Week 8 deliverable

I. Team members details

Group name: ***Data science Geeks***

Names: Tasnime Hamdeni

Refka Mejri

Email : hamdeni.tasnime@gmail.com

refka.mejri@enit.utm.tn

Country: Tunisia

Tunisia

College: National Engineering School of Tunis

National Engineering School of Tunis

Specialization: Data science

II. Problem description

We want to get some insight from the data of a bank called ABC that 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).

Business need: Buying Product for customer.

Method: using ML model to help companies shortlist customers whose chances of buying products is more so that their marketing channel can focus on them.

III. Data Understanding

We have information about 41188 clients in a csv file of size 4.70 Mo. For each client, 21 attributes are available in the data. Some demographic data such as:

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')

Some data related with the last contact of the current campaign like:

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.

Some other attributes are as well fully available like:

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)

The output variable or so called the desired target is given by the varibale:
21 - **y** - has the client subscribed a term deposit? (binary: 'yes','no')

IV. Type of data

Entrée [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype  
---  ------          --------------  -----  
 0   age             41188 non-null  int64  
 1   job             41188 non-null  object 
 2   marital         41188 non-null  object 
 3   education       41188 non-null  object 
 4   default         41188 non-null  object 
 5   housing         41188 non-null  object 
 6   loan            41188 non-null  object 
 7   contact         41188 non-null  object 
 8   month           41188 non-null  object 
 9   day_of_week     41188 non-null  object 
 10  duration        41188 non-null  int64  
 11  campaign        41188 non-null  int64  
 12  pdays           41188 non-null  int64  
 13  previous        41188 non-null  int64  
 14  poutcome        41188 non-null  object 
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  object 
dtypes: float64(5), int64(5), object(11)
memory usage: 6.6+ MB
```

Data of type object should be converted into category data. The type of data after conversion is:

change the type of the data

Entrée [13]:
```
# changer le type de la colonne sexe de object à category
df["job"] = df["job"].astype('category')
df["marital"] = df["marital"].astype('category')
df["education"] = df["education"].astype('category')
df["default"] = df["default"].astype('category')
df["housing"] = df["housing"].astype('category')
df["loan"] = df["loan"].astype('category')
df["contact"] = df["contact"].astype('category')
df["month"] = df["month"].astype('category')
df["day_of_week"] = df["day_of_week"].astype('category')
df["duration"] = df["duration"].astype('int64')
df["poutcome"] = df["poutcome"].astype('category')
df["y"] = df["y"].astype('category')

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 41188 entries, 0 to 41187
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             41188 non-null  int64
 1   job             41188 non-null  category
 2   marital         41188 non-null  category
 3   education       41188 non-null  category
 4   default         41188 non-null  category
 5   housing         41188 non-null  category
 6   loan            41188 non-null  category
 7   contact         41188 non-null  category
 8   month           41188 non-null  category
 9   day_of_week     41188 non-null  category
 10  duration        41188 non-null  int64
 11  campaign        41188 non-null  int64
 12  pdays           41188 non-null  int64
 13  previous        41188 non-null  int64
 14  poutcome        41188 non-null  category
 15  emp.var.rate    41188 non-null  float64
 16  cons.price.idx  41188 non-null  float64
 17  cons.conf.idx   41188 non-null  float64
 18  euribor3m       41188 non-null  float64
 19  nr.employed     41188 non-null  float64
 20  y               41188 non-null  category
dtypes: category(11), float64(5), int64(5)
memory usage: 3.6 MB
```

The size of the dataframe as well as the descriptive statistics are given below:

```
df.shape
```

```
(41188, 21)
```

```
print(df.describe())
```

```
                age      duration      campaign         pdays      previous  \
count   41188.00000  41188.000000  41188.000000  41188.000000  41188.000000
mean       40.02406    258.285010      2.567593    962.475454      0.172963
std        10.42125    259.279249      2.770014    186.910907      0.494901
min        17.00000      0.000000      1.000000      0.000000      0.000000
25%        32.00000    102.000000      1.000000    999.000000      0.000000
50%        38.00000    180.000000      2.000000    999.000000      0.000000
75%        47.00000    319.000000      3.000000    999.000000      0.000000
max        98.00000   4918.000000     56.000000    999.000000      7.000000

