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

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

In [3]:
```
bank.head(20)
```

Out[3]:

contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no
telephone	may	mon	...	1	999	0	nonexistent	1.1	93.994	-36.4	4.857	5191.0	no

In [30]:
```
Bank=pd.DataFrame(bank)
```

In [35]:
```
Bank1=Bank.drop(Bank.index[20000:41188],axis=0)
```

In [36]:
```
Bank1.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 20000 entries, 0 to 19999
Data columns (total 21 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             20000 non-null  int64
 1   job             20000 non-null  int64
 2   marital         20000 non-null  int64
 3   education       20000 non-null  int64
 4   default         20000 non-null  int64
 5   housing         20000 non-null  int64
 6   loan            20000 non-null  int64
 7   contact         20000 non-null  int64
 8   month           20000 non-null  int64
 9   day_of_week     20000 non-null  int64
 10  duration        20000 non-null  int64
 11  campaign        20000 non-null  int64
 12  pdays           20000 non-null  int64
 13  previous        20000 non-null  int64
 14  poutcome        20000 non-null  int64
 15  emp.var.rate    20000 non-null  int64
 16  cons.price.idx  20000 non-null  int64
 17  cons.conf.idx   20000 non-null  int64
 18  euribor3m       20000 non-null  int64
 19  nr.employed     20000 non-null  int64
 20  y               20000 non-null  int64
dtypes: int64(21)
memory usage: 3.4 MB
```

In [5]:
```
import sklearn
from sklearn.preprocessing import LabelEncoder
LE=LabelEncoder()
```

In [37]:
```
for i in Bank1:
    Bank1[i]=LE.fit_transform(Bank1[i])
```

In [38]:
```
Bank1
```

Out[38]:

oan	contact	month	day_of_week	...	campaign	pdays	previous	poutcome	emp.var.rate	cons.price.idx	cons.conf.idx	euribor3m	nr.employed	y
0	1	3	1	...	0	0	0	0	0	2	2	2	0	0
0	1	3	1	...	0	0	0	0	0	2	2	2	0	0
0	1	3	1	...	0	0	0	0	0	2	2	2	0	0
0	1	3	1	...	0	0	0	0	0	2	2	2	0	0
2	1	3	1	...	0	0	0	0	0	2	2	2	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
2	0	0	0	...	1	0	0	0	1	0	3	19	1	0
0	0	0	0	...	2	0	0	0	1	0	3	19	1	0
0	0	0	0	...	2	0	0	0	1	0	3	19	1	0
0	0	0	0	...	6	0	0	0	1	0	3	19	1	1
0	0	0	0	...	3	0	0	0	1	0	3	19	1	1

In [39]:
```
Bank1_corr=Bank1.corr()
Bank1_corr
```

Out[39]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	pdays
age	1.000000	0.025188	-0.357162	-0.073913	0.235125	0.007317	-0.005065	0.037383	0.025578	-0.036335	...	0.005475	NaN
job	0.025188	1.000000	0.002551	0.178512	-0.018782	0.002032	-0.004887	-0.008060	-0.014212	0.003956	...	-0.008169	NaN
marital	-0.357162	0.002551	1.000000	0.080394	-0.057921	-0.000973	0.006898	-0.041816	-0.043661	0.011324	...	0.008293	NaN
education	-0.073913	0.178512	0.080394	1.000000	-0.178807	0.003394	0.006401	-0.068510	-0.083131	-0.005986	...	0.014176	NaN
default	0.235125	-0.018782	-0.057921	-0.178807	1.000000	0.002325	-0.004799	0.050574	0.052058	-0.014677	...	-0.002241	NaN
housing	0.007317	0.002032	-0.000973	0.003394	0.002325	1.000000	0.037236	-0.069239	-0.057048	0.013103	...	0.000886	NaN
loan	-0.005065	-0.004887	0.006898	0.006401	-0.004799	0.037236	1.000000	-0.011903	-0.012396	-0.011393	...	0.001398	NaN
contact	0.037383	-0.008060	-0.041816	-0.068510	0.050574	-0.069239	-0.011903	1.000000	0.817144	-0.037410	...	-0.001711	NaN
month	0.025578	-0.014212	-0.043661	-0.083131	0.052058	-0.057048	-0.012396	0.817144	1.000000	-0.015309	...	-0.044489	NaN
day_of_week	-0.036335	0.003956	0.011324	-0.005986	-0.014677	0.013103	-0.011393	-0.037410	-0.015309	1.000000	...	-0.045569	NaN
duration	-0.016448	-0.002893	0.010215	-0.011039	-0.015488	-0.006142	-0.002742	-0.041859	-0.002542	0.027843	...	-0.080365	NaN
campaign	0.005475	-0.008169	0.008293	0.014176	-0.002241	0.000886	0.001398	-0.001711	-0.044489	-0.045569	...	1.000000	NaN
pdays	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
previous	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
poutcome	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	NaN
emp.var.rate	-0.032921	0.000558	0.036115	0.048843	-0.039420	0.027264	0.010100	-0.575176	-0.867607	-0.014782	...	0.092569	NaN
cons.price.idx	0.019667	-0.019576	-0.039464	-0.078526	0.047744	-0.074973	-0.011973	0.788668	0.663152	-0.061435	...	0.036309	NaN
cons.conf.idx	0.088264	0.027820	-0.049757	0.014114	0.039690	-0.027353	-0.015924	0.497977	0.528907	-0.020172	...	-0.117000	NaN
euribor3m	-0.019516	0.017209	0.035816	0.071532	-0.045986	0.042442	0.014152	-0.674751	-0.899244	-0.039120	...	0.089024	NaN
nr.employed	-0.032921	0.000558	0.036115	0.048843	-0.039420	0.027264	0.010100	-0.575176	-0.867607	-0.014782	...	0.092569	NaN
y	-0.015405	-0.002452	0.012913	0.003693	-0.011680	-0.004998	0.003164	-0.070989	-0.065353	0.000955	...	-0.011530	NaN

21 rows × 21 columns

In [9]:
```
import seaborn as sns
import matplotlib.pyplot as plt
```

In [40]:
```
x_ind=Bank1.drop("y",axis=1)
y_dep=Bank1.y
```

In [41]:
```
from sklearn.model_selection import train_test_split
```

In [42]:
```
x_train, x_test, y_train, y_test = train_test_split(x_ind, y_dep, test_size=0.2, random_state=2)
```

In [43]:
```
import sklearn
from sklearn.svm import SVC
svM=SVC(kernel="linear")
y_pred_svm=svM.fit(x_train,y_train).predict(x_test)
```

In [45]:
```
from sklearn.metrics import confusion_matrix,accuracy_score
```

In [46]:
```
confusion_matrix(y_test,y_pred_svm)
```

Out[46]:
```
array([[3782,   37],
       [ 119,   62]], dtype=int64)
```

In [47]:
```
accuracy_score(y_test,y_pred_svm)
```

Out[47]:
```
0.961
```

In [48]:
```
svM.n_support_
```

Out[48]:
```
array([736, 728])
```

In [49]:
```
kernel=["linear",'rbf','poly','sigmoid']
```

In [50]:
```
for i in kernel:
    new_model=SVC(kernel=i)
    new_model.fit(x_train,y_train)
    print("kernel :",i)
    print("acc score :" ,accuracy_score(y_test,new_model.predict(x_test)))
```

```
kernel : linear
acc score : 0.961
kernel : rbf
acc score : 0.9615
kernel : poly
acc score : 0.961
kernel : sigmoid
acc score : 0.957
```

rbf and linear gives the high accuracy of 96%

In [ ]: