# Assignment -2

## preprocessing the dataset

### 2148059

In [1]:
```
import pandas as pd
```

In [2]:
```
df=pd.read_csv("/Users/persie/Downloads/bank-12.csv")
```

In [3]:
```
df.head(5)
```

Out[3]:

| | age | job | marital | education | default | balance | housing | loan | contact | day | m |
|---|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | unemployed | married | primary | no | 1787 | no | no | cellular | 19 | |
| **1** | 33 | services | married | secondary | no | 4789 | yes | yes | cellular | 11 | |
| **2** | 35 | management | single | tertiary | no | 1350 | yes | no | cellular | 16 | |
| **3** | 30 | management | married | tertiary | no | 1476 | yes | yes | unknown | 3 | |
| **4** | 59 | blue-collar | married | secondary | no | 0 | yes | no | unknown | 5 | |

In [4]:
```
#value counts of the target variable
df["y"].value_counts()
```

Out[4]:
```
no      4000
yes      521
Name: y, dtype: int64
```

In [5]:
```
rem=["contact","day"]
df=df.drop(rem,axis=1)
df.columns
```

Out[5]:
```
Index(['age', 'job', 'marital', 'education', 'default', 'balance',
'housing',
       'loan', 'month', 'duration', 'campaign', 'pdays', 'previous
',
       'poutcome', 'y'],
      dtype='object')
```

In [6]:
```
#asssigning 1 if target variable is yes and 0 if target is no
df["y"]=[1 if x=="yes" else 0 for x in df["y"]]

#x as dataframe of features and y as the target variable
x=df.drop("y",1)
y=df.y
```

In [7]:
```
x.head(5)
```

Out[7]:

| | age | job | marital | education | default | balance | housing | loan | month | duration |
|---|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | unemployed | married | primary | no | 1787 | no | no | oct | 79 |
| **1** | 33 | services | married | secondary | no | 4789 | yes | yes | may | 220 |
| **2** | 35 | management | single | tertiary | no | 1350 | yes | no | apr | 185 |
| **3** | 30 | management | married | tertiary | no | 1476 | yes | yes | jun | 199 |
| **4** | 59 | blue-collar | married | secondary | no | 0 | yes | no | may | 226 |

In [8]:
```
y.head(5)
```

Out[8]:
```
0    0
1    0
2    0
3    0
4    0
Name: y, dtype: int64
```

## Data Cleaning

#### A. dealing with the data types

#### ***converting categorical data to numerical data***

In [9]:
```
#categorical variable
x["marital"].head()
```

Out[9]:
```
0    married
1    married
2     single
3    married
4    married
Name: marital, dtype: object
```

In [10]:
```
#cheking the no of categories in all the features
for col_names in x.columns:
    if x[col_names].dtype=="object":
        cat=len(x[col_names].unique())
        print("features: {col_names} has {cat} categories".format(c
```

```
features: job has 12 categories
features: marital has 3 categories
features: education has 4 categories
features: default has 2 categories
features: housing has 2 categories
features: loan has 2 categories
features: month has 12 categories
features: poutcome has 4 categories
```

**Categorise all the other features exceopth month and job**

In [14]:
```
#list of features to dummy
todummy=["marital","education","default","housing","loan","poutcome
```

In [15]:
```
#function to dummy all the categorical variables for modelling
def dummy(df,todummy):
    for x in todummy:
        dummies=pd.get_dummies(df[x],prefix=x,dummy_na=False)
        df=df.drop(x,1)
        df=pd.concat([df,dummies],axis=1)
    return df
```

In [16]:
```
x= dummy(x,todummy)
x.head(5)
```

Out[16]:

| | age | job | balance | month | duration | campaign | pdays | previous | marital_divorce |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | unemployed | 1787 | oct | 79 | 1 | -1 | 0 | |
| **1** | 33 | services | 4789 | may | 220 | 1 | 339 | 4 | |
| **2** | 35 | management | 1350 | apr | 185 | 1 | 330 | 1 | |
| **3** | 30 | management | 1476 | jun | 199 | 4 | -1 | 0 | |
| **4** | 59 | blue-collar | 0 | may | 226 | 1 | -1 | 0 | |

5 rows × 25 columns

In [18]:
```
x.columns
```

Out[18]:
```
Index(['age', 'job', 'balance', 'month', 'duration', 'campaign', '
pdays',
       'previous', 'marital_divorced', 'marital_married', 'marital
_single',
       'education_primary', 'education_secondary', 'education_tert
iary',
       'education_unknown', 'default_no', 'default_yes', 'housing_
no',
       'housing_yes', 'loan_no', 'loan_yes', 'poutcome_failure',
       'poutcome_other', 'poutcome_success', 'poutcome_unknown'],
      dtype='object')
```