       emp.var.rate  cons.price.idx  cons.conf.idx     euribor3m   nr.employed
count  41188.000000    41188.000000   41188.000000  41188.000000  41188.000000
mean       0.081886       93.575664     -40.502600      3.621291   5167.035911
std        1.570960        0.578840       4.628198      1.734447     72.251528
min       -3.400000       92.201000     -50.800000      0.634000   4963.600000
25%       -1.800000       93.075000     -42.700000      1.344000   5099.100000
50%        1.100000       93.749000     -41.800000      4.857000   5191.000000
75%        1.400000       93.994000     -36.400000      4.961000   5228.100000
max        1.400000       94.767000     -26.900000      5.045000   5228.100000
```

V. Problems in the data

- **Skewness**

Entrée [6]:
```
df.skew()
```

Out[6]:
```
age               0.784697
duration          3.263141
campaign          4.762507
pdays            -4.922190
previous          3.832042
emp.var.rate     -0.724096
cons.price.idx   -0.230888
cons.conf.idx     0.303180
euribor3m        -0.709188
nr.employed      -1.044262
dtype: float64
```

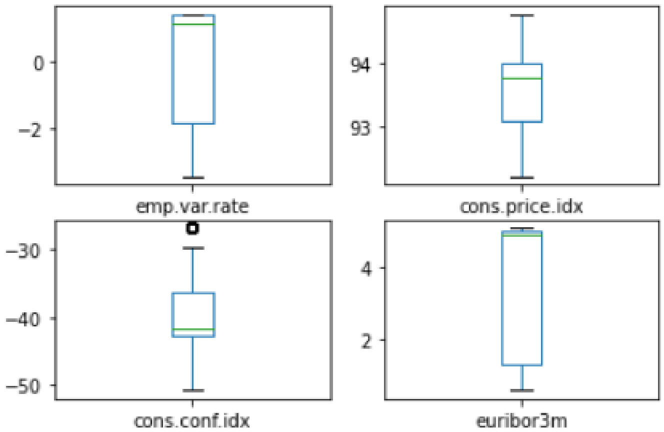
- **Outliers**

We calculate the z-score to detect outliers and we can also draw the box plot to check the eventual existence of outliers:

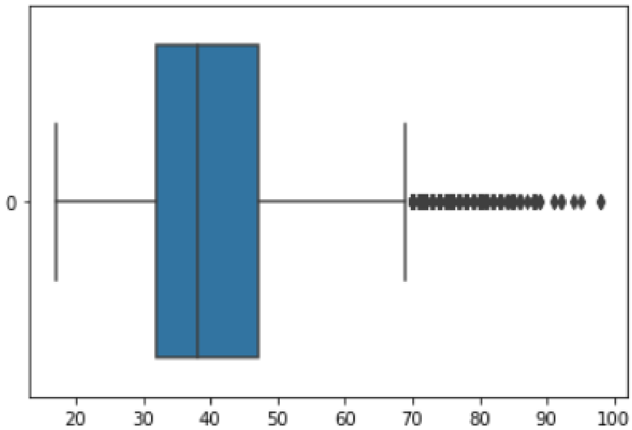
Data Glacier

Your Deep Learning Partner

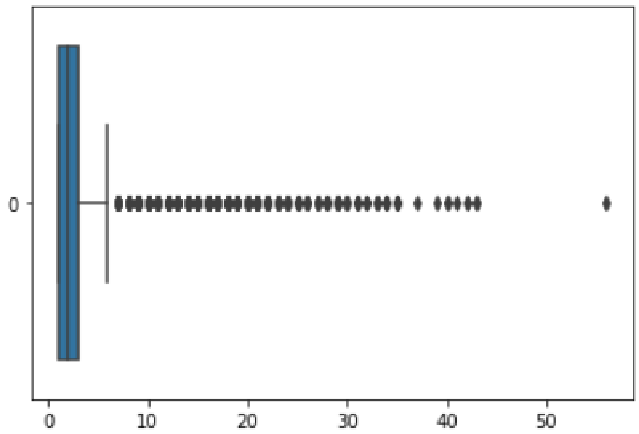
```
: # box and whisker plots
df1=df.iloc[:,15:19]
df1.plot(kind='box', subplots=True, layout=(2,2), sharex=False, sharey=False)
pyplot.show()
```




```
Entrée [8]: sns.boxplot(data = df.age, orient ='h')
plt.show()
```



Entrée [14]:
```
sns.boxplot(data = df.campaign, orient ='h')
plt.show()
```



- **NA values:** There are no NA values in the data

Entrée [54]:
```
df.isna().any()
```

Out[54]:
```
age               False
job               False
marital           False
education         False
default           False
housing           False
loan              False
contact           False
month             False
day_of_week       False
duration          False
campaign          False
pdays             False
previous          False
poutcome          False
emp.var.rate      False
cons.price.idx    False
cons.conf.idx     False
euribor3m         False
nr.employed       False
y                 False
dtype: bool
```

VI. Problems overcoming techniques

- **Removing outliers:**

We can deal with outliers either by calculating the z-score or by calculating the interquartile range.

Using Z-score

The z-score is a numerical value that quatifies the relationship to the mean of of the values in our data. Z-score is measured in terms of standard deviations from the mean.

removing outliers

Entrée [44]:
```
#detecting outliers using z score

z = np.abs(stats.zscore(df.age))
print(z)
```

```
[1.53303429 1.62899323 0.29018564 ... 1.53303429 0.38152696 3.26029527]
```

Entrée [45]:
```
threshold = 3
print(np.where(z > 3))
```

