Support Vector Machines Assignment

Data Set - Salary_data

1. Import Necessary libraries

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

from matplotlib import pyplot as plt
import seaborn as sns

import warnings
warnings.filterwarnings('ignore')
```

2. Import Data

In [2]:
```
test_data = pd.read_csv('SalaryData_Test(1).csv')
test_data
```

Out[2]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
15055	33	Private	Bachelors	13	Never-married	Prof-specialty	Own-child	White	Male	0	0	40	United-States	<=50K
15056	39	Private	Bachelors	13	Divorced	Prof-specialty	Not-in-family	White	Female	0	0	36	United-States	<=50K
15057	38	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Husband	White	Male	0	0	50	United-States	<=50K
15058	44	Private	Bachelors	13	Divorced	Adm-clerical	Own-child	Asian-Pac-Islander	Male	5455	0	40	United-States	<=50K
15059	35	Self-emp-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	60	United-States	>50K

15060 rows × 14 columns

In [3]:
```
train_data = pd.read_csv('SalaryData_Train(1).csv')
train_data
```

Out[3]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
30156	27	Private	Assoc-acdm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	38	United-States	<=50K
30157	40	Private	HS-grad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	40	United-States	>50K
30158	58	Private	HS-grad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	40	United-States	<=50K
30159	22	Private	HS-grad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	20	United-States	<=50K
30160	52	Self-emp-inc	HS-grad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	40	United-States	>50K

30161 rows × 14 columns

3. Data Understanding

3.1 Initial Analysis :

a) For Test data

In [4]:
```
test_data.head()
```

Out[4]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	25	Private	11th	7	Never-married	Machine-op-inspct	Own-child	Black	Male	0	0	40	United-States	<=50K
1	38	Private	HS-grad	9	Married-civ-spouse	Farming-fishing	Husband	White	Male	0	0	50	United-States	<=50K
2	28	Local-gov	Assoc-acdm	12	Married-civ-spouse	Protective-serv	Husband	White	Male	0	0	40	United-States	>50K
3	44	Private	Some-college	10	Married-civ-spouse	Machine-op-inspct	Husband	Black	Male	7688	0	40	United-States	>50K
4	34	Private	10th	6	Never-married	Other-service	Not-in-family	White	Male	0	0	30	United-States	<=50K

In [5]:
```
test_data.shape
```

Out[5]:
```
(15060, 14)
```

In [6]:
```
test_data.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 [7]:
```
test_data.isna().sum()
```

Out[7]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

In [8]:
```
test_data.describe()
```

Out[8]:

	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

In [9]:
```
test_data.dtypes
```

Out[9]:
```
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 [10]:
```
test_data.columns
```

Out[10]:
```
Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
       'occupation', 'relationship', 'race', 'sex', 'capitalgain',
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object')
```

b) For Train data

In [11]:
```
train_data.head()
```

Out[11]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	39	State-gov	Bachelors	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	40	United-States	<=50K
1	50	Self-emp-not-inc	Bachelors	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	13	United-States	<=50K
2	38	Private	HS-grad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	40	United-States	<=50K
3	53	Private	11th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	40	United-States	<=50K
4	28	Private	Bachelors	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	40	Cuba	<=50K

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

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

In [13]:
```
train_data.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 [14]:
```
train_data.isna().sum()
```

Out[14]:
```
age              0
workclass        0
education        0
educationno      0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
capitalgain      0
capitalloss      0
hoursperweek     0
native           0
Salary           0
dtype: int64
```

In [15]:
```
train_data.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

In [16]:
```
train_data.dtypes
```

Out[16]:
```
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 [17]:
```
train_data.columns
```

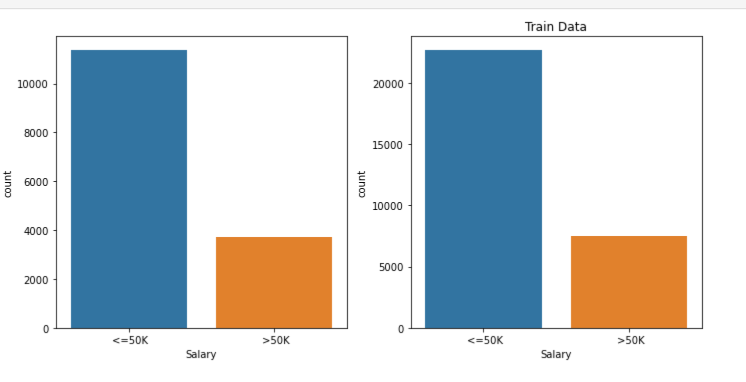
Out[17]:
```
Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
       'occupation', 'relationship', 'race', 'sex', 'capitalgain',
       'capitalloss', 'hoursperweek', 'native', 'Salary'],
      dtype='object')
```

