Project: Bank Marketing (Campaign)

Week 9: Deliverables

Name: Archana Devi Ramesh

Email: archanadevi4688@gmail.com

Country: Canada

Batch Code: LISUM16

Specialization: Data Science

Submission Date: 2nd February 2023

Submitted to: Data Glacier

(Individual project)

Table of Contents

1. Problem Description

ABC Bank wants to sell it's term deposit product to customers and before launching the product they want to develop a model which help them in understanding whether a particular customer will buy their product or not (based on customer's past interaction with bank or other Financial Institution). This is an application of the organization’s marketing data.

2. Business Understanding

In predicting the results of marketing campaign for each customer and interpreting which all features affect the results, will help the organization understand how to make campaign more efficient. Moreover, in categorizing which segment of customers subscribed the term deposit, helps to identify who is more likely to buy the product in future thereby developing more targeted marketing campaigns. This can be achieved using ML model that shortlists the customer whose chance of buying the product is more so that their marketing channel (tele marketing, SMS/email marketing etc) can focus only to those customers. This will save resource and their time.

3. Data understanding

- Types of data

The dataset contains 20 independent variables and 1 dependent variable. The independent variables are the following:

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.
12 - campaign: number of contacts performed during this campaign and for this client (numeric, includes last contact)
13 - pdays: number of days that passed by after the client was last contacted from a previous campaign (numeric; 999 means client was not previously contacted)
14 - previous: number of contacts performed before this campaign and for this client (numeric)
15 - poutcome: outcome of the previous marketing campaign (categorical: 'failure','nonexistent','success')
social and economic context attributes
16 - emp.var.rate: employment variation rate - quarterly indicator (numeric)
17 - cons.price.idx: consumer price index - monthly indicator (numeric)
18 - cons.conf.idx: consumer confidence index - monthly indicator (numeric)
19 - euribor3m: euribor 3 month rate - daily indicator (numeric)
20 - nr.employed: number of employees - quarterly indicator (numeric)
Output variable (desired target):
21 - y - has the client subscribed a term deposit? (binary: 'yes','no')

The independent variables are a mix of categorical and numerical values.

As instructed in the dataset, the **duration** feature is dropped from the dataset.

- Problems in the data

  The dataset contains a total of 41188 rows, out of which there were a total of 12 duplicate rows. The pandas **drop_duplicates** method is used to drop the duplicate rows.

  Missing Values:

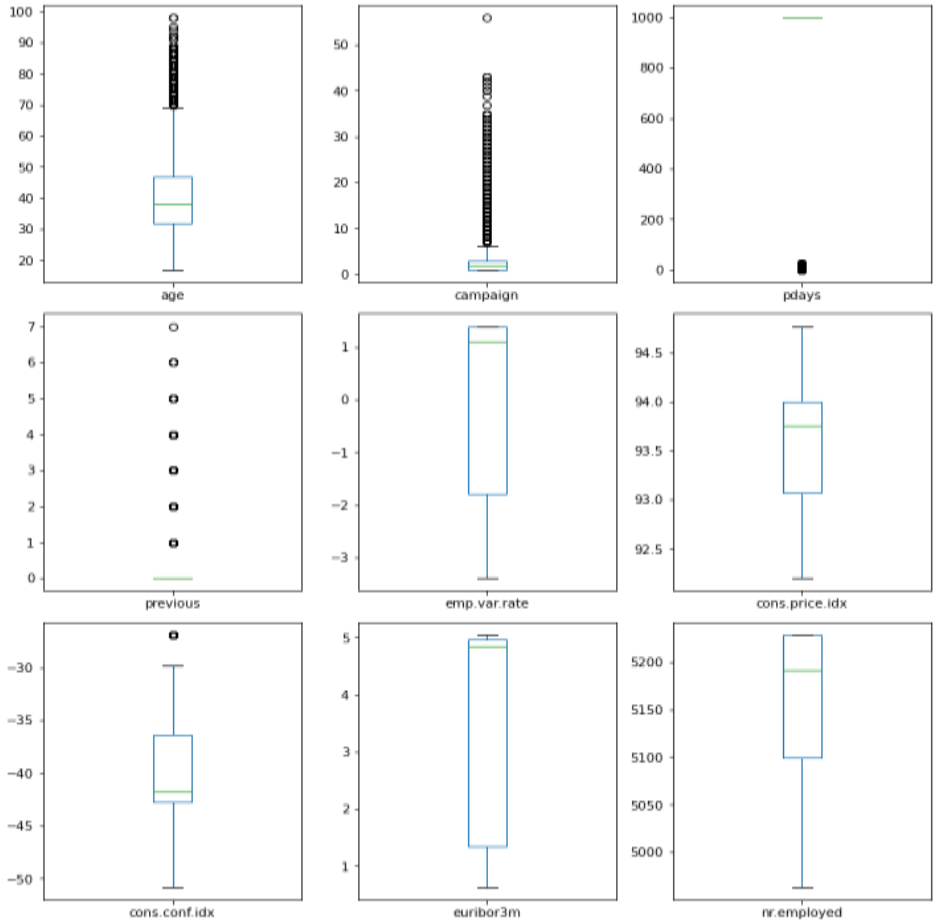
  There are no missing values in the dataset.

```
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
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
```

  Outlier detection

  Outliers are detected using box plots.



Outliers represent those values which are at an abnormal distance from the central distribution of data points. These values affect the statistics of the data mainly mean and mode. Therefore it's important to deal with outliers in the data cleaning stage so that the performance of the model don't get compromised.

