Name: housing, dtype: object
```

housing	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	40.213159	260.585644	2.588363	964.982599	0.172376	0.176998	93.636321	-40.358728	3.719115	5168.835454
unknown	39.523810	243.923810	2.580952	961.190476	0.285714	0.228571	93.634800	-40.207619	3.743962	5171.364762
yes	40.057931	254.198161	2.491954	956.529195	0.200920	0.000230	93.529174	-40.631862	3.532779	5164.255816

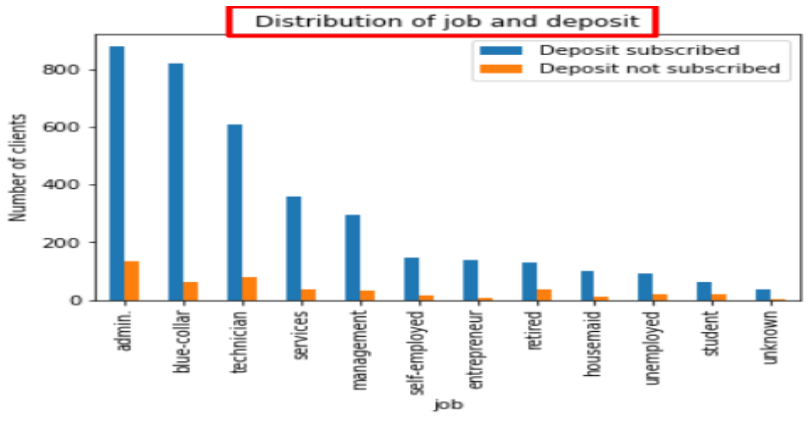
Mean of various columns with respect to their outcome of housing loan

housing	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
no	10.348194	263.060251	2.683058	180.664933	0.509089	1.535665	0.576427	4.468868	1.707474	73.253679
unknown	9.117919	209.279565	2.209163	190.902064	0.716754	1.528745	0.550635	4.237531	1.717736	73.736166
yes	10.341072	249.513559	2.483766	200.998526	0.558121	1.583422	0.578747	4.712248	1.752154	73.973839

Mean of various columns with respect to their outcome of housing loan

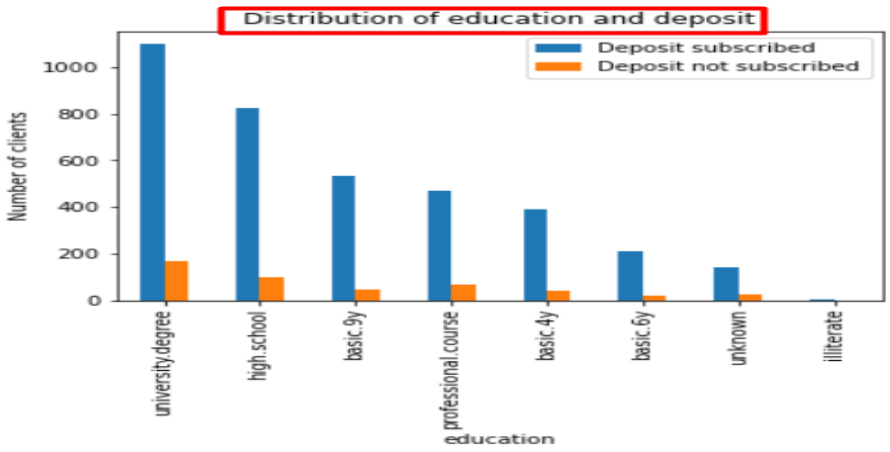
3.2.2 Exploring the relationships between all pairs of columns by using appropriate descriptive statistics and/or graphical visualizations.

3.2.2.1 Finding relationship between job and deposit : It is very important to find key features that play a dominant role in predicting the output of the model. So I started finding some meaningful relationships of class label with categorical features. In this figure job is the input variable and deposit is the output variable. Below is the distribution of people who have a subscription for term deposit of this client. Administrative staff and technical specialists opened the deposit most of all. **data[['job','y']].describe()**



	job	y
count	4119	4119
unique	12	2
top	admin.	no
freq	1012	3668

3.2.2.2 Finding relationship between education and deposit : As per this visualization the university degree holders have maximum subscriptions followed by high schoolers with illiterates having the least count.



data[['education','y']].describe()

	education	y
count	4119	4119
unique	8	2
top	university.degree	no
freq	1264	3668

3.2.2.3 Finding relationship between contact and deposit : This graph indicates that cellular is the best mode of communication.

data[['contact','y']].describe()

	contact	y
count	4119	4119
unique	2	2
top	cellular	no
freq	2652	3668

3.2.2.4 Finding relationship between loan and deposit : This shows that people who have not taken any loans have subscribed the product heavily as compared to the ones who have taken.

data[['loan','y']].describe()

	loan	y
count	4119	4119
unique	3	2
top	no	no
freq	3349	3668

3.2.2.5 Finding relationship between housing and deposit : Home ownership does not have a huge impact on marketing company performance. It is balanced for both the scenarios.

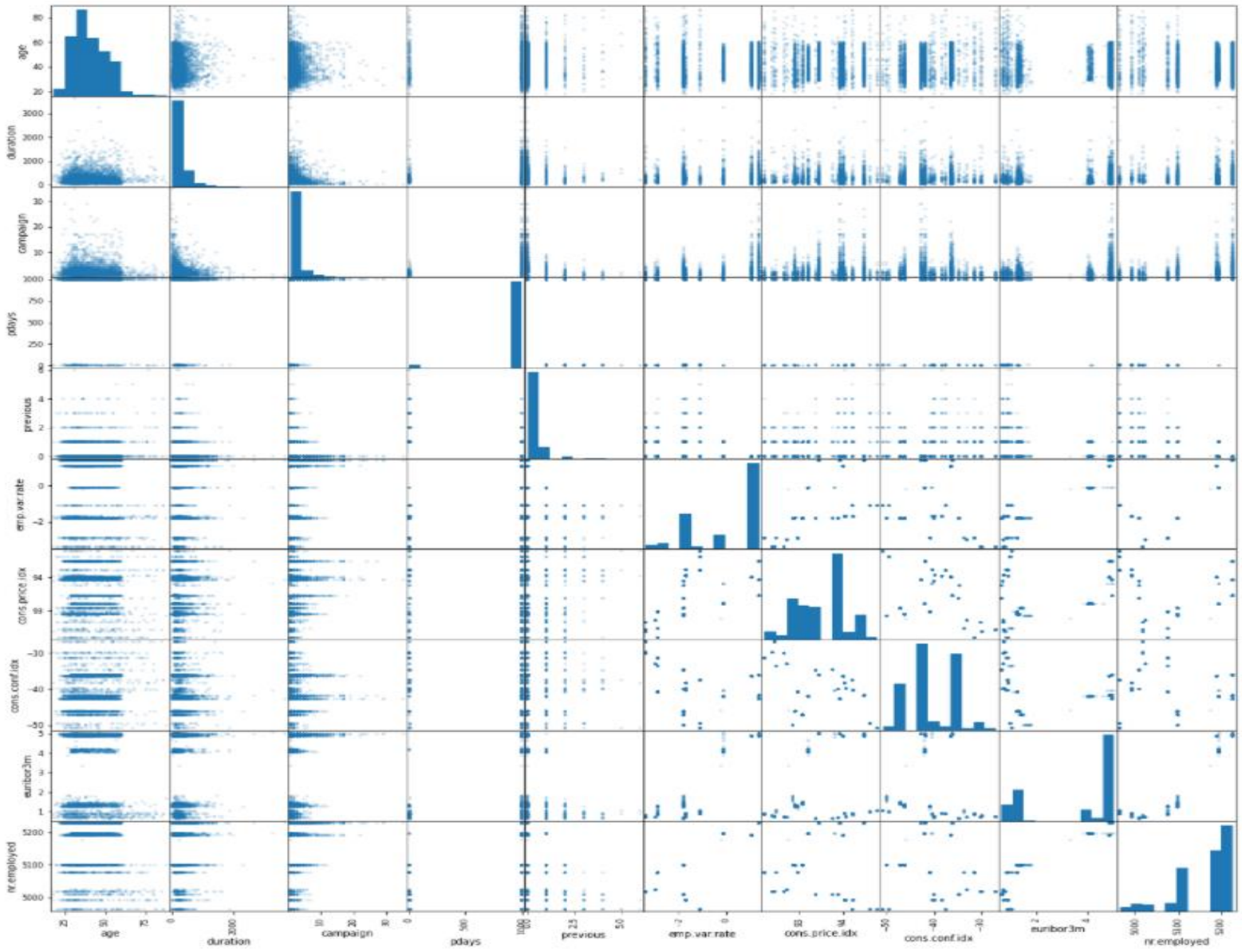
	poutcome	y
count	4119	4119
unique	3	2
top	nonexistent	no
freq	3523	3668

3.2.2.6Exploring relationship between all the numerical columns :

Using data.describe() method I figured out the count, mean, std for all the numerical columns.

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356	5166.481695
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591	73.667904
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000	4963.600000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000	5099.100000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

For further analysis I have created a scatter matrix where we can easily find relationship between all the numerical columns.



3.2.3 Posing one meaningful question and exploring the data by using appropriate methods to find its answer

Question – How can I figure out which numerical features are the most crucial for building a data model?

