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

In [2]:
```
data=pd.read_csv("bank.csv")
```

In [3]:
```
data.isnull().any()
```

Out[3]:
```
age          False
job          False
marital      False
education    False
default      False
balance      False
housing      False
loan         False
contact      False
day          False
month        False
duration     False
campaign     False
pdays        False
previous     False
poutcome     False
deposit      False
dtype: bool
```

In [4]:
```
data.head(2)
```

Out[4]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes
1	56	admin.	married	secondary	no	45	no	no	unknown	5	may	1467	1	-1	0	unknown	yes

In [5]:
```
data['marital'].unique() # so 3 col will generate 0 1 2
```

Out[5]:
```
array(['married', 'single', 'divorced'], dtype=object)
```

In [6]:
```
data["education"].unique()# 4 col will generate 0 1 2 3
```
Out[6]:
```
array(['secondary', 'tertiary', 'primary', 'unknown'], dtype=object)
```

In [7]:
```
data["job"].unique()#12 col will generate 0 1 2 ..12
```
Out[7]:
```
array(['admin.', 'technician', 'services', 'management', 'retired',
       'blue-collar', 'unemployed', 'entrepreneur', 'housemaid',
       'unknown', 'self-employed', 'student'], dtype=object)
```

In [8]:
```
data["default"].unique()
```
Out[8]:
```
array(['no', 'yes'], dtype=object)
```

In [9]:
```
data["contact"].unique()# 3 col will generate
```
Out[9]:
```
array(['unknown', 'cellular', 'telephone'], dtype=object)
```

In [10]:
```
data["poutcome"].unique()#4 col will generate
```
Out[10]:
```
array(['unknown', 'other', 'failure', 'success'], dtype=object)
```

In [11]:
```
data["housing"].unique()
```
Out[11]:
```
array(['yes', 'no'], dtype=object)
```

In [12]:
```
data["month"].unique()
```
Out[12]:
```
array(['may', 'jun', 'jul', 'aug', 'oct', 'nov', 'dec', 'jan', 'feb',
       'mar', 'apr', 'sep'], dtype=object)
```

In [13]:
```
data["deposit"].unique()
```
Out[13]:
```
array(['yes', 'no'], dtype=object)
```

In [14]:
```
data["loan"].unique()
```
Out[14]:
```
array(['no', 'yes'], dtype=object)
```

In [15]:
```
data.head(1)
```

Out[15]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	admin.	married	secondary	no	2343	yes	no	unknown	5	may	1042	1	-1	0	unknown	yes

In [16]:
```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
```

In [17]:
```
data['job'] = le.fit_transform(data['job'] )
data['marital'] = le.fit_transform(data['marital'] )
data['education'] = le.fit_transform(data['education'] )
data['default'] = le.fit_transform(data['default'] )
data['housing'] = le.fit_transform(data['housing'] )
data['contact'] = le.fit_transform(data['contact'] )
data['month'] = le.fit_transform(data['month'] )
data['poutcome'] = le.fit_transform(data['poutcome'] )
data['loan'] = le.fit_transform(data['loan'] )
data['deposit'] = le.fit_transform(data['deposit'] )
```

In [18]:
```
data.head(2)
```

Out[18]:

	age	job	marital	education	default	balance	housing	loan	contact	day	month	duration	campaign	pdays	previous	poutcome	deposit
0	59	0	1	1	0	2343	1	0	2	5	8	1042	1	-1	0	3	1
1	56	0	1	1	0	45	0	0	2	5	8	1467	1	-1	0	3	1

In [19]:
```
x=data.iloc[:,0:16].values
y=data.iloc[:,16].values
```

```
In [20]: x
```

```
Out[20]: array([[ 59,   0,   1, ...,  -1,   0,   3],
               [ 56,   0,   1, ...,  -1,   0,   3],
               [ 41,   9,   1, ...,  -1,   0,   3],
               ...,
               [ 32,   9,   2, ...,  -1,   0,   3],
               [ 43,   9,   1, ..., 172,   5,   0],
               [ 34,   9,   1, ...,  -1,   0,   3]], dtype=int64)
```

```
In [21]: y
```

```
Out[21]: array([1, 1, 1, ..., 0, 0, 0])
```

In [22]:
```
from sklearn.preprocessing import OneHotEncoder
one=OneHotEncoder()
a=one.fit_transform(x[:,1:2]).toarray()
b=one.fit_transform(x[:,2:3]).toarray()
d=one.fit_transform(x[:,3:4]).toarray()
g=one.fit_transform(x[:,8:9]).toarray()
h=one.fit_transform(x[:,10:11]).toarray()
k=one.fit_transform(x[:,15:16]).toarray()
x=np.delete(x,[1,2,3,8,10,15],axis=1)
```

```
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
```

```
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
C:\Users\anikp\Anaconda3\lib\site-packages\sklearn\preprocessing\_encoders.py:368: FutureWarning: The handling of integer da
ta will change in version 0.22. Currently, the categories are determined based on the range [0, max(values)], while in the f
uture they will be determined based on the unique values.
If you want the future behaviour and silence this warning, you can specify "categories='auto'".
In case you used a LabelEncoder before this OneHotEncoder to convert the categories to integers, then you can now use the On
eHotEncoder directly.
  warnings.warn(msg, FutureWarning)
```

In [23]:
```
x.shape
```

Out[23]:
```
(11162, 10)
```

In [24]:
```
x=np.concatenate((k,h,g,d,b,a,x),axis=1)
```

In [25]:
```
x.shape
```

Out[25]:
```
(11162, 48)
```

In [26]:
```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=0)
```

In [27]:
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

In [28]:
```
x.shape
```

Out[28]:
```
(11162, 48)
```

In [ ]:
```

```