Problem Statement:

Prepare a classification model using Naive Bayes for salary data test 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
from sklearn.preprocessing import LabelEncoder
```

2. Import Data

In [3]:
```
Naive_bayes_Test_Data = pd.read_csv('SalaryData_Test.csv')
Naive_bayes_Test_Data
```

Out[3]:

	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	Fe
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

3. Data Understanding

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

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

In [6]:
```
Naive_bayes_Test_Data.dtypes
```

Out[6]:
```
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 [7]:
```
Naive_bayes_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]:
```
Naive_bayes_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 [9]:
```
Naive_bayes_Test_Data.describe(include='all')
```

Out[9]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship
count	15060.000000	15060	15060	15060.000000	15060	15060	15060
unique	NaN	7	16	NaN	7	14	6
top	NaN	Private	HS-grad	NaN	Married-civ-spouse	Exec-managerial	Husband
freq	NaN	11021	4943	NaN	6990	1992	6203
mean	38.768327	NaN	NaN	10.112749	NaN	NaN	NaN
std	13.380676	NaN	NaN	2.558727	NaN	NaN	NaN
min	17.000000	NaN	NaN	1.000000	NaN	NaN	NaN
25%	28.000000	NaN	NaN	9.000000	NaN	NaN	NaN
50%	37.000000	NaN	NaN	10.000000	NaN	NaN	NaN
75%	48.000000	NaN	NaN	13.000000	NaN	NaN	NaN
max	90.000000	NaN	NaN	16.000000	NaN	NaN	NaN

In [10]:
```
Naive_bayes_Test_Data.describe(include='all').nunique()
```

Out[10]:
```
age              8
workclass        4
education        4
educationno      8
maritalstatus    4
occupation       4
relationship     4
race             4
sex              4
capitalgain      5
capitalloss      5
hoursperweek     7
native           4
Salary           4
dtype: int64
```

In [12]:
```
Naive_bayes_Test_Data['workclass'].nunique()
```

Out[12]:
```
7
```

In [11]:
```
Naive_bayes_Test_Data['workclass'].unique()
```

Out[11]:
```
array([' Private', ' Local-gov', ' Self-emp-not-inc', ' Federal-gov',
       ' State-gov', ' Self-emp-inc', ' Without-pay'], dtype=object)
```

In [13]:
```
Naive_bayes_Test_Data
```

Out[13]:

	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

4. Data Preparation

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

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

In [18]:
```
Naive_bayes_Test_Data['workclass'] = Label.fit_transform(Naive_bayes_Test_Data['w
Naive_bayes_Test_Data['education'] = Label.fit_transform(Naive_bayes_Test_Data['e
Naive_bayes_Test_Data['maritalstatus'] = Label.fit_transform(Naive_bayes_Test_Dat
Naive_bayes_Test_Data['occupation'] = Label.fit_transform(Naive_bayes_Test_Data['
Naive_bayes_Test_Data['relationship'] = Label.fit_transform(Naive_bayes_Test_Data
Naive_bayes_Test_Data['race'] = Label.fit_transform(Naive_bayes_Test_Data['race']
Naive_bayes_Test_Data['sex'] = Label.fit_transform(Naive_bayes_Test_Data['sex'])
Naive_bayes_Test_Data['native'] = Label.fit_transform(Naive_bayes_Test_Data['nati
Naive_bayes_Test_Data['Salary'] = Label.fit_transform(Naive_bayes_Test_Data['Sala
```

In [19]:
```
Naive_bayes_Test_Data
```

Out[19]:

	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 [20]:
```
Naive_bayes_Test_Data.dtypes
```

Out[20]:
```
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 [21]:
```
X = Naive_bayes_Test_Data.drop(['Salary'], axis=1)
y = Naive_bayes_Test_Data['Salary']
```

In [22]:
```
X
```

Out[22]:

	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 × 13 columns

In [23]:
```
y
```

Out[23]:
```
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 [47]:
```
from sklearn.model_selection import train_test_split
```

In [49]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, train_size=0.20, random_
```

In [50]:
```
X_train.shape, y_train.shape
```

Out[50]:
```
((3012, 13), (3012,))
```

In [51]:
```
X_test.shape, y_test.shape
```

Out[51]:
```
((12048, 13), (12048,))
```

6. Model training | Testing | Evaluation

In [52]:
```
from sklearn.naive_bayes import MultinomialNB
nb_Classifier = MultinomialNB()
```

In [59]:
```
nb_Classifier.fit(X_test, y_test)
MultinomialNB()
```

Out[59]:
```
MultinomialNB()
```

In [60]:
```
y_pred = nb_Classifier.predict(X_train)
y_pred
```

Out[60]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [61]:
```
from sklearn.metrics import accuracy_score, precision_score,confusion_matrix, cla
```

In [63]:
```
print("Accuracy Score:", round(accuracy_score(y_train, y_pred),4))
```

```
Accuracy Score: 0.7915
```

In [64]:
```
print("Confusion Matrix:", confusion_matrix(y_train, y_pred))
```

```
Confusion Matrix: [[2241   75]
 [ 553  143]]
```

In [65]:
```
print("Classification Report:",  classification_report(y_train, y_pred))
```

```
Classification Report:               precision    recall  f1-score   support

           0       0.80      0.97      0.88      2316
           1       0.66      0.21      0.31       696

    accuracy                           0.79      3012
   macro avg       0.73      0.59      0.60      3012
weighted avg       0.77      0.79      0.75      3012
```

In [66]:
```
print("Precision Scoree:", round(precision_score(y_train, y_pred),4))
```

```
Precision Scoree: 0.656
```

7. Model Deployment

In [73]:
```
from pickle import dump
```

In [74]:
```
dump(nb_Classifier, open('lognb_Classifier.plktest', 'wb'))
```

In [75]:
```
from pickle import load
```

In [76]:
```
nb_Classifier_pickle = load(open('lognb_Classifier.plktest', 'rb'))
```

In [77]:
```
Pickle_pred = nb_Classifier_pickle.predict(X_train)
```

Conclusion:

Classification model using Naive Bayes for salary data train data with model accuracy 79%