After exploring the data set with descriptive and statistical analysis I figured that almost all the categorical features are necessary to build up the model as clearly seen in the visualization section. There is dependency on them. After that I was really interested in finding which numerical features are closely correlated with the class label. But creating visualization for each and every feature is tedious task. I wanted something like a scatter matrix but a more refined version which can also provide me with some mathematical output instead of a visualization which is relatively easier to understand. One should always try to find a path that takes minimum effort and gives desirable results which is why I used a correlation matrix to figure out the degree of correlation between numerical columns and class label. To prepare this matrix, I first converted the output categorical variable (Y) into numerical variable by mapping 0 for no and 1 for yes.

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
age	1.000000	0.041299	-0.014169	-0.043425	0.050931	-0.019192	-0.000482	0.098135	-0.015033	-0.041936	0.060374
duration	0.041299	1.000000	-0.085348	-0.046998	0.025724	-0.028848	0.016672	-0.034745	-0.032329	-0.044218	0.418565
campaign	-0.014169	-0.085348	1.000000	0.058742	-0.091490	0.176079	0.145021	0.007882	0.159435	0.161037	-0.076091
pdays	-0.043425	-0.046998	0.058742	1.000000	-0.587941	0.270684	0.058472	-0.092090	0.301478	0.381983	-0.332012
previous	0.050931	0.025724	-0.091490	-0.587941	1.000000	-0.415238	-0.164922	-0.051420	-0.458851	-0.514853	0.255697
emp.var.rate	-0.019192	-0.028848	0.176079	0.270684	-0.415238	1.000000	0.755155	0.195022	0.970308	0.897173	-0.283216
cons.price.idx	-0.000482	0.016672	0.145021	0.058472	-0.164922	0.755155	1.000000	0.045835	0.657159	0.472560	-0.098326
cons.conf.idx	0.098135	-0.034745	0.007882	-0.092090	-0.051420	0.195022	0.045835	1.000000	0.276595	0.107054	0.054393
euribor3m	-0.015033	-0.032329	0.159435	0.301478	-0.458851	0.970308	0.657159	0.276595	1.000000	0.942589	-0.298565
nr.employed	-0.041936	-0.044218	0.161037	0.381983	-0.514853	0.897173	0.472560	0.107054	0.942589	1.000000	-0.349241
y	0.060374	0.418565	-0.076091	-0.332012	0.255697	-0.283216	-0.098326	0.054393	-0.298565	-0.349241	1.000000

The textboxes contain the correlation factor for each of the features with respect to other features. For example, age is correlated with age with a coefficient of 1 which means they are same, higher the coefficient higher the correlation. The most correlated features are **duration and previous**. There are few columns which are highly corelated with each other like euribor3m and nr.employed. Dropping such columns is not advisable as it can impact the performance of the model(the red highlighted ones).

3.1.3 Data Preparation **:**After data exploration the next crucial step is data preparation. This is the most important step after data exploration as the entire model's foundation is dependent on this. This involves cleaning the data, adding missing values, merging, feature engineering and many more. This cleaned data can then be used for further exploratory data analysis. Hence, I have devoted more than 50% of my time doing the cleaning process.

Below are the steps taken to explore the data for data cleaning –

1)Firstly I did some exploring about the dataset, so I used the data.shape command. I found that the number of rows were same as given in excel. It returned (4119, 21)

2)data.head() : Used this command to check the first five rows. It returned 5 rows* 21 columns.

3)data.tail() : Used this command to check the last five rows. . It returned 5 rows* 21 columns.

4)data.describe() : Used this command to understand the data with the help of various parameters like mean, median, SD, min and max values. For all the numerical columns we have this data. Since there are cases where standard deviation is reaching close to 250 or 190, it is necessary to do data transformation using normalization or standardization to make the data uniform across the dataset so that the deviation can be reduced, and a good accuracy rate of model can be achieved.

```
data.describe()
```

	age	duration	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000
mean	40.113620	256.788055	2.537266	960.422190	0.190337	0.084972	93.579704	-40.499102	3.621356	5166.481695
std	10.313362	254.703736	2.568159	191.922786	0.541788	1.563114	0.579349	4.594578	1.733591	73.667904
min	18.000000	0.000000	1.000000	0.000000	0.000000	-3.400000	92.201000	-50.800000	0.635000	4963.600000
25%	32.000000	103.000000	1.000000	999.000000	0.000000	-1.800000	93.075000	-42.700000	1.334000	5099.100000
50%	38.000000	181.000000	2.000000	999.000000	0.000000	1.100000	93.749000	-41.800000	4.857000	5191.000000
75%	47.000000	317.000000	3.000000	999.000000	0.000000	1.400000	93.994000	-36.400000	4.961000	5228.100000
max	88.000000	3643.000000	35.000000	999.000000	6.000000	1.400000	94.767000	-26.900000	5.045000	5228.100000

5)data.isnull().any() : Used this command to know if there are any null values or not. This dataset does not contain any null values which saved the effort of replacing them with equivalent data.

Below are the steps taken as part of data cleaning –

Next step that I followed was converting all the categorical values into numerical ones. Since the majority of corelated columns are categorical and almost 3-4 numerical columns are corelated to the output variable it is necessary to convert the categorical columns into numerical ones to infer better hypothesis. In categorical data, for columns having only two unique values I have used binary code for data conversion by using a simple **replace** method. And in cases where columns have more than two unique values, I have made use of **one hot encoder for creating dummy variables**.

Replacing categorical columns with binary values:

1)contact : For contact there are only two values cellular and telephone as the mode of communication. Hence, I used the following command for data conversion.

data['contact'] = data['contact'].replace(['cellular', 'telephone'],[1,0])

Before conversion

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate
0	25	services	married	high.school	no	yes	no	elephone	jun	wed	...	1	999	0	nonexistent	1.4
1	36	self-employed	single	basic.4y	no	no	no	cellular	jul	thu	...	1	999	0	nonexistent	1.4
2	38	technician	married	professional.course	no	yes	no	cellular	aug	mon	...	1	999	0	nonexistent	1.4
3	34	admin.	married	university.degree	no	no	no	cellular	aug	tue	...	1	999	0	nonexistent	1.4
4	28	blue-collar	single	basic.9y	no	yes	no	elephone	jun	wed	...	1	999	0	nonexistent	1.4

5 rows × 21 columns

After conversion

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	c
0	25	services	married	high.school	0	1	0	0	jun	wed	...	1	0	0	0	1.4	
1	36	self-employed	single	basic.4y	0	0	0	1	jul	thu	...	1	0	0	0	1.4	
2	38	technician	married	professional.course	0	1	0	1	aug	mon	...	1	0	0	0	1.4	
3	34	admin.	married	university.degree	0	0	0	1	aug	tue	...	1	0	0	0	1.4	
4	28	blue-collar	single	basic.9y	0	1	0	0	jun	wed	...	1	0	0	0	1.4	

5 rows × 21 columns

2)Similarly **for loan, housing, default, pdays, previous** I have replaced the values with binary code. There are some unknown values which I have marked as 0 because there is no information about their status and since we are creating a **model** for Portuguese banking institution, which is very crucial for the firm in predicting the output of the subscription, there should not be any false positive values. The model should be made keeping in view the true data and no replacement on your own must be done. The count of these unknown values is anyway in single digits. Hence, there will not be much effect on the performance of the model.

- data['loan'] = data['loan'].replace(['yes', 'unknown', 'no'],[1,0,0])
- data['housing'] = data['housing'].replace(['yes', 'unknown', 'no'],[1,0,0])
- data['default'] = data['default'].replace(['yes', 'unknown', 'no'],[1,0,0])
- data.pdays = data.pdays.replace(999, 0)
  (As mentioned in the dataset that if pdays is 999 it means that the user was never contacted, hence replaced this value with 0)
- data.previous = data.previous.apply(lambda x: 1 if x > 0 else 0).astype('uint8')

3)data.poutcome = data['poutcome'] = data['poutcome'].replace(['nonexistent','failure','success'], [0,0,1])

For poutcome there is majority of data which is non-existent which means it does not exist. For maintaining consistency across this column I replaced it with value 0 because we do not have any information about this.

Before Conversion

default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.e
no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.962	
no	no	no	cellular	jul	thu	...	1	999	0	nonexistent	1.4	93.918	-42.7	4.968	
no	yes	no	cellular	aug	mon	...	1	999	0	nonexistent	1.4	93.444	-36.1	4.965	
no	no	no	cellular	aug	tue	...	1	999	0	nonexistent	1.4	93.444	-36.1	4.963	
no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4	94.465	-41.8	4.864	

After Conversion

default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.empl
0	1	0	0	jun	wed	...	1	0	0	0	1.4	94.465	-41.8	4.962	5:
0	0	0	1	jul	thu	...	1	0	0	0	1.4	93.918	-42.7	4.968	5:
0	1	0	1	aug	mon	...	1	0	0	0	1.4	93.444	-36.1	4.965	5:
0	0	0	1	aug	tue	...	1	0	0	0	1.4	93.444	-36.1	4.963	5:
0	1	0	0	jun	wed	...	1	0	0	0	1.4	94.465	-41.8	4.864	5:

So a total of 6 categorical columns have now been replaced with the binary values. There are still four columns that needs to be converted to the numerical data. Please note that pdays is already numerical and only change I did is changing 999 to 0.

4)After this I created dummy variables for five categorical values namely (job, month, days of week, marital, education) and dropped the actual columns by using one hot encoder.

Before dummy variables creation

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate
0	25	services	married	high.school	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4
1	36	self-employed	single	basic.4y	no	no	no	cellular	jul	thu	...	1	999	0	nonexistent	1.4
2	38	technician	married	professional.course	no	yes	no	cellular	aug	mon	...	1	999	0	nonexistent	1.4
3	34	admin.	married	university.degree	no	no	no	cellular	aug	tue	...	1	999	0	nonexistent	1.4
4	28	blue-collar	single	basic.9y	no	yes	no	telephone	jun	wed	...	1	999	0	nonexistent	1.4

5 rows × 21 columns

After dummy variable creation the dataset looks something like this now with a total of 55 columns.

	age	default	housing	loan	contact	duration	campaign	pdays	previous	poutcome	...	marital_single	marital_unknown	education_basic.4y	education_b
0	25	0	1	0	0	227	1	0	0	0	...	0	0	0	
1	36	0	0	0	1	148	1	0	0	0	...	1	0	1	
2	38	0	1	0	1	479	1	0	0	0	...	0	0	0	
3	34	0	0	0	1	109	1	0	0	0	...	0	0	0	
4	28	0	1	0	0	525	1	0	0	0	...	1	0	0	

5 rows × 55 columns

Most of the data conversion has been done and it looks pretty good but still there are a few features that needs attention.

5)For **duration** column I created some interval values based on the call length. If the duration is longer, and a value falls under that range then it will be assigned a higher priority. Since duration is the most correlated feature, assigning a priority value to it will actually be helpful for modelling. If the duration of a call falls in the range less than 100, I have assigned it the least value which is 1 and if the value falls over the range of 645, I have assigned it the highest value which is 7.

Before categorization

```
data.duration
0        227
1        148
2        479
3        109
4        525
       ...
4114     495
4115     253
4116     122
4117     323
4118     152
Name: duration, Length: 4119, dtype: int64
```

After categorization

	age	marital	education	default	housing	loan	contact	duration	campaign	pdays	...	month_mar	month_may	month_nov	month_
0	25	married	high.school	0	1	0	0	3	1	0	...	0	0	0	
1	36	single	basic.4y	0	0	0	1	2	1	0	...	0	0	0	
2	38	married	professional.course	0	1	0	1	5	1	0	...	0	0	0	
3	34	married	university.degree	0	0	0	1	2	1	0	...	0	0	0	
4	28	single	basic.9y	0	1	0	0	6	1	0	...	0	0	0	

5 rows × 45 columns

6) emp.var.rate column contained negative values; hence I converted all the negative values into positive ones using lambda function. I did the same thing for cons.conf.idx as well for the same reason

```
data['emp.var.rate'] = data['emp.var.rate'].apply(lambda x: x* -1 if x < 0 else x*1)
print(data['emp.var.rate'])
```

```
0       1.4
1       1.4
2       1.4
3       1.4
4       1.4
       ...
4114    3.4
4115    2.9
4116    1.1
4117    3.4
4118    3.4
Name: emp.var.rate, Length: 4119, dtype: float64
```

```
data['cons.conf.idx'] = data['cons.conf.idx'] * -1
print(data['cons.conf.idx'])
```