```
(array([27757, 27780, 27800, 27802, 27805, 27808, 27810, 27811, 27812,
       27813, 27814, 27815, 27816, 27817, 27818, 27826, 27851, 27875,
       27930, 27950, 27951, 27963, 28220, 28221, 28312, 28456, 29263,
       29498, 29625, 29682, 29973, 29977, 29981, 29990, 30000, 30004,
       30006, 30072, 30078, 30079, 30103, 30110, 30133, 30171, 30214,
       30225, 30241, 30334, 30430, 30460, 30589, 35833, 35856, 35878,
       35973, 36183, 36285, 36311, 36383, 36384, 36816, 36998, 37136,
       37137, 37186, 37190, 37192, 37193, 37195, 37206, 37207, 37213,
       37219, 37235, 37237, 37239, 37257, 37260, 37341, 37355, 37403,
       37454, 37455, 37472, 37479, 37493, 37505, 37509, 37512, 37525,
       37532, 37597, 37601, 37602, 37604, 37635, 37675, 37679, 37690,
       37692, 37715, 37735, 37736, 37743, 37756, 37769, 37775, 37784,
       37818, 37819, 37820, 37861, 37868, 37870, 37873, 37905, 37920,
       37946, 37951, 37952, 37954, 37999, 38005, 38019, 38020, 38022,
       38032, 38033, 38045, 38052, 38054, 38065, 38136, 38166, 38178,
       38179, 38184, 38191, 38192, 38193, 38195, 38206, 38229, 38241,
       38246, 38252, 38260, 38279, 38288, 38314, 38316, 38322, 38326,
       38410, 38415, 38452, 38455, 38471, 38486, 38505, 38517, 38518,
       38536, 38548, 38549, 38556, 38557, 38577, 38580, 38582, 38587,
       38600, 38643, 38676, 38697, 38700, 38703, 38722, 38726, 38735,
       38740, 38744, 38751, 38783, 38810, 38824, 38825, 38831, 38846,
       38876, 38878, 38880, 38892, 38901, 38909, 38921, 38924, 38936,
       38942, 38943, 38944, 38946, 38953, 38960, 38967, 38968, 38984,
       39001, 39011, 39032, 39038, 39041, 39042, 39043, 39055, 39058,
       39061, 39062, 39093, 39115, 39124, 39133, 39184, 39186, 39190,
       39204, 39261, 39264, 39275, 39319, 39332, 39342, 39348, 39360,
       39377, 39402, 39410, 39411, 39415, 39444, 39452, 39466, 39471,
       39472, 39473, 39474, 39475, 39476, 39477, 39478, 39479, 39486,
       39487, 39488, 39489, 39493, 39495, 39498, 39504, 39577, 39578,
       39601, 39614, 39625, 39639, 39650, 39655, 39676, 39678, 39692,
       39719, 39722, 39724, 39730, 39734, 39737, 39752, 39762, 39766,
       39768, 39786, 39795, 39847, 39890, 39892, 39923, 39947, 39971,
       39974, 39975, 40001, 40045, 40050, 40060, 40076, 40078, 40080,
       40081, 40085, 40114, 40117, 40119, 40129, 40142, 40149, 40162,
       40194, 40195, 40196, 40197, 40201, 40218, 40219, 40220, 40262,
       40273, 40277, 40289, 40291, 40331, 40344, 40356, 40400, 40414,
       40421, 40437, 40445, 40450, 40468, 40469, 40470, 40484, 40488,
       40529, 40546, 40554, 40575, 40592, 40611, 40621, 40624, 40631,
       40636, 40638, 40639, 40651, 40667, 40669, 40686, 40702, 40714,
       40716, 40718, 40727, 40748, 40756, 40879, 40915, 40950, 40965,
       40966, 40969, 40982, 40983, 40986, 40996, 41004, 41183, 41187],
      dtype=int64),)
```

```
#shape before removing outliers
df.shape
```

```
(41188, 21)
```

```
# removing outliers
df1 = df[(z < 3)]
#shape after removing outliers
df1.shape
```

```
(40819, 21)
```