Assignment -3

Feature Selection

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 [11]:
```
#list of features to dummy
todummy=["marital","education","default","housing","loan","poutcome
```

In [12]:
```
#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 [13]:
```
x= dummy(x,todummy)
x.head(5)
```

Out[13]:

	age	balance	duration	campaign	pdays	previous	marital_divorced	marital_married	n
0	30	1787	79	1	-1	0	0	1	
1	33	4789	220	1	339	4	0	1	
2	35	1350	185	1	330	1	0	0	
3	30	1476	199	4	-1	0	0	1	
4	59	0	226	1	-1	0	0	1	

5 rows × 47 columns

In [14]:
```
x.columns
```

Out[14]:
```
Index(['age', 'balance', 'duration', 'campaign', 'pdays', 'previou
s',
       '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', '
job_admin.',
       'job_blue-collar', 'job_entrepreneur', 'job_housemaid',
       'job_management', 'job_retired', 'job_self-employed', 'job_
services',
       'job_student', 'job_technician', 'job_unemployed', 'job_unk
nown',
       'month_apr', 'month_aug', 'month_dec', 'month_feb', 'month_
jan',
       'month_jul', 'month_jun', 'month_mar', 'month_may', 'month_
nov',
       'month_oct', 'month_sep'],
      dtype='object')
```

b. handling missing values

In [15]:
```
x.isnull().sum().sort_values(ascending=True)
```

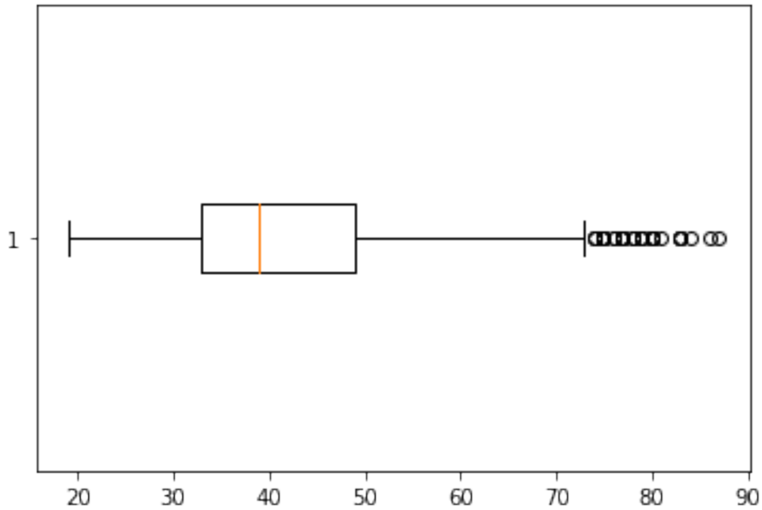
Out[15]:
```
age                    0
job_entrepreneur       0
job_housemaid          0
job_management         0
job_retired            0
job_self-employed      0
job_services           0
job_student            0
job_technician         0
job_unemployed         0
job_blue-collar        0
job_unknown            0
month_aug              0
month_dec              0
month_feb              0
month_jan              0
month_jul              0
