First Insights

This notebook is the first attempt towards data exploration and understanding. The data is loaded in a pandas dataframe format for efficient handeling. This part mostly contains mathematical analysis and ideas about handeling categorical features. The visualisation based analysis is displayed separately in visualisations.ipynb file. All the findings and intutions is documented below as and when required

The first cell contains the dependencies which we will be using in this notebook

In [1]:

```
import numpy as np #Matrix-Maths
import pandas as pd #DataFrame operations
```

The bank-additional-full.csv file instructed to use is loaded and the first five entries is shown below

In [2]:

```
data = pd.read_csv('../Data/bank-additional-full.csv', delimiter=';')
```

In [3]:

```
data.head()
```

Out[3]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_w
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	n
1	57	services	married	high.school	unknown	no	no	telephone	may	n
2	37	services	married	high.school	no	yes	no	telephone	may	n
3	40	admin.	married	basic.6y	no	no	no	telephone	may	n
4	56	services	married	high.school	no	no	yes	telephone	may	n

5 rows × 21 columns

The data contains 21 columns and their name is displayed below. First 20 columns are the input variables and the last column `y` is output label which shows whether a term-deposit was bought by the customer or not as a binary yes/no value

In [4]:

```
cols = list(data.columns)
cols
```

Out[4]:

```
['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']
```

The table below displays various mathematical parameters on numeric type columns in the given data. This gives a good mental picture of how the data is distributed and the range of values it contain

In [5]:

```
data.describe()
```

Out[5]:

	age	duration	campaign	pdays	previous	emp.var.rate	(
count	41188.00000	41188.000000	41188.000000	41188.000000	41188.000000	41188.000000	
mean	40.02406	258.285010	2.567593	962.475454	0.172963	0.081886	
std	10.42125	259.279249	2.770014	186.910907	0.494901	1.570960	
min	17.00000	0.000000	1.000000	0.000000	0.000000	-3.400000	
25%	32.00000	102.000000	1.000000	999.000000	0.000000	-1.800000	
50%	38.00000	180.000000	2.000000	999.000000	0.000000	1.100000	
75%	47.00000	319.000000	3.000000	999.000000	0.000000	1.400000	
max	98.00000	4918.000000	56.000000	999.000000	7.000000	1.400000	

It is important to know what fraction of customer actually buy the term-deposit. The next cell gives the percentage of yes and no in the dataset

In [6]:

```
count_yes = len(data[data['y']=='yes'])
count_no = len(data[data['y']=='no'])
count_unknown = len(data[data['y']=='unknown'])
print('Count of Yes: ', count_yes)
print('Count of No: ', count_no)
print('Count of Unknown: ', count_unknown)
print('Sum of both the counts: ', count_yes+count_no)
print('Total number of datapoints: ', len(data['y']))
```

```
Count of Yes:  4640
Count of No:  36548
Count of Unknown:  0
Sum of both the counts:  41188
Total number of datapoints:  41188
```

In [7]:

```
perc_y = count_yes/len(data['y'])
perc_n = count_no/len(data['y'])
print('Percent of Yes: ', perc_y)
print('Percent of No: ', perc_n)
```

```
Percent of Yes:  0.11265417111780131
Percent of No:  0.8873458288821987
```

The above cells shows that we have 11.2% yes values and 88.7% no values. There is no missing values in responses. The data is highly biased towards no and thus we need to take proper precaution while building a classification system. Some possible options in hand are Resampling, precision-recall, F1 score, weighted crossentropy and graidient boosting. This problem will be dealt in much greater detail after Exploration and Visualisations of data

Let us now see number of distinct values in each of the columns. This will give us ideas about handeling categorical features efficiently

In [8]:

```
for i in cols:
    print('{0}: {1}'.format(i, len(list(set(data[i])))))
```

```
age: 78
job: 12
marital: 4
education: 8
default: 3
housing: 3
loan: 3
contact: 2
month: 10
day_of_week: 5
duration: 1544
campaign: 42
pdays: 27
previous: 8
poutcome: 3
emp.var.rate: 10
cons.price.idx: 26
cons.conf.idx: 26
euribor3m: 316
nr.employed: 11
y: 2
```

The above analysis shows that most of the features are categorical with certain number of distinct features

The cell below displays the frequency of occurances of different types of categorical values

In [9]:

```
for i in cols:
    if(len(list(set(data[i])))<15):
        print(i.upper())
        print(data[i].value_counts())
        print('\n')
```

```
JOB
admin.           10422
blue-collar       9254
technician        6743
services          3969
management        2924
retired           1720
entrepreneur      1456
self-employed     1421
housemaid         1060
unemployed        1014
student            875
unknown            330
Name: job, dtype: int64
```

```
MARITAL
married     24928
single      11568
divorced     4612
unknown        80
Name: marital, dtype: int64


EDUCATION
university.degree      12168
high.school             9515
basic.9y                6045
professional.course     5243
basic.4y                4176
basic.6y                2292
unknown                 1731
illiterate                18
Name: education, dtype: int64


DEFAULT
no         32588
unknown     8597
yes            3
Name: default, dtype: int64


HOUSING
yes        21576
no         18622
unknown      990
Name: housing, dtype: int64


LOAN
no         33950
yes         6248
unknown      990
Name: loan, dtype: int64


CONTACT
cellular     26144
telephone    15044
Name: contact, dtype: int64


MONTH
may    13769
jul     7174
aug     6178
jun     5318
nov     4101
apr     2632
oct      718
sep      570
mar      546
```

```
dec       182
Name: month, dtype: int64


DAY_OF_WEEK
thu    8623
mon    8514
wed    8134
tue    8090
fri    7827
Name: day_of_week, dtype: int64


PREVIOUS
0    35563
1     4561
2      754
3      216
4       70
5       18
6        5
7        1
Name: previous, dtype: int64


POUTCOME
nonexistent    35563
failure         4252
success         1373
Name: poutcome, dtype: int64


EMP.VAR.RATE
 1.4    16234
-1.8     9184
 1.1     7763
-0.1     3683
-2.9     1663
-3.4     1071
-1.7      773
-1.1      635
-3.0      172
-0.2       10
Name: emp.var.rate, dtype: int64


NR.EMPLOYED
5228.1    16234
5099.1     8534
5191.0     7763
5195.8     3683
5076.2     1663
5017.5     1071
4991.6      773
5008.7      650
4963.6      635
5023.5      172
```

```
5176.3       10
Name: nr.employed, dtype: int64


y
no     36548
yes     4640
Name: y, dtype: int64
```

The observations above displays a fairly varied distribution of data we have. This can be analysed only after finding the relation of each with the responses. This is done below

It is also important to know the number of unknowns and NaN values we have in each column. This is done below

In [10]:

```
data.isnull().sum(axis=0)
```

Out[10]:

```
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
```

The data contains no NaN values and thus handeling of missing data is not required in this case

The analysis done below seem to be more prevalant with visualisations. For now, mathematical conclusions are drawn by obseving the data and then data visualisation will be used in the next file to produce more concrete results

In [11]:

```
set_age = list(set(data['age']))
for i in set_age:
    c=0
    for j in range(len(data['age'])):
        if(data['age'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
17: 2
18: 12
19: 20
20: 23
21: 29
22: 36
23: 48
24: 86
25: 93
26: 122
27: 114
28: 151
29: 186
30: 202
31: 220
32: 184
33: 210
34: 184
35: 167
36: 154
37: 137
38: 143
39: 114
40: 84
41: 113
42: 91
43: 88
44: 77
45: 92
46: 79
47: 58
48: 97
49: 55
50: 87
51: 72
52: 81
53: 68
54: 64
55: 56
56: 80
57: 62
58: 58
59: 69
60: 58
61: 32
62: 25
63: 17
64: 27
65: 23
66: 29
```

```
67: 11
68: 15
69: 14
70: 19
71: 21
72: 13
73: 13
74: 15
75: 11
76: 18
77: 13
78: 14
79: 7
80: 18
81: 8
82: 11
83: 8
84: 3
85: 7
86: 5
87: 1
88: 9
89: 2
91: 0
92: 3
94: 0
95: 0
98: 2
```

We see that the buyers mostly belong to age group 23-60 which is the working age group. Subjects with less than 23 yrs of age cannot afford term-deposit and subjects with age greater than 60 also do not want to purchase term deposit with long maturity period. Thus, there is a distinct conclusion that the target must be made on working class people with relatively young age

In [12]:

```
set_job = list(set(data['job']))
for i in set_job:
    c=0
    for j in range(len(data['job'])):
        if(data['job'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
services: 323
management: 328
unknown: 37
entrepreneur: 124
student: 275
technician: 730
admin.: 1352
retired: 434
blue-collar: 638
self-employed: 149
unemployed: 144
housemaid: 106
```

Here, we see students, self-employed, entrepreuners, unemployed, housemaid and unknowns seems to be broke and cannot afford to buy a term-deposit. On the other hand, technician, blue-collar, management, services, retired and admin have sufficient funds to invest in term-deposit. Again, the conclusion is very distinct here that it is more useful to target subjects that have money to invest these kinds of scheme and products. Already broke people have little chances to buy the product

In [13]:

```
set_marital = list(set(data['marital']))
for i in set_marital:
    c=0
    for j in range(len(data['marital'])):
        if(data['marital'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
unknown: 12
single: 1620
married: 2532
divorced: 476
```

Divorced and unknowns seem to have less positive response compared to singles and married. They have a positive aura around that might drive them to buy the product. Also, there might be cases where their future plans and savings might be a major driving factor in decision making

In [14]:

```
set_default = list(set(data['default']))
for i in set_default:
    c=0
    for j in range(len(data['default'])):
        if(data['default'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
no: 4197
yes: 0
unknown: 443
```

Again we got a very distinctive conclusion where anyone who is a loan defaulter might be broke and thus there is no chance of they buying a term deposit

In [15]:

```
set_loan = list(set(data['loan']))
for i in set_loan:
    c=0
    for j in range(len(data['loan'])):
        if(data['loan'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
no: 3850
yes: 683
unknown: 107
```

People who are not under any kind of loan repayment burden are more likely to buy term insurance and thus those group of people should be targeted more

In [16]:

```
set_housing = list(set(data['housing']))
for i in set_housing:
    c=0
    for j in range(len(data['housing'])):
        if(data['housing'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
no: 2026
yes: 2507
unknown: 107
```

There is not much difference between buyer of term-deposit from either class. Thus housing is not a very distinctive parameter

In [17]:

```
set_edu = list(set(data['education']))
for i in set_edu:
    c=0
    for j in range(len(data['education'])):
        if(data['education'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
basic.4y: 428
university.degree: 1670
high.school: 1031
unknown: 251
professional.course: 595
basic.9y: 473
basic.6y: 188
illiterate: 4
```

Now, analysing the effect of Month and Day of Week

In [18]:

```
set_month = list(set(data['month']))
for i in set_month:
    c=0
    for j in range(len(data['month'])):
        if(data['month'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
jun: 559
jul: 649
may: 886
nov: 416
oct: 315
apr: 539
aug: 655
sep: 256
mar: 276
dec: 89
```

In [30]:

```
set_dow = list(set(data['day_of_week']))
for i in set_dow:
    c=0
    for j in range(len(data['day_of_week'])):
        if(data['day_of_week'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
fri: 676
wed: 809
thu: 879
tue: 789
mon: 706
```