3.2 Visualization using countplot :

In [18]:
```
fig, ax = plt.subplots(1,2,figsize = (10,5))

sns.countplot(test_data.Salary, ax = ax[0])
sns.countplot(train_data.Salary, ax = ax[1])
plt.title('Train Data')

plt.tight_layout()
plt.show()
```



3.3 Lable Encoder :

In [19]:
```
from sklearn.preprocessing import LabelEncoder
```

a) For Train data

In [20]:
```
train_data = train_data.apply(LabelEncoder().fit_transform)
train_data.head()
```

Out[20]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	22	5	9	12	4	0	1	4	1	24	0	39	37	0
1	33	4	9	12	2	3	0	4	1	0	0	12	37	0
2	21	2	11	8	0	5	1	4	1	0	0	39	37	0
3	36	2	1	6	2	5	0	2	1	0	0	39	37	0
4	11	2	9	12	2	9	5	2	0	0	0	39	4	0

b) For Test data

In [21]:
```
test_data = test_data.apply(LabelEncoder().fit_transform)
test_data.head()
```

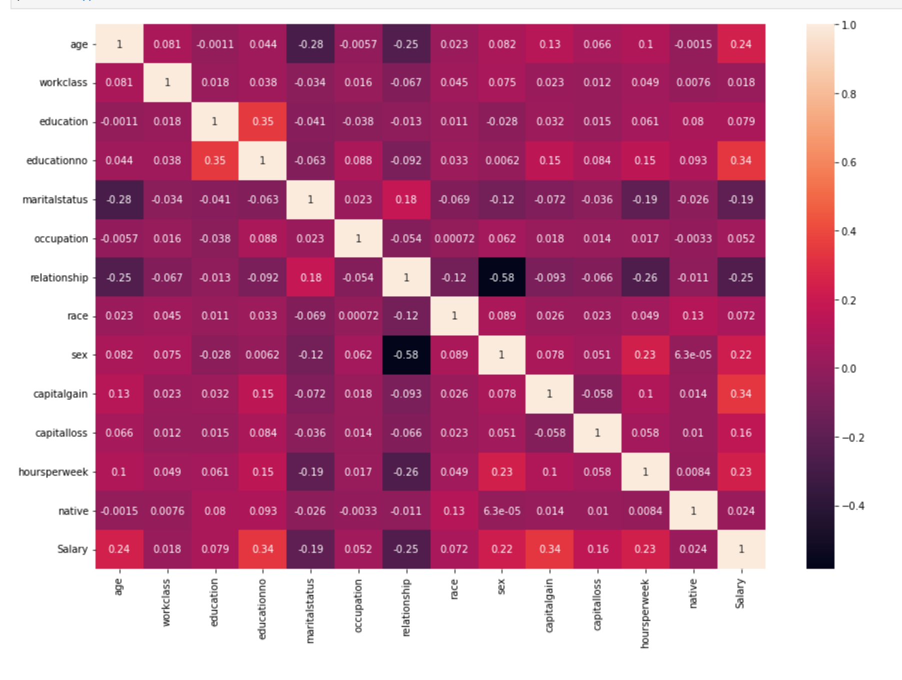
Out[21]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	8	2	1	6	4	6	3	2	1	0	0	39	37	0
1	21	2	11	8	2	4	0	4	1	0	0	49	37	0
2	11	1	7	11	2	10	0	4	1	0	0	39	37	1
3	27	2	15	9	2	6	0	2	1	87	0	39	37	1
4	17	2	0	5	4	7	1	4	1	0	0	29	37	0

3.4 Correlation Matrix :

a) For Train data

In [22]:
```
plt.figure(figsize = (15,10))
sns.heatmap(train_data.corr(),annot = True)
plt.show()
```



b) For Test data

In [23]:
```
plt.figure(figsize = (15,10))
sns.heatmap(test_data.corr(),annot = True)
plt.show()
```

