6) Design and implement SVM for classification with the proper dataset of your choice commenton design and implementation for linearly non sepearble datsset

In [6]:
```
#importing required libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

In [7]:
```
train= pd.read_csv("SalaryData_Train.csv")
```

In [8]:
```
test= pd.read_csv("SalaryData_Test.csv")
```

In [9]:
```
train.head()
```

Out[9]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	ca
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	

In [10]:
```
train.describe()
```

Out[10]:

	age	educationno	capitalgain	capitalloss	hoursperweek
count	30161.000000	30161.000000	30161.000000	30161.000000	30161.000000
mean	38.438115	10.121316	1092.044064	88.302311	40.931269
std	13.134830	2.550037	7406.466611	404.121321	11.980182
min	17.000000	1.000000	0.000000	0.000000	1.000000
25%	28.000000	9.000000	0.000000	0.000000	40.000000
50%	37.000000	10.000000	0.000000	0.000000	40.000000
75%	47.000000	13.000000	0.000000	0.000000	45.000000
max	90.000000	16.000000	99999.000000	4356.000000	99.000000

In [11]:
```
train.describe(include="all")
```

Out[11]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	ra
count	30161.000000	30161	30161	30161.000000	30161	30161	30161	301
unique	NaN	7	16	NaN	7	14	6	
top	NaN	Private	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband	Wh
freq	NaN	22285	9840	NaN	14065	4038	12463	259
mean	38.438115	NaN	NaN	10.121316	NaN	NaN	NaN	N
std	13.134830	NaN	NaN	2.550037	NaN	NaN	NaN	N
min	17.000000	NaN	NaN	1.000000	NaN	NaN	NaN	N
25%	28.000000	NaN	NaN	9.000000	NaN	NaN	NaN	N
50%	37.000000	NaN	NaN	10.000000	NaN	NaN	NaN	N
75%	47.000000	NaN	NaN	13.000000	NaN	NaN	NaN	N
max	90.000000	NaN	NaN	16.000000	NaN	NaN	NaN	N

In [12]:

```
#no missing data
```

In [13]:

```
from sklearn.preprocessing import LabelEncoder
lb = LabelEncoder()
train.education = lb.fit_transform(train.education)
test.education = lb.fit_transform(test.education)
```

In [14]:

```
train.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 30161 entries, 0 to 30160
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            30161 non-null  int64 
 1   workclass      30161 non-null  object
 2   education      30161 non-null  int32 
 3   educationno    30161 non-null  int64 
 4   maritalstatus  30161 non-null  object
 5   occupation     30161 non-null  object
 6   relationship   30161 non-null  object
 7   race           30161 non-null  object
 8   sex            30161 non-null  object
 9   capitalgain    30161 non-null  int64 
 10  capitalloss    30161 non-null  int64 
 11  hoursperweek   30161 non-null  int64 
 12  native         30161 non-null  object
 13  Salary         30161 non-null  object
dtypes: int32(1), int64(5), object(8)
memory usage: 3.1+ MB
```

In [15]:

```
train = pd.get_dummies(train,columns=["workclass","maritalstatus","occupation","relati
```

In [16]:

```
test = pd.get_dummies(test,columns=["workclass","maritalstatus","occupation","relatior
```

In [17]:

```
#checking categories in Salary column
train.Salary.value_counts()
```

Out[17]:

```
 <=50K    22653
 >50K      7508
Name: Salary, dtype: int64
```

In [18]:

```
x_train = train.drop("Salary",axis=1)
x_test = test.drop("Salary",axis = 1)
y_train = train.Salary
y_test = test.Salary
```

Linear model

In [24]:

```
from sklearn.svm import SVC
model1 = SVC(kernel="linear",max_iter=100000)
```

In [25]:

```
model1.fit(x_train,y_train)
```

```
C:\Users\theas\anaconda3\lib\site-packages\sklearn\svm\_base.py:255: ConvergenceWarni
ng: Solver terminated early (max_iter=100000).  Consider pre-processing your data wit
h StandardScaler or MinMaxScaler.
  warnings.warn('Solver terminated early (max_iter=%i).'
```

Out[25]:

```
SVC(kernel='linear', max_iter=100000)
```

In [36]:

```
test_pred = model1.predict(x_test)
```

In [56]:

```
linear_accuracy = np.mean(y_test == test_pred)
linear_accuracy
```

Out[56]:

```
0.2353253652058433
```

rgf Model

In [52]:

```
model2 = SVC(kernel="rbf",max_iter=150000)
model2.fit(x_train,y_train)
```

Out[52]:

```
SVC(max_iter=150000)
```

In [53]:

```
rbf_pred=model2.predict(x_test)
```

In [54]:

```
rbf_accuracy = np.mean(y_test == rbf_pred)
rbf_accuracy
```

Out[54]:

```
0.7964143426294821
```

Poly Model

In [40]:

```
model3 = SVC(kernel="poly",max_iter=100000)
model3.fit(x_train,y_train)
```

Out[40]:

```
SVC(kernel='poly', max_iter=100000)
```

In [42]:

```
poly_pred = model3.predict(x_test)
```

In [43]:

```
poly_accuracy = np.mean(y_test == poly_pred)
poly_accuracy
```

Out[43]:

```
0.7795484727755644
```

Sigmoid Model

In [44]:

```
model4 = SVC(kernel="sigmoid",max_iter=100000)
model4.fit(x_train,y_train)
```

Out[44]:

```
SVC(kernel='sigmoid', max_iter=100000)
```

In [45]:

```
sigmoid_pred=model4.predict(x_test)
```

In [46]:

```
sig_accuracy = np.mean(y_test == sigmoid_pred)
sig_accuracy
```

Out[46]:

```
0.7567729083665339
```

In [57]:

```
results = pd.DataFrame({"linear_model": linear_accuracy,"rbf_model": rbf_accuracy,"pol
```

In [58]:

```
results
```

Out[58]:

	linear_model	rbf_model	poly_accuracy	sigmoid_accuracy
Accuracy	0.235325	0.796414	0.779548	0.756773