We see Day of Week has little effect on term-deposit sell. In contrast, Month shows large variation with sell in May almost four times the sell in March and more than 8 times the sell in Dec. This gives us a clear hint and indication that the sell remain moderate in year start, peaks in the middle and drops significantly towards year end

A situation of doubt is in keeping unknown values encountered since these values doesn't seem to add any information at all, it might suffice to remove such rows

In [19]:

```
for i in ['job', 'marital', 'education', 'default', 'housing', 'loan']:
    data.drop(data.loc[data[i]=='unknown'].index, inplace=True)
```

In [20]:

```
data = data.reset_index(drop=True)
```

In [21]:

```
count_yes = len(data[data['y']=='yes'])
count_no = len(data[data['y']=='no'])
count_unknown = len(data[data['y']=='unknown'])
print('Count of Yes: ', count_yes)
print('Count of No: ', count_no)
print('Count of Unknown: ', count_unknown)
print('Sum of both the counts: ', count_yes+count_no)
print('Total number of datapoints: ', len(data['y']))

perc_y = count_yes/len(data['y'])
perc_n = count_no/len(data['y'])
print('Percent of Yes: ', perc_y)
print('Percent of No: ', perc_n)
```

```
Count of Yes:  3859
Count of No:  26629
Count of Unknown:  0
Sum of both the counts:  30488
Total number of datapoints:  30488
Percent of Yes:  0.1265743899239045
Percent of No:  0.8734256100760955
```

Removing unknowns form the data increases the percentage of positive responses and thus reduced the bias with a small fraction. It largly reduced the noise in dataset

Now, let us explore a specific feature of interest: duration

Duration specifies the last contact duration, in seconds. It is expected by intution that if a call lasted for longer duration, the subject might be well interested and likely to take the term-deposit. On the other hand, if the person has never been contacted before, chances are that he doesn't know about the product and almost never buy it

In [22]:

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

Out[22]:

```
count    30488.000000
mean       259.484092
std        261.714262
min          0.000000
25%        103.000000
50%        181.000000
75%        321.000000
max       4918.000000
Name: duration, dtype: float64
```

In [23]:

```
set_dur = [10*x for x in range(50)]
for i in set_dur:
```

```python
    c=0
    for j in range(len(data['duration'])):
        if(data['duration'][j]>=i and data['duration'][j]<i+1 and data['y'][j]=='
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
0: 0
10: 0
20: 0
30: 0
40: 0
50: 0
60: 0
70: 0
80: 1
90: 2
100: 3
110: 4
120: 2
130: 6
140: 6
150: 9
160: 8
170: 7
180: 6
190: 6
200: 13
210: 11
220: 5
230: 6
240: 3
250: 8
260: 8
270: 9
280: 3
290: 6
300: 8
310: 6
320: 3
330: 5
340: 3
350: 7
360: 5
370: 5
380: 2
390: 3
400: 6
410: 4
420: 2
430: 4
440: 1
450: 3
460: 8
470: 3
480: 2
490: 2
```

The analysis of call duration clearly indicates that short duration calls almost always produce negative response and long duration calls with 1 standard deviation deviation from mean produces positive result. But the goal of this analysis and predictive model is to create a system that can tell the response to be positive or negative before the call takes place. Thus it is better to remove this feature as it doesnot add any information to the predictive model

In [26]:

```
data = data.drop('duration', axis=1)
```

Now we have a lot of insights and understanding about the data. The only thing left to analyse is campain parameteres and socio-economic attributes. Lets get done with that

Analyzing Campaign: number of contacts performed during this campaign and for this client

In [31]:

```
set_camp = list(set(data['campaign']))
for i in set_camp:
    c=0
    for j in range(len(data['campaign'])):
        if(data['campaign'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
1: 1920
2: 1020
3: 477
4: 200
5: 101
6: 53
7: 30
8: 15
9: 14
10: 10
11: 10
12: 2
13: 1
14: 1
15: 0
16: 0
17: 4
18: 0
19: 0
20: 0
21: 0
22: 0
23: 1
24: 0
25: 0
26: 0
27: 0
28: 0
29: 0
30: 0
31: 0
32: 0
33: 0
34: 0
35: 0
37: 0
39: 0
40: 0
41: 0
42: 0
43: 0
```

We observe an interesting pattern here. Most of the subjects who bought the term-deposit made it clear in first few contacts and the number of positive responses fell drastically as the number of contacts increased. Thus we can safely conclude that it is a waste of cost, time and effort to contact the same client multiple times and it is more benefitial to target new customers

Analysing pdays: number of days that passed by after the client was last contacted from a previous campaign

In [32]:

```
set_pdays = list(set(data['pdays']))
for i in set_pdays:
    c=0
    for j in range(len(data['pdays'])):
        if(data['pdays'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
0: 10
1: 7
2: 31
3: 259
4: 53
5: 27
6: 255
7: 34
8: 8
9: 27
10: 25
11: 13
12: 22
13: 26
14: 10
15: 16
16: 5
17: 1
18: 2
19: 1
21: 2
22: 2
25: 1
26: 1
27: 1
999: 3020
```

Here, for the subjects already contacted before, the number of sells is max after 2-7 days since last contact. This might be inferred as the time a customer to rethink about a deal and make a deal. As the number of days passed increases after 10 days, the client is less likely to buy the product and thus can be contacted and urged again

Analyzing previous: number of contacts performed before this campaign and for this client

In [33]:

```
set_prev = list(set(data['previous']))
for i in set_prev:
    c=0
    for j in range(len(data['previous'])):
        if(data['previous'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
0: 2572
1: 832
2: 297
3: 115
4: 30
5: 11
6: 2
7: 0
```

It is clearly visible that new customers tend to buy the term-deposit more compared to the old customers. Hence the next campaign should focus on contacting new customers more to increase its sell compared to urging same old customers multiple times

Analysing outcome of previous campaign and its relation to this campaigns outcome

In [34]:

```
set_pout = list(set(data['poutcome']))
for i in set_pout:
    c=0
    for j in range(len(data['poutcome'])):
        if(data['poutcome'][j]==i and data['y'][j]=='yes'):
            c+=1
    print('{0}: {1}'.format(i, c))
```

```
nonexistent: 2572
failure: 508
success: 779
```

In [36]:

```
data_corr = data.corr(method='pearson')
data_corr
```

Out[36]:

	age	campaign	pdays	previous	emp.var.rate	cons.price.idx	cons.c
age	1.000000	-0.002364	-0.050891	0.049231	-0.050409	-0.035762	0.
campaign	-0.002364	1.000000	0.054312	-0.080766	0.157739	0.127260	-0.
pdays	-0.050891	0.054312	1.000000	-0.590248	0.268763	0.068010	-0.
previous	0.049231	-0.080766	-0.590248	1.000000	-0.403502	-0.176775	-0.
emp.var.rate	-0.050409	0.157739	0.268763	-0.403502	1.000000	0.766055	0.
cons.price.idx	-0.035762	0.127260	0.068010	-0.176775	0.766055	1.000000	0.
cons.conf.idx	0.125017	-0.011664	-0.102368	-0.027930	0.157593	0.027217	1.
euribor3m	-0.036481	0.140836	0.295188	-0.438863	0.969412	0.667292	0.
nr.employed	-0.064586	0.148069	0.370845	-0.488365	0.900390	0.488871	0.

Lastly, The Socio-Economic Factors tend to affect all the groups in masses and does not have a specific implication on a particular group. The correlation matrix above is useful for this analysis but the graphical methods will further help in exploration process

The next set of experimentations I did before creating a predictive model was use Visualisations and analyse the variation. This is present in the next notebook

In [ ]:

```

```