4. Extrating the independent and dependent variables

a) For Train data

In [24]:
```
X_train = train_data.drop(['workclass','education','relationship','native','maritalstatus','sex','race'],axis = 1)
X_train.head()
```

Out[24]:

	age	educationno	occupation	capitalgain	capitalloss	hoursperweek	Salary
0	22	12	0	24	0	39	0
1	33	12	3	0	0	12	0
2	21	8	5	0	0	39	0
3	36	6	5	0	0	39	0
4	11	12	9	0	0	39	0

In [25]:
```
Y_train = train_data["Salary"]
Y_train
```

Out[25]:
```
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 [26]:
```
X_train.shape, Y_train.shape
```

Out[26]:
```
((30161, 7), (30161,))
```

b) For Test data

In [27]:
```
X_test = test_data.drop(['workclass','education','relationship','native','maritalstatus','sex','race'],axis = 1)
X_test.head()
```

Out[27]:

	age	educationno	occupation	capitalgain	capitalloss	hoursperweek	Salary
0	8	6	6	0	0	39	0
1	21	8	4	0	0	49	0
2	11	11	10	0	0	39	1
3	27	9	6	87	0	39	1
4	17	5	7	0	0	29	0

In [28]:
```
Y_test = test_data["Salary"]
Y_test
```

Out[28]:
```
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 [29]:
```
X_test.shape, Y_test.shape
```

Out[29]:
```
((15060, 7), (15060,))
```

5. SVM with Kernel rbf

In [30]:
```
from sklearn.preprocessing import StandardScaler
from sklearn import svm
from sklearn.svm import SVC
from sklearn.metrics import accuracy_score, confusion_matrix,classification_report
```

In [31]:
```
clf = SVC()
```

In [32]:
```
clf = SVC(kernel = 'rbf')
clf.fit(X_train , Y_train)

y_pred = clf.predict(X_test)
```

In [33]:
```
acc = accuracy_score(Y_test, y_pred) * 100
print('Accuracy For Kernal rbf :', acc)
```

```
Accuracy For Kernal rbf : 98.56573705179282
```

In [34]:
```
confusion_matrix(Y_test, y_pred)
```

Out[34]:
```
array([[11293,    67],
       [  149,  3551]], dtype=int64)
```

6. Linear Support Vector Machine

In [35]:
```
svc = SVC(gamma = 0.22)
svc.fit(X_train, Y_train)

score_svc = svc.score(X_test, Y_test)
print('The accuracy of Linear SVC :', score_svc)
```

```
The accuracy of Linear SVC : 0.9072377158034528
```

7. SVM with Kernel poly

In [36]:
```
clf = SVC(kernel = 'poly',C = 10, gamma = 0.1)
clf.fit(X_train , Y_train)

y_pred = clf.predict(X_test)
```

In [37]:
```
acc = accuracy_score(Y_test, y_pred) * 100
print('Accuracy For Kernal Poly :', acc)
```

```
Accuracy For Kernal Poly : 99.99335989375831
```

In [38]:
```
confusion_matrix(Y_test, y_pred)
```

Out[38]:
```
array([[11360,     0],
       [    1,  3699]], dtype=int64)
```

8. SVM with Kernel sigmoid

In [39]:
```
clf = SVC(kernel = 'sigmoid',C = 10, gamma = 0.1)
clf.fit(X_train , Y_train)

y_pred = clf.predict(X_test)
```

In [40]:
```
acc = accuracy_score(Y_test, y_pred) * 100
print('Accuracy For Kernal Sigmoid :', acc)
```

```
Accuracy For Kernal Sigmoid : 75.41832669322709
```

In [41]:
```
confusion_matrix(Y_test, y_pred)
```

Out[41]:
```
array([[11358,     2],
       [ 3700,     0]], dtype=int64)
```

Conclusion :

```
The Accuracy Model For Kernal rbf     : 98.56%
The Accuracy Model For Linear SVM     : 90.72%
The Accuracy Model For Kernal Poly    : 99.99%
The Accuracy Model For Kernal Sigmoid : 75.41%
```

SVM with Kernel Poly have the Best Model Accuracy with 99.99%