Problem Statement - Prepare a classification model using SVM for salary data ¶

1. Import Necessary Libraries

In [6]:
```
import numpy as np
import pandas as pd
import tensorflow as tf
import seaborn as sns
```

2. Import Data

In [7]:
```
Salary_data_train = pd.read_csv('SalaryData_Train(1).csv')
Salary_data_test = pd.read_csv('SalaryData_Test(1).csv')
Salary_data_train.columns
Salary_data_test.columns
String_columns = ['workclass', 'education', 'maritalstatus', 'occupation', 'relat
```

In [8]:
```
Salary_data_test.columns
```

Out[8]:
```
Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
       'occupation', 'relationship', 'race', 'sex', 'capitalgain',
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object')
```

In [9]:
```
Salary_data_test
```

Out[9]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White
...	...	...	...	...	...	...	...	...
15055	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White
15056	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White
15057	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White
15058	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander
15059	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White

15060 rows × 14 columns

In [10]:
```
Salary_data_train
```

Out[10]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	s
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	M
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	M
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	M
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	M
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Fem
...	...	...	...	...	...	...	...	...	
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Fem
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	M
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Fem
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	M
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Fem

30161 rows × 14 columns

In [11]:
```
Salary_data_train.columns
```

Out[11]:
```
Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
       'occupation', 'relationship', 'race', 'sex', 'capitalgain',
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object')
```

3. Data Understanding

In [12]:
```
Salary_data_train.shape
```

Out[12]:
```
(30161, 14)
```

In [13]:
```
Salary_data_test.shape
```

Out[13]:
```
(15060, 14)
```

In [14]:
```
Salary_data_train.dtypes,Salary_data_test.dtypes
```

Out[14]:
```
(age              int64
 workclass       object
 education       object
 educationno      int64
 maritalstatus   object
 occupation      object
 relationship    object
 race            object
 sex             object
 capitalgain      int64
 capitalloss      int64
 hoursperweek     int64
 native          object
 Salary          object
 dtype: object,
 age              int64
 workclass       object
 education       object
 educationno      int64
 maritalstatus   object
 occupation      object
 relationship    object
 race            object
 sex             object
 capitalgain      int64
 capitalloss      int64
 hoursperweek     int64
 native          object
 Salary          object
 dtype: object)
```

In [15]:
```
Salary_data_train.describe(),Salary_data_test.describe()
```

Out[15]:
```
(                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,
                age   educationno   capitalgain   capitalloss  hoursperweek
 count  15060.000000  15060.000000  15060.000000  15060.000000  15060.000000
 mean      38.768327     10.112749   1120.301594     89.041899     40.951594
 std       13.380676      2.558727   7703.181842    406.283245     12.062831
 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%       48.000000     13.000000      0.000000      0.000000     45.000000
 max       90.000000     16.000000  99999.000000   3770.000000     99.000000)
```

4. Data Preparation

In [16]:
```
from sklearn.preprocessing import LabelEncoder
```

In [17]:
```
LabelEncoder = LabelEncoder()
```

In [18]:
```
for i in String_columns:
    Salary_data_train[i] = LabelEncoder.fit_transform(Salary_data_train[i])
    Salary_data_test[i] = LabelEncoder.fit_transform(Salary_data_test[i])
```

In [19]:
```
Salary_data_train
```

Out[19]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	39	5	9	13	4	0	1	4	1
1	50	4	9	13	2	3	0	4	1
2	38	2	11	9	0	5	1	4	1
3	53	2	1	7	2	5	0	2	1
4	28	2	9	13	2	9	5	2	0
...	...	...	...	...	...	...	...	...	...
30156	27	2	7	12	2	12	5	4	0
30157	40	2	11	9	2	6	0	4	1
30158	58	2	11	9	6	0	4	4	0
30159	22	2	11	9	4	0	3	4	1
30160	52	3	11	9	2	3	5	4	0

30161 rows × 14 columns

In [20]:
```
Salary_data_test
```

Out[20]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	25	2	1	7	4	6	3	2	1
1	38	2	11	9	2	4	0	4	1
2	28	1	7	12	2	10	0	4	1
3	44	2	15	10	2	6	0	2	1
4	34	2	0	6	4	7	1	4	1
...	...	...	...	...	...	...	...	...	...
15055	33	2	9	13	4	9	3	4	1
15056	39	2	9	13	0	9	1	4	0
15057	38	2	9	13	2	9	0	4	1
15058	44	2	9	13	0	0	3	1	1
15059	35	3	9	13	2	3	0	4	1