## b. handling missing values

In [19]:
```
x.isnull().sum().sort_values(ascending=True)
```

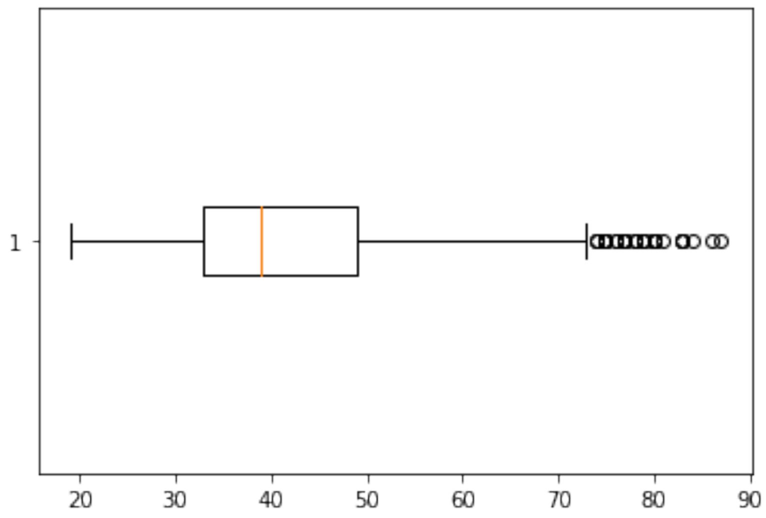
Out[19]:
```
age                    0
poutcome_other         0
poutcome_failure       0
loan_yes               0
loan_no                0
housing_yes            0
housing_no             0
default_yes            0
default_no             0
education_unknown      0
education_tertiary     0
poutcome_success       0
education_secondary    0
marital_single         0
marital_married        0
marital_divorced       0
previous               0
pdays                  0
campaign               0
duration               0
month                  0
balance                0
job                    0
education_primary      0
poutcome_unknown       0
dtype: int64
```

**there is no missing values in the data**

## outlier detection

In [20]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [21]:
```
plt.boxplot(x["age"],vert=False)
plt.show()
```



In [22]:
```
def outlier(x):
    q1=np.percentile(x,25)
    q3=np.percentile(x,75)
    iqr=q3-q1
    flr=q1-1.5*iqr
    ceil=q3+1.5*iqr
    outlier_indices=list(x.index[(x<flr)|(x>ceil)])
    outlier_values=list(x[outlier_indices])
    return outlier_values,outlier_indices
```

In [23]:
```
values,indices=outlier(x["age"])
print(np.sort(values))
```

```
[74 74 74 75 75 75 75 75 75 76 76 77 77 77 77 77 77 78 78 78 79 79
79 79
 80 80 80 80 80 80 81 83 83 83 83 84 86 87]
```

**the above values are the outliers**

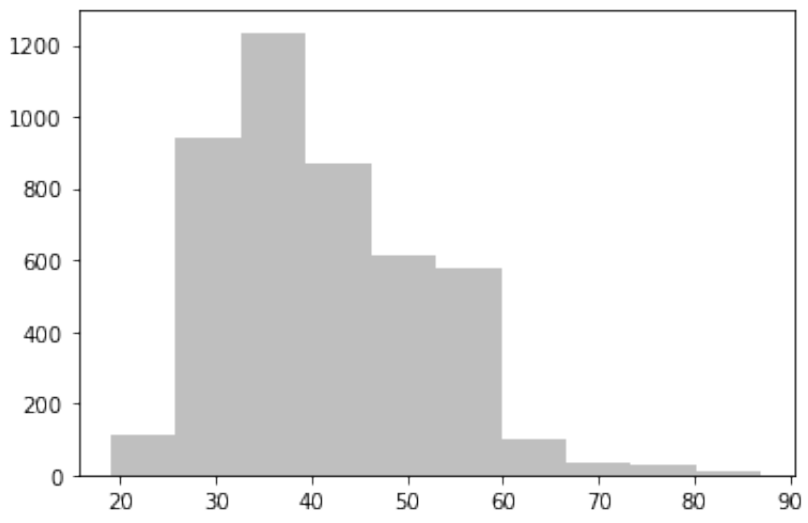
In [24]:
```
x.head(5)
```

Out[24]:

| | age | job | balance | month | duration | campaign | pdays | previous | marital_divorce |
|---|---|---|---|---|---|---|---|---|---|
| **0** | 30 | unemployed | 1787 | oct | 79 | 1 | -1 | 0 | |
| **1** | 33 | services | 4789 | may | 220 | 1 | 339 | 4 | |
| **2** | 35 | management | 1350 | apr | 185 | 1 | 330 | 1 | |
| **3** | 30 | management | 1476 | jun | 199 | 4 | -1 | 0 | |
| **4** | 59 | blue-collar | 0 | may | 226 | 1 | -1 | 0 | |

5 rows × 25 columns

In [25]:
```
plt.hist(x["age"], color='gray',alpha=0.5)
plt.show()
```



**the graph of the feature 'age' is rightly skewed**