```
0       41.8
1       42.7
2       36.1
3       36.1
4       41.8
        ...
4114    30.1
4115    31.4
4116    49.5
4117    29.8
4118    30.1
Name: cons.conf.idx, Length: 4119, dtype: float64
```

7) Then I normalized the values of three columns nr.employed, cons.price.idx, cons.conf.idx by using log method of NumPy so as to reduce the variance. It helps to reduce the dimensional values. Also, column euribor3m had values in float, so I converted the entire column into integer using .astype('uint8') method. I have used the describe method in the next section to demonstrate the reduction in standard deviation after performing feature engineering and transformation.

3.3Data Modelling

3.3.1 Splitting the data into a training set and a test set

For this task I first divided my dataset into X and y where X has all the input features and y is the output class label. X has 54 features.

X = data.drop('y', axis='columns')

y= data['y']

I then used the describe methods for both X and y to verify the data. (Please note that standard deviation has highly been reduced after cleaning and transforming the data.)

X.describe()

	age	default	housing	loan	contact	duration	campaign	pdays	previous	poutcome	...	marital_single	r
count	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	4119.000000	...	4119.000000	
mean	40.113620	0.000243	0.528041	0.161447	0.643846	11.652828	2.537266	0.227725	0.144695	0.034474	...	0.279922	
std	10.313362	0.015581	0.499274	0.367987	0.478920	73.548049	2.568159	1.369053	0.351836	0.182466	...	0.449015	
min	18.000000	0.000000	0.000000	0.000000	0.000000	1.000000	1.000000	0.000000	0.000000	0.000000	...	0.000000	
25%	32.000000	0.000000	0.000000	0.000000	0.000000	2.000000	1.000000	0.000000	0.000000	0.000000	...	0.000000	
50%	38.000000	0.000000	1.000000	0.000000	1.000000	2.000000	2.000000	0.000000	0.000000	0.000000	...	0.000000	
75%	47.000000	0.000000	1.000000	0.000000	1.000000	4.000000	3.000000	0.000000	0.000000	0.000000	...	1.000000	
max	88.000000	1.000000	1.000000	1.000000	1.000000	645.000000	35.000000	21.000000	1.000000	1.000000	...	1.000000	

8 rows × 54 columns

y.describe()

```
y.describe()

count    4119.000000
mean        0.109493
std         0.312294
min         0.000000
25%         0.000000
50%         0.000000
75%         0.000000
max         1.000000
Name: y, dtype: float64
```

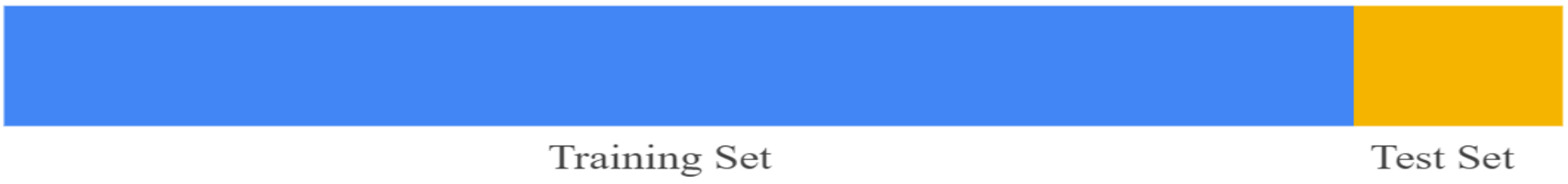
After diving the dataset into X and y, I then used **model_selection** from Sklearn library and imported **train_test_split** method for dividing the data into training and test set.

But what is training and test set??

Training set – It is a subset of the dataset which is used to train a model.

Test set – It is a subset of the dataset which is used to test the trained data.

Given below is a reference image where Training set is 80% and Test set is 20% of the data set.



As given in the task I created three suites and divided the data accordingly into training and test data(0.5 for first test data, 0.4 for second test data and 0.2 for third test data). Below is the command I used .

from sklearn.model_selection import train_test_split

Divided the data into training and test data in three suites using train_test_split.

```
X_train1, X_test1, y_train1, y_test1 = train_test_split(X, y, test_size=0.5, random_state=11)

X_train2, X_test2, y_train2, y_test2 = train_test_split(X, y, test_size=0.4, random_state=11)

X_train3, X_test3, y_train3, y_test3 = train_test_split(X, y, test_size=0.2, random_state=11)
```

The shape of first suite 2059+2060=4119

```
'Shape of splitted train and test sets for first suite'
(2059, 54)
(2059,)
(2060, 54)
(2060,)
```

0.5*4119 ~ 2059(Test Data)

The shape of first suite 2471+1648=4119

```
'Shape of splitted train and test sets for second suite'
(2471, 54)
(2471,)
(1648, 54)
(1648,)
```

0.4*4119 ~ 1648 (Test Data)

The shape of first suite 3295+824=4119

```
'Shape of splitted train and test sets for third suite'
(3295, 54)
(3295,)
(824, 54)
(824,)
```

0.2*4119 ~ 824 (Test Data)

3.3.2 Performing the following steps for each of the two chosen models on each of the above three suites:

3.2.2.1 Identifying the method in the "scikit-learn" package which implements the chosen model

For this dataset I have identified two classification models as :

1) **K-nearest neighbors** – This algorithm falls under the category of supervised algorithm used for mostly classification problems. It is also known as the lazy learner. This algorithm calculates the distance between the data point and the neighbors. The neighbors with the shortest distance are selected. The data point is then assigned to the class with maximum numbers of neighbors. K is a hyper parameter which means it has to be explicitly provided by the data scientist on the basis of its optimal value. If the value of K =2, it means that two neighbors need to be considered.

   Reasons for selecting this algorithm :

- Simple and intuitive
- No training step is required because it learns from the past data and classification is done on the basis of maximum neighbors
- It constantly improves as it is an instance-based algorithm. It improves itself with respect to the new data.
- There is only one hyper parameter.

2) **Random forest –** Random forest algorithm is a very flexible kind of algorithm that produces great results even without hyper-parameter tuning. It is also a supervised learning algorithm which is basically an ensemble(collection) of decision trees. It builds several decision trees and merges them together to form a precise and accurate prediction.

Reasons for selecting this algorithm :

- It eliminates the risk of overfitting
- It is generally very robust with outliers if any
- It is noise resistant and is very stable for large data sets
- It is based on bagging algorithm which uses ensemble learning technique. Many decision trees are created, and output is combined which reduces variance and hence the efficiency is improved.

The methods that make use of these models are as follows:

- K-nearest neighbors
  1) KNeighborsClassifier
  2) cross_val_score
- Random forest
  1) RandomForestClassifier
  2) GridSearchCV

3.2.2.2 Selecting appropriate model parameters and using them to train the model via the above-identified method.