month_jun              0
month_mar              0
month_may              0
month_nov              0
month_apr              0
month_oct              0
job_admin.             0
poutcome_success       0
balance                0
duration               0
campaign               0
pdays                  0
previous               0
marital_divorced       0
marital_married        0
marital_single         0
education_primary      0
poutcome_unknown       0
education_secondary    0
education_unknown      0
default_no             0
default_yes            0
housing_no             0
housing_yes            0
loan_no                0
loan_yes               0
poutcome_failure       0
poutcome_other         0
education_tertiary     0
month_sep              0
dtype: int64
```

there is no missing values in the data

outlier detection

In [16]:
```
import matplotlib.pyplot as plt
import numpy as np
```

In [17]:
```
plt.boxplot(x["age"],vert=False)
plt.show()
```



In [18]:
```
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 [19]:
```
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

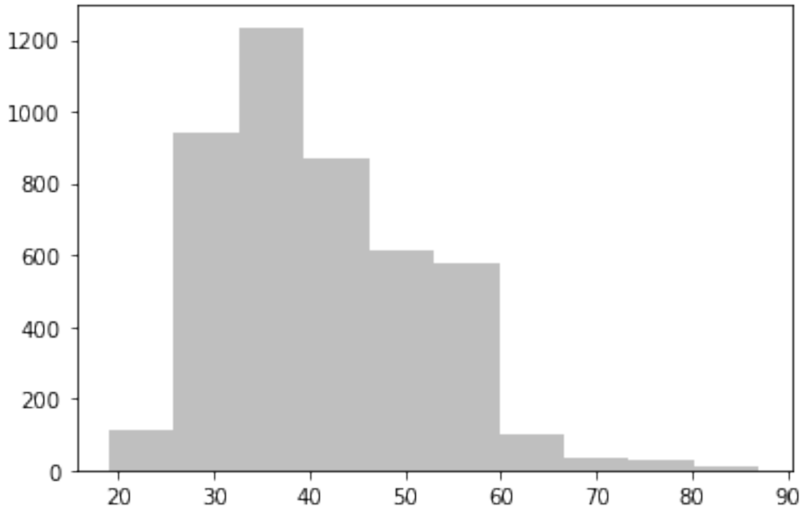
In [20]:
```
x.head(5)
```

Out[20]:

	age	balance	duration	campaign	pdays	previous	marital_divorced	marital_married	n
0	30	1787	79	1	-1	0	0	1	
1	33	4789	220	1	339	4	0	1	
2	35	1350	185	1	330	1	0	0	
3	30	1476	199	4	-1	0	0	1	
4	59	0	226	1	-1	0	0	1	

5 rows × 47 columns

In [21]:
```
plt.hist(x["age"], color='gray',alpha=0.5)
plt.show()
```



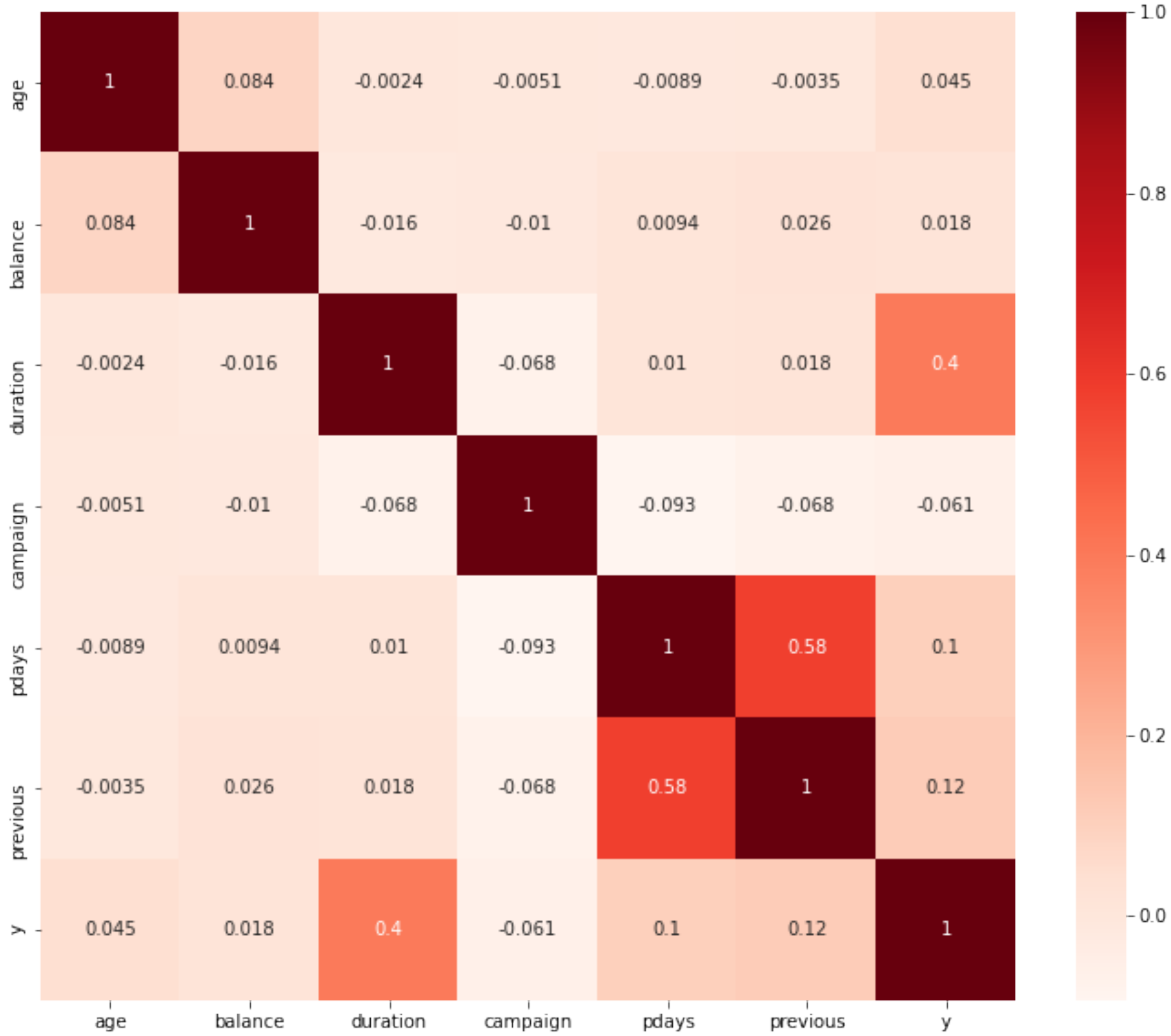
the graph of the feature 'age' is rightly skewed

Feature Selection

(I) Relationship Between target and features

In [22]:
```
import seaborn as sns
plt.figure(figsize=(12,10))
cor=df.corr()
sns.heatmap(cor, annot= True,cmap=plt.cm.Reds)