In the box plot, outliers are seen in the features **age, campaign, pdays and previous** which are indicated as data points outside the whiskers of the box plot.

- <u>Approaches to overcome the problems</u>

  Considering the statistics of the outlier features

```
df[['age', 'pdays', 'campaign', 'previous']].describe()
```

	age	pdays	campaign	previous
count	41188.00000	41188.000000	41188.000000	41188.000000
mean	40.02406	962.475454	2.567593	0.172963
std	10.42125	186.910907	2.770014	0.494901
min	17.00000	0.000000	1.000000	0.000000
25%	32.00000	999.000000	1.000000	0.000000
50%	38.00000	999.000000	2.000000	0.000000
75%	47.00000	999.000000	3.000000	0.000000
max	98.00000	999.000000	56.000000	7.000000

i. **age:** The maximum value of age is 98 which seems to be realistic. Therefore, it's not dropped.

ii. **pdays:** number of days that passed by after the client was last contacted from a previous campaign. From the statistics, the maximum value of **pdays** is 999. In the features description, this value means the client was not previously contacted. Moreover around 96% of rows contains this value, therefore dropping rows containing 999 seems unrealistic.

```
len(df[df['pdays'] == 999]) / len(df) * 100
```

```
96.32174419733903
```

iii. **campaign:** 'campaign' holds the number of contacts performed during this campaign and for this client. From the statistics, the maximum value is given as 56 which is clearly noise. Numbers for 'campaign' above 20 is around 0.38%.

```
len(df[df['campaign'] > 20])  / len(df) * 100
```

```
0.3811789841701467
```

Therefore, I suggest to impute those rows with average of campaign values.

iv. **previous:** 'previous' holds the number of contacts performed before this campaign and for this client. Since the maximum value of 7 doesn't seem to be a noise, we I chose to ignore this outlier.

4. Treating outliers

Imputation using median

```
#The value which is outside the whisker
print(df['campaign'].quantile(0.95))
```

```
7.0
```

```
#replacing the values which are greater than the 95th percentile
import numpy as np
df['campaign1'] = np.where(df['campaign'] > 7, 2, df['campaign'])
df[['campaign', 'campaign1']].describe()
```

	campaign	campaign1
count	41176.000000	41176.000000
mean	2.567879	2.118127
std	2.770318	1.383215
min	1.000000	1.000000
25%	1.000000	1.000000
50%	2.000000	2.000000
75%	3.000000	3.000000
max	56.000000	7.000000

Imputation using mean

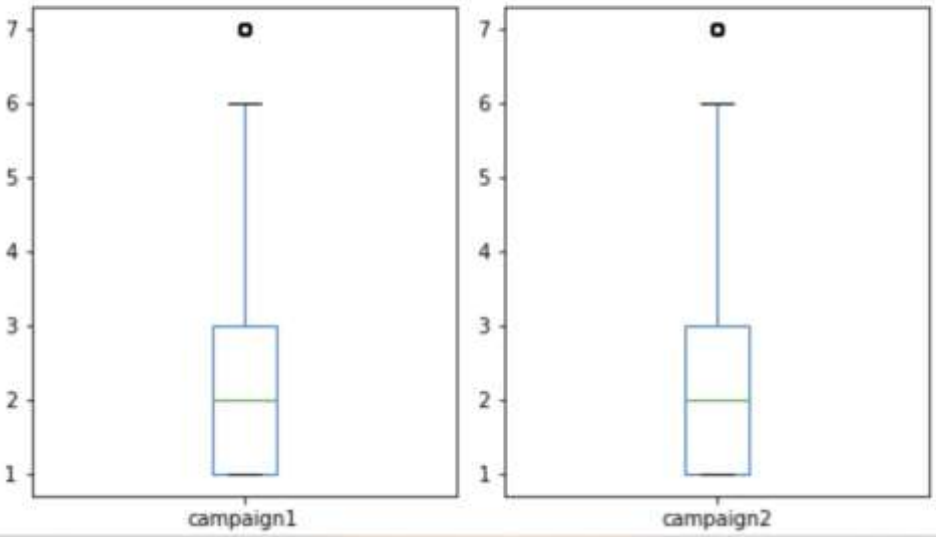
From df.describe(), the mean is 2.56

```
#replacing the values which are greater than the 95th percentile
df['campaign2'] = np.where(df['campaign'] > 7, 2.56, df['campaign'])
df[['campaign', 'campaign1','campaign2']].describe()
```

	campaign	campaign1	campaign2
count	41176.000000	41176.000000	41176.000000
mean	2.567879	2.118127	2.142295
std	2.770318	1.383215	1.385829
min	1.000000	1.000000	1.000000
25%	1.000000	1.000000	1.000000
50%	2.000000	2.000000	2.000000
75%	3.000000	3.000000	3.000000
max	56.000000	7.000000	7.000000

Statistics of the dataset after both median and mean imputation remains more or less the same.

```
#outlier detection  after imputation
import matplotlib.pyplot as plt
cols = ['campaign1','campaign2']
plt.figure(figsize=(10,15))
for i, col in enumerate(cols):
    plt.subplot(4,3,i+1)
    df.boxplot(col)
    plt.grid()
    plt.tight_layout()
```



5. Github Repo link

https://github.com/ArchanaDeviRamesh/Data-Glacier-Project/tree/main/Week9