```
X = data.drop('y', axis='columns')

y= data['y']
```

After dummy columns creation for categorical data there were a total of 55 columns, and I decided not to drop any of the columns in the final model creation. Earlier I had removed these 4 columns because they were negatively correlated to the output column ‘y’ (In correlation matrix).

`emp.var.rate` `cons.price.idx` `euribor3m` `nr.employed`

But then just for comparison purpose and doing further analysis, I added them again to the dataset and I found that there was some slight change in the overall accuracy of the model. Hence, I looked at the correlation matrix again to understand why that happened. I then realized that these 5 columns are closely related to each other and with cons.conf.idx as well (**which in turn is positively correlated with the output variable “y”**). I am not sure however, but this could be the most probable reason for this change.

	K- Nearest Neighbours	Random Forest
Suite 1	0.891747572815534	0.9019417475728155
Suite 2	0.8980582524271845	0.9065533980582524
Suite 3	0.8871359223300971	0.8968446601941747

This is the accuracy after dropping the 5 columns

	K- Nearest Neighbours	Random Forest
Suite 1	0.8932038834951457	0.9053398058252428
Suite 2	0.9004854368932039	0.9077669902912622
Suite 3	0.8919902912621359	0.8956310679611651

This is the accuracy after adding back the 5 columns. Even though there is not a very significant shift in the accuracy, but I anyway decided to keep the columns.

3.2.2.3 Evaluating the performances of the model on the training and test sets, respectively, in terms of various factors

In order to evaluate the performances of these models I had to make use of certain Sklearn libraries and made use of following commands -