plt.show()
```



In [23]:
```
#correlation with output variable
cor_target=abs(cor['y'])
print("cor target \n",cor_target)

relevant_feature=cor_target[cor_target>0.5]
relevant_feature
```

```
cor target
 age         0.045092
balance     0.017905
duration    0.401118
campaign    0.061147
pdays       0.104087
previous    0.116714
y           1.000000
Name: y, dtype: float64
```

Out[23]:
```
y    1.0
Name: y, dtype: float64
```

From above results we can find that there very less correlation between the features and target and thus this method is not significant for this data set

(II) Entropy Based Feature Selection

In [24]:
```
from sklearn.feature_selection import mutual_info_classif as MIC
mi_score=MIC(x,y)
print(mi_score)
```

```
[0.00599398 0.01139159 0.07096436 0.00159616 0.02679685 0.01572376
 0.         0.         0.00298128 0.         0.00168663 0.
 0.         0.         0.         0.00749268 0.00454537 0.00319249
 0.01640568 0.00521922 0.         0.01485622 0.01020222 0.
 0.         0.         0.01036719 0.         0.0044331  0.00251556
 0.00681661 0.00208851 0.00562023 0.00362384 0.         0.00824617
 0.00534494 0.         0.01284615 0.         0.         0.
 0.         0.0106623  0.00666852 0.01156312 0.00243792]
```

In [25]:
```
from sklearn.model_selection import train_test_split as tts
x_train_1,x_test_1,y_train_1,y_test_1=tts(x,y,random_state=0,strati
```

In [26]:
```
mi_score_selected_index=np.where(mi_score>0.02)[0]
x_2=x.iloc[:,mi_score_selected_index]
x_train_2,x_test_2,y_train_2,y_test_2=tts(x_2,y,random_state=0,stra
```

In [27]:
```
mi_score_selected_index=np.where(mi_score<0.02)[0]
x_3=x.iloc[:,mi_score_selected_index]
x_train_3,x_test_3,y_train_3,y_test_3=tts(x_3,y,random_state=0,stra
```

In [28]:
```
from sklearn.tree import DecisionTreeClassifier as DTC
model_1=DTC().fit(x_train_1,y_train_1)
model_2=DTC().fit(x_train_2,y_train_2)
model_3=DTC().fit(x_train_3,y_train_3)
score_1=model_1.score(x_test_1,y_test_1)
score_2=model_2.score(x_test_2,y_test_2)
score_3=model_3.score(x_test_3,y_test_3)
```

In [29]:
```
print(f"score1:{score_1}\n score2:{score_2}\n score_3:{score_3}")
```

