SVM Salary (train_data)

In [1]:

```
import pandas as pd
import numpy as np
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV
from sklearn.metrics import classification_report
from sklearn.metrics import accuracy_score, confusion_matrix
from sklearn.model_selection import train_test_split, cross_val_score
```

In [2]:

```
salary=pd.read_csv("C:/Users/Hp/Downloads/SalaryData_Train(1).csv")
```

In [3]:

```
salary.head()
```

Out[3]:

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

In [4]:

```
salary.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  object
 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: int64(5), object(9)
memory usage: 3.2+ MB
```

In [5]:

```
salary['workclass']=salary['workclass'].astype('category')
salary['education']=salary['education'].astype('category')
salary['maritalstatus']=salary['maritalstatus'].astype('category')
salary['occupation']=salary['occupation'].astype('category')
salary['relationship']=salary['relationship'].astype('category')
salary['race']=salary['race'].astype('category')
salary['native']=salary['native'].astype('category')
salary['sex']=salary['sex'].astype('category')
```

In [6]:

```
salary.dtypes
```

Out[6]:

```
age                int64
workclass       category
education       category
educationno        int64
maritalstatus   category
occupation      category
relationship    category
race            category
sex             category
capitalgain        int64
capitalloss        int64
hoursperweek       int64
native          category
Salary            object
dtype: object
```

In [7]:

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
```

we need Salary string type data into binary numbers

In [8]:

```
salary['Salary'] = label_encoder.fit_transform(salary['Salary'])
```

In [9]:

```
salary.Salary
```

Out[9]:

```
0        0
1        0
2        0
3        0
4        0
        ..
30156    0
30157    1
30158    0
30159    0
30160    1
Name: Salary, Length: 30161, dtype: int32
```

we also need to convert categories into numbers

In [11]:

```
salary['workclass'] = label_encoder.fit_transform(salary['workclass'])
salary['education'] = label_encoder.fit_transform(salary['education'])
salary['maritalstatus'] = label_encoder.fit_transform(salary['maritalstatus'])
salary['occupation'] = label_encoder.fit_transform(salary['occupation'])
salary['relationship'] = label_encoder.fit_transform(salary['relationship'])
salary['race'] = label_encoder.fit_transform(salary['race'])
salary['sex'] = label_encoder.fit_transform(salary['sex'])
salary['native'] = label_encoder.fit_transform(salary['native'])
```

In [12]:

```
salary
```

Out[12]:

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

30161 rows × 14 columns

In [16]:

```
#Splitting the data into x and y as input and output

X = salary.iloc[:,0:13]
Y = salary.iloc[:,13]
```

In [19]:

```
X
```

Out[19]:

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

30161 rows × 13 columns

In [20]:

```
Y
```

Out[20]:

```
0         0
1         0
2         0
3         0
4         0
         ..
30156     0
30157     1
30158     0
30159     0
30160     1
Name: Salary, Length: 30161, dtype: int32
```

In [21]:

```
salary.Salary.unique()
```

Out[21]:

```
array([0, 1])
```

In [22]:

```
salary.Salary.value_counts()
```

Out[22]:

```
0    22653
1     7508
Name: Salary, dtype: int64
```

In [23]:

```
# Splitting the data into training and test dataset

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3, random_state=0)
```

In [24]:

```
clf=SVC()
clf.fit(x_train , y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy =", acc)
confusion_matrix(y_test, y_pred)
```

```
Accuracy = 79.23527461597966
```

Out[24]:

```
array([[6580,  218],
       [1661,  590]], dtype=int64)
```

In [25]:

```
y_pred=clf.predict(x_test)
```

In [26]:

```
y_pred
```

Out[26]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

SVM Salary (test_data)

In [27]:

```
salary=pd.read_csv("C:/Users/Hp/Downloads/SalaryData_Test(1).csv")
```

In [28]:

```
salary.head()
```

Out[28]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male

In [29]:

```
salary.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 15060 entries, 0 to 15059
Data columns (total 14 columns):
 #   Column         Non-Null Count  Dtype 
---  ------         --------------  ----- 
 0   age            15060 non-null  int64 
 1   workclass      15060 non-null  object
 2   education      15060 non-null  object
 3   educationno    15060 non-null  int64 
 4   maritalstatus  15060 non-null  object
 5   occupation     15060 non-null  object
 6   relationship   15060 non-null  object
 7   race           15060 non-null  object
 8   sex            15060 non-null  object
 9   capitalgain    15060 non-null  int64 
 10  capitalloss    15060 non-null  int64 
 11  hoursperweek   15060 non-null  int64 
 12  native         15060 non-null  object
 13  Salary         15060 non-null  object
dtypes: int64(5), object(9)
memory usage: 1.6+ MB
```

In [30]:

```
salary['workclass']=salary['workclass'].astype('category')
salary['education']=salary['education'].astype('category')
salary['maritalstatus']=salary['maritalstatus'].astype('category')
salary['occupation']=salary['occupation'].astype('category')
salary['relationship']=salary['relationship'].astype('category')
salary['race']=salary['race'].astype('category')
salary['native']=salary['native'].astype('category')
salary['sex']=salary['sex'].astype('category')
```

In [31]:

```
salary.dtypes
```

Out[31]:

```
age                int64
workclass       category
education       category
educationno        int64
maritalstatus   category
occupation      category
relationship    category
race            category
sex             category
capitalgain        int64
capitalloss        int64
hoursperweek       int64
native          category
Salary            object
dtype: object
```

In [32]:

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()
```

In [33]:

```
salary['Salary'] = label_encoder.fit_transform(salary['Salary'])
```

In [34]:

```
salary.Salary
```

Out[34]:

```
0        0
1        0
2        1
3        1
4        0
        ..
15055    0
15056    0
15057    0
15058    0
15059    1
Name: Salary, Length: 15060, dtype: int32
```

In [35]:

```
#we also need to convert categories into numbers
salary['workclass'] = label_encoder.fit_transform(salary['workclass'])
salary['education'] = label_encoder.fit_transform(salary['education'])
salary['maritalstatus'] = label_encoder.fit_transform(salary['maritalstatus'])
salary['occupation'] = label_encoder.fit_transform(salary['occupation'])
salary['relationship'] = label_encoder.fit_transform(salary['relationship'])
salary['race'] = label_encoder.fit_transform(salary['race'])
salary['sex'] = label_encoder.fit_transform(salary['sex'])
salary['native'] = label_encoder.fit_transform(salary['native'])
salary
```

Out[35]:

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

15060 rows × 14 columns

In [36]:

```
#define x&y

X = salary.iloc[:,0:13]
Y = salary.iloc[:,13]
```

In [37]:

```
X
```

Out[37]:

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

15060 rows × 13 columns

In [38]:

```
Y
```

Out[38]:

```
0         0
1         0
2         1
3         1
4         0
        ..
15055     0
15056     0
15057     0
15058     0
15059     1
Name: Salary, Length: 15060, dtype: int32
```

In [40]:

```
salary.Salary.unique()
```

Out[40]:

```
array([0, 1])
```

In [42]:

```
salary.Salary.value_counts()
```

Out[42]:

```
0    11360
1     3700
Name: Salary, dtype: int64
```

In [43]:

```
# Splitting the data into training and test dataset

x_train, x_test, y_train, y_test = train_test_split(X,Y, test_size=0.3, random_state=0)
```

In [44]:

```
#model building by using SVM
clf=SVC()
clf.fit(x_train , y_train)
y_pred = clf.predict(x_test)
acc = accuracy_score(y_test, y_pred) * 100
print("Accuracy =", acc)
confusion_matrix(y_test, y_pred)
```

```
Accuracy = 79.6812749003984
```

Out[44]:

```
array([[3290,   95],
       [ 823,  310]], dtype=int64)
```

In [45]:

```
y_pred=clf.predict(x_test)
y_pred
```

Out[45]:

```
array([0, 0, 0, ..., 0, 0, 0])
```

In [ ]:

```

```