```
from sklearn.metrics import confusion_matrix

from sklearn.metrics import precision_score

from sklearn.metrics import recall_score

from sklearn.metrics import f1_score

from sklearn.metrics import accuracy_score
```

But before that what exactly is a confusion matrix, accuracy, recall, precision and F1 values and how do we calculate them?

Let us understand what the actual meaning behind these terms is. So let us begin with the confusion matrix. A confusion matrix is basically an intuitive metric to find the accuracy of the classification model which may or may not have more than 2 categories. It is a matrix with True Positive, False negative, False positive, True negative values where **True Positive** and **True Negative** gives the correct information of predicted model with respect to the real data.

TRUE POSITIVE (TP)	FALSE POSTIVE (FP)
FALSE NEGATIVE(FN)	TRUE NEGATIVE(TN)

Accuracy is nothing but TP+TN/TP+TN+FP+FN

Recall is out of all the true data how many were right. TP/TP+FP

Precision is out of all the true predicted data how many were right. TP/TP+TN

F1 score is the harmonic mean of precision and recall

We do not have to calculate these values on our own, but we can use the methods from sklearn.metrics mentioned above. This is the advantage of using such libraries which saves us a lot of effort and time.

So below is the confusion matrix for three suites for both the algorithms. As it is pretty much evident that both TP and TN values are dominating the matrix for the algorithms which suggests the number of correct predictions.

1) Confusion matrix for the set of three suites for both the models:

	K- Nearest Neighbours	**Random Forest**
Suite 1	`array([[1822, 10], [ 210, 18]], dtype=int64)`	`array([[1800, 32], [ 163, 65]], dtype=int64)`
Suite 2	`array([[1458, 9], [ 155, 26]], dtype=int64)`	`array([[1436, 31], [ 121, 60]], dtype=int64)`
Suite 3	`array([[718, 6], [ 83, 17]], dtype=int64)`	`array([[705, 19], [ 67, 33]], dtype=int64)`

Accuracy is higher for Random forest as compared to K-nearest neighbors for every suite even though the difference is not a lot.

2)Classification accuracy

	K- Nearest Neighbours	**Random Forest**
Suite 1	0.8932038834951457	0.9053398058252428
Suite 2	0.9004854368932039	0.9077669902912622
Suite 3	0.8919902912621359	0.8956310679611651

KNN has a higher precision compared to Random forest

3)Precision

	K- Nearest Neighbours	**Random Forest**
Suite 1	0.6428571428571429	0.6701030927835051
Suite 2	0.7428571428571429	0.6593406593406593
Suite 3	0.7391304347826086	0.6346153846153846

Random forest as expected has a higher recall value for all the suites

4)Recall

	K- Nearest Neighbours	**Random Forest**
Suite 1	0.07894736842105263	0.2850877192982456
Suite 2	0.143646408839779	0.3314917127071823
Suite 3	0.17	0.33

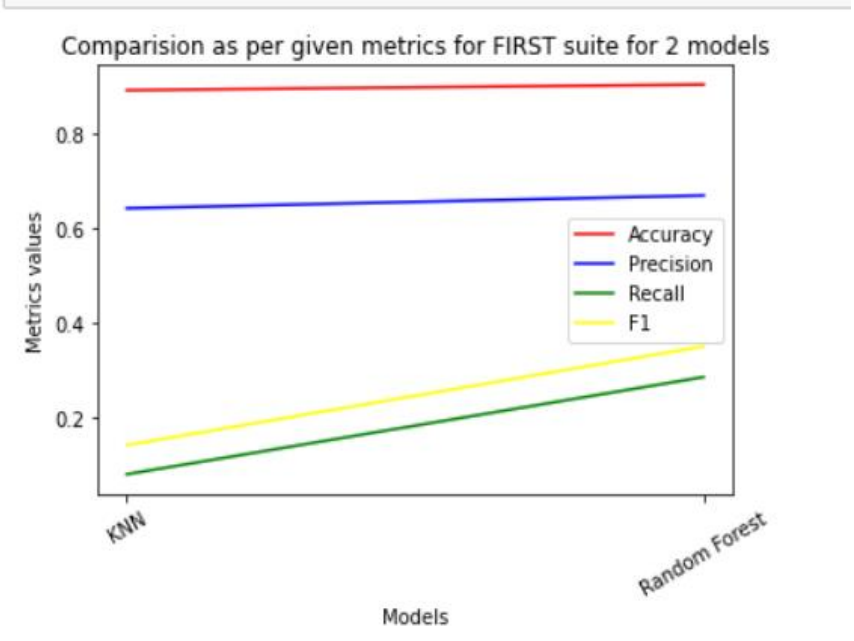
For F1 score as well random forest is dominating

5)F1 score

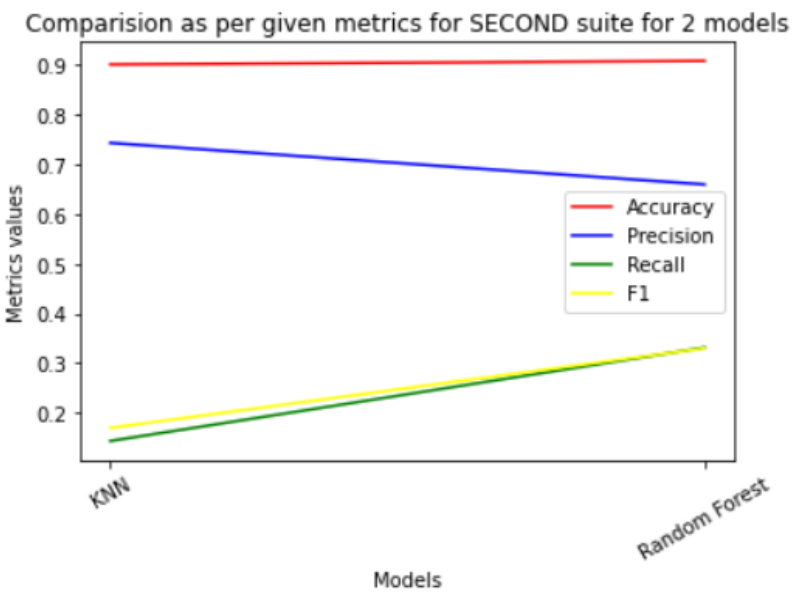
	K- Nearest Neighbours	**Random Forest**
Suite 1	0.140625	0.35031847133757965
Suite 2	0.24074074074074073	0.3706563706563707
Suite 3	0.2764227642276423	0.40268456375838924

3.3.3 Comparing the two chosen models via appropriate graphical visualizations based upon the above evaluation results :

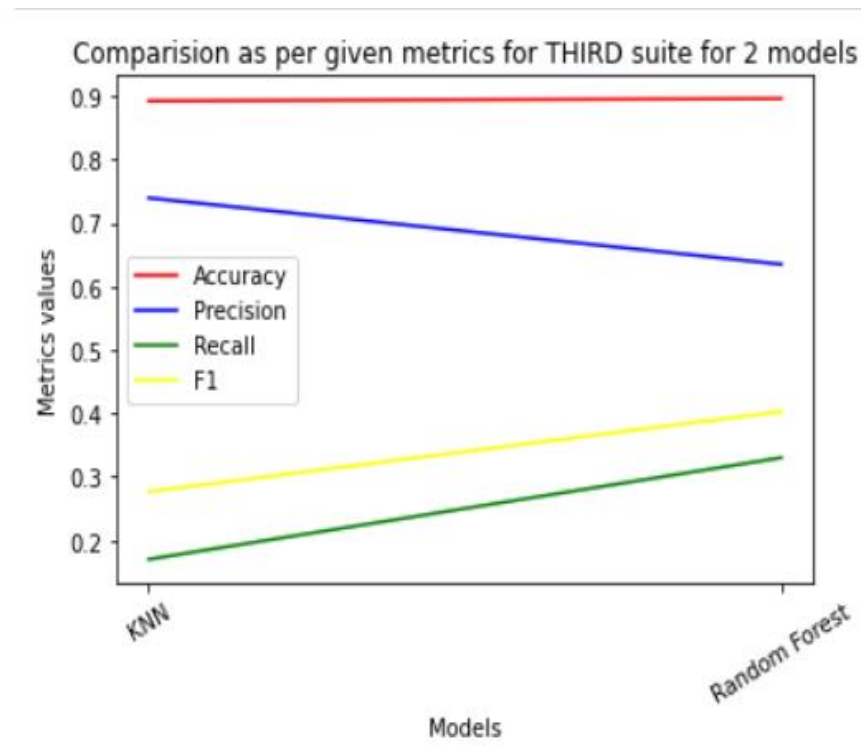
Below is the comparison of both the models with respect to the performance metrics. The legends define the type of metric used for both the models. This graph is created for the first suite where Random Forest is slightly more efficient than KNN and same is the case for other metrics as well.