```
score1:0.8620689655172413
 score2:0.8762157382847038
 score_3:0.8275862068965517
```

we can see that score 2 has predicted a good accuracy compared to other cases

(III) EMBEDDED METHOD DECISION TREE

RANDOM FOREST

FEATURE IMPORTANCE IS CALCULATED

In [35]:
```
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
num_feats=30
embeded_rf_selector = SelectFromModel(RandomForestClassifier(n_esti
embeded_rf_selector.fit(x, y)
```

Out[35]:
```
SelectFromModel(estimator=RandomForestClassifier(), max_features=3
0)
```

In [36]:
```
embeded_rf_support = embeded_rf_selector.get_support()
embeded_rf_feature = x.loc[:,embeded_rf_support].columns.tolist()
print(str(len(embeded_rf_feature)), 'selected features')
```

```
7 selected features
```

In [37]:
```
embeded_rf_feature
```

Out[37]:
```
['age',
 'balance',
 'duration',
 'campaign',
 'pdays',
 'previous',
 'poutcome_success']
```

Using The above model, we reduce the feature to 7. thus we have selected the 7 important features

(IV) Recursive Feature Elimination

Accuracy Based

In [38]:
```
#RECURSIVE FEATURE ELMINIATION
#ACCURACY BASED
from sklearn.feature_selection import RFE
from sklearn.linear_model import LogisticRegression
rfe_selector = RFE(estimator=LogisticRegression(), n_features_to_se
rfe_selector.fit(x, y)
```

```
Fitting estimator with 47 features.
Fitting estimator with 37 features.
Fitting estimator with 27 features.

/Users/persie/opt/anaconda3/lib/python3.8/site-packages/sklearn/li
near_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
/Users/persie/opt/anaconda3/lib/python3.8/site-packages/sklearn/li
near_model/_logistic.py:763: ConvergenceWarning: lbfgs failed to c
onverge (status=1):
STOP: TOTAL NO. of ITERATIONS REACHED LIMIT.

Increase the number of iterations (max_iter) or scale the data as
shown in:
    https://scikit-learn.org/stable/modules/preprocessing.html
(https://scikit-learn.org/stable/modules/preprocessing.html)
Please also refer to the documentation for alternative solver opti
ons:
    https://scikit-learn.org/stable/modules/linear_model.html#logi
stic-regression (https://scikit-learn.org/stable/modules/linear_mo
del.html#logistic-regression)
  n_iter_i = _check_optimize_result(
```

Out[38]:
```
RFE(estimator=LogisticRegression(), n_features_to_select=20, step=
10, verbose=5)
```

In [39]:
```
rfe_support = rfe_selector.get_support()
rfe_feature = x.loc[:,rfe_support].columns.tolist()
print(str(len(rfe_feature)), 'selected features')
```

```
20 selected features
```

In [32]:
```
rfe_feature
```

Out[32]:
```
['marital_divorced',
 'marital_married',
 'education_secondary',
 'education_tertiary',
 'education_unknown',
 'default_no',
 'housing_no',
 'housing_yes',
 'loan_no',
 'loan_yes',
 'poutcome_failure',
 'poutcome_other',
 'poutcome_success',
 'poutcome_unknown',
 'job_blue-collar',
 'job_retired',
 'job_services',
 'job_technician',
 'job_unemployed',
```

Thus we reduce and select 20 feature using Recursive Method

In [ ]:
```

```