15060 rows × 14 columns

In [21]:
```
Salary_data_train.dtypes,Salary_data_test.dtypes
```

Out[21]:
```
(age              int64
 workclass        int32
 education        int32
 educationno      int64
 maritalstatus    int32
 occupation       int32
 relationship     int32
 race             int32
 sex              int32
 capitalgain      int64
 capitalloss      int64
 hoursperweek     int64
 native           int32
 Salary           int32
 dtype: object,
 age              int64
 workclass        int32
 education        int32
 educationno      int64
 maritalstatus    int32
 occupation       int32
 relationship     int32
 race             int32
 sex              int32
 capitalgain      int64
 capitalloss      int64
 hoursperweek     int64
 native           int32
 Salary           int32
 dtype: object)
```

5. Model Building

In [22]:
```
X_train=Salary_data_train.iloc[0:500,0:13]
y_train=Salary_data_train.iloc[0:500,13]
X_test=Salary_data_test.iloc[0:300,0:13]
y_test=Salary_data_test.iloc[0:300,13]
```

In [23]:
```
X_train
```

Out[23]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	ca
0	39	5	9	13	4	0	1	4	1	
1	50	4	9	13	2	3	0	4	1	
2	38	2	11	9	0	5	1	4	1	
3	53	2	1	7	2	5	0	2	1	
4	28	2	9	13	2	9	5	2	0	
...	...	...	...	...	...	...	...	...	...	
495	31	3	9	13	4	4	1	4	0	
496	44	2	15	10	2	13	0	4	1	
497	29	2	10	16	4	9	3	4	1	
498	30	2	15	10	4	5	4	4	0	
499	27	2	11	9	4	13	1	4	1	

500 rows × 13 columns

In [24]:
```
y_train
```

Out[24]:
```
0      0
1      0
2      0
3      0
4      0
      ..
495    0
496    0
497    0
498    0
499    0
Name: Salary, Length: 500, dtype: int32
```

In [25]:
```
X_test
```

Out[25]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	ca
0	25	2	1	7	4	6	3	2	1	
1	38	2	11	9	2	4	0	4	1	
2	28	1	7	12	2	10	0	4	1	
3	44	2	15	10	2	6	0	2	1	
4	34	2	0	6	4	7	1	4	1	
...	...	...	...	...	...	...	...	...	...	
295	56	3	9	13	2	3	0	4	1	
296	37	2	11	9	2	2	0	4	1	
297	52	2	3	2	2	6	0	4	1	
298	26	2	11	9	5	5	3	4	1	
299	33	2	12	14	2	9	0	4	1	

300 rows × 13 columns

In [26]:
```
y_test
```

Out[26]:
```
0      0
1      0
2      1
3      1
4      0
      ..
295    0
296    0
297    0
298    0
299    0
Name: Salary, Length: 300, dtype: int32
```

6. Model Training, Testing & Evaluation

In [35]:
```
from sklearn.svm import SVC
```

In [36]:
```
model_linear =SVC(kernel = 'linear')
model_linear.fit(X_train,y_train)
train_pred_lin = model_linear.predict(X_train)
test_pred_lin = model_linear.predict(X_test)
train_lin_acc = np.mean(train_pred_lin==y_train)
test_lin_acc= np.mean(test_pred_lin==y_test)
print("Accuracy Of train-data using Linear", train_lin_acc)
print("Accurancy of test-data using Linear",test_lin_acc )
```

```
Accuracy Of train-data using Linear 0.818
Accurancy of test-data using Linear 0.8166666666666667
```

In [37]:
```
model_poly=SVC(kernel='poly')
model_poly.fit(X_train,y_train)
train_pred_poly=model_poly.predict(X_train)
test_pred_poly=model_poly.predict(X_test)
train_poly_acc=np.mean(train_pred_poly==y_train)
test_poly_acc=np.mean(test_pred_poly==y_test)
print("Accuracy of train-data using POLY",train_poly_acc)
print("Accuracy of test-data using POLY",test_poly_acc)
```

```
Accuracy of train-data using POLY 0.812
Accuracy of test-data using POLY 0.8033333333333333
```

In [38]:
```
model_rbf=SVC(kernel='rbf')
model_rbf.fit(X_train,y_train)
train_pred_rbf=model_rbf.predict(X_train)
test_pred_rbf=model_rbf.predict(X_test)
train_rbf_acc=np.mean(train_pred_rbf==y_train)
test_rbf_acc=np.mean(test_pred_rbf==y_test)
print("Accuracy of train-data using rbf",train_rbf_acc)
print("Accuracy of test-data using rbf",test_rbf_acc)
```

```
Accuracy of train-data using rbf 0.812
Accuracy of test-data using rbf 0.8033333333333333
```

Conclusion: The accuracy of model is good with linear kernel tric