This visualization is for second suite where again Random Forest is leading the accuracy, recall and F1 metrics however, KNN has higher precision in this case.



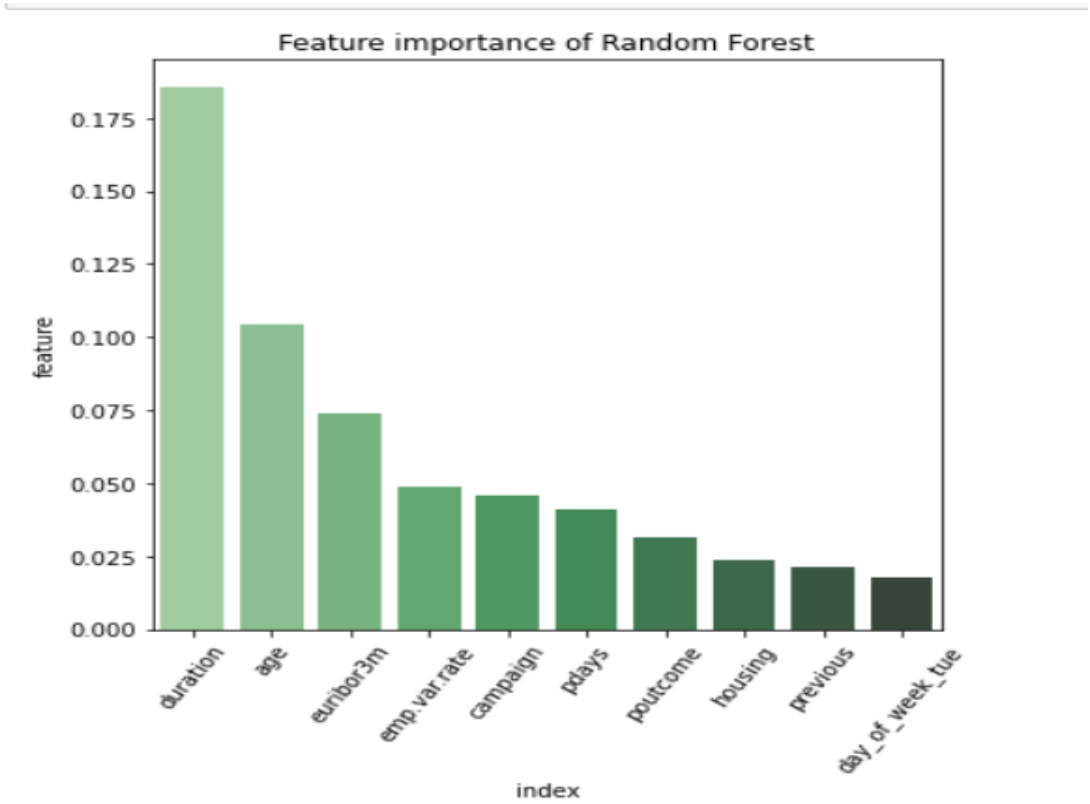
This bar graph represents the comparison for third suite. Same behavior is being witnessed here as well. KNN has higher precision and Random forest has slightly higher accuracy, recall and F1 values.



I have also calculated the **time** taken by each algorithm to predict the output . Clearly Random forest is more efficient and faster.

	K- Nearest Neighbours (sec)	**Random Forest (sec)**
Suite 1	0.253	0.139
Suite 2	0.209	0.152
Suite 3	0.135	0.187

I have created feature importance graph for Random Forest model. This graph represents the significance of each feature with **duration** being the most important feature followed by age and euribor3m. I have used **feature_importances_** method to create this graph. I created this graph only for the first suite as it will be same for all the suites.



4.0Conclusion

In conclusion I would like to say that each and every step associated with model creation is important. Be it data cleaning, be it finding out meaningful relationship among the columns or be it data transformation. Each step has its own significance. After rigorously analyzing and working on this assignment for almost a week I strongly feel that feature engineering can have a very powerful impact on the overall performance of the model provided that the steps which are performed before feature engineering are properly undertaken.

Scaling down the data for a column and syncing it with overall dataset can strongly reduce the standard deviation. It is very crucial to scale down each and every feature. Not only this but transforming the categorical data into numerical ones by using dummy data and one hot encoder can simplify things to next level. I had studied these topics during weekly lectures but after doing this assignment I understood the real mechanism and the essence of these beautiful techniques.

So Random Forest has proved to be a better model than KNN though the leading values differ in fractions only. KNN gave a good competition to Random Forest. However, a winner is a winner.

I guess I have pretty much explained everything in detail with respect to the tasks. Like mentioned in the task guide I have attached a readme.txt file, assignment2.py and the data file BankStatement.xlsx along with this report as a zip folder and uploaded in canvas.

I would hereby like to conclude this report by saying that Data Science is an amazing field where a learner can transform various real-life problems or situations to provide insightful and meaningful solutions. I thoroughly enjoyed working on this assignment and I learnt so many new things which have further elevated my interest and enthusiasm for this subject. Also, I would like to express my immense gratitude to the mentors who have whole heartedly extended their support and made this learning process a very exciting one!

5.0References

https://www.geeksforgeeks.org/plotting-correlation-matrix-using-python/

https://www.displayr.com/what-are-dummy-variables/

https://stackoverflow.com/questions/64314899/how-does-numpy-astypenp-uint8-convert-a-float-array-1-2997805-became-255

https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OneHotEncoder.html

https://machinelearningmastery.com/calculate-feature-importance-with-python/

https://aihubprojects.com/performance-metrics-in-machine-learning-model/