Problem Statement:

Prepare a classification model using Naive Bayes for salary data train data

1. Import Necessary Libraries

In [2]:
```
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_Train_Data = pd.read_csv('SalaryData_Train.csv')
Naive_bayes_Train_Data
```

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	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

3. Data Understanding

In [4]:
```
Naive_bayes_Train_Data.shape
```

Out[4]:
```
(30161, 14)
```

In [5]:
```
Naive_bayes_Train_Data.dtypes
```

Out[5]:
```
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 [6]:
```
Naive_bayes_Train_Data.isna().sum()
```

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

Out[9]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship
count	30161.000000	30161	30161	30161.000000	30161	30161	30161
unique	NaN	7	16	NaN	7	14	6
top	NaN	Private	HS-grad	NaN	Married-civ-spouse	Prof-specialty	Husband
freq	NaN	22285	9840	NaN	14065	4038	12463
mean	38.438115	NaN	NaN	10.121316	NaN	NaN	NaN
std	13.134830	NaN	NaN	2.550037	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%	47.000000	NaN	NaN	13.000000	NaN	NaN	NaN
max	90.000000	NaN	NaN	16.000000	NaN	NaN	NaN

In [11]:
```
Naive_bayes_Train_Data.describe(include='all').nunique()
```

Out[11]:
```
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 [17]:
```
Naive_bayes_Train_Data['workclass'].nunique()
```

Out[17]:
```
7
```

In [18]:
```
Naive_bayes_Train_Data['workclass'].unique()
```

Out[18]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' Self-emp-inc', ' Without-pay'], dtype=object)
```

In [29]:
```
Naive_bayes_Train_Data
```

Out[29]:

ɔn	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hour
ırs	13	Never-married	Adm-clerical	Not-in-family	White	Male	2174	0	
ırs	13	Married-civ-spouse	Exec-managerial	Husband	White	Male	0	0	
ad	9	Divorced	Handlers-cleaners	Not-in-family	White	Male	0	0	
th	7	Married-civ-spouse	Handlers-cleaners	Husband	Black	Male	0	0	
ırs	13	Married-civ-spouse	Prof-specialty	Wife	Black	Female	0	0	
...	...	...	...	...	...	...	...	...	
ɔc-lm	12	Married-civ-spouse	Tech-support	Wife	White	Female	0	0	
ad	9	Married-civ-spouse	Machine-op-inspct	Husband	White	Male	0	0	
ad	9	Widowed	Adm-clerical	Unmarried	White	Female	0	0	
ad	9	Never-married	Adm-clerical	Own-child	White	Male	0	0	
ad	9	Married-civ-spouse	Exec-managerial	Wife	White	Female	15024	0	

4. Data Preparation

In [24]:
```
from sklearn.preprocessing import LabelEncoder
```

In [25]:
```
Label = LabelEncoder()
```

In [30]:
```
Naive_bayes_Train_Data['workclass'] = Label.fit_transform(Naive_bayes_Train_Data[
Naive_bayes_Train_Data['education'] = Label.fit_transform(Naive_bayes_Train_Data[
Naive_bayes_Train_Data['maritalstatus'] = Label.fit_transform(Naive_bayes_Train_D
Naive_bayes_Train_Data['occupation'] = Label.fit_transform(Naive_bayes_Train_Data
Naive_bayes_Train_Data['relationship'] = Label.fit_transform(Naive_bayes_Train_Da
Naive_bayes_Train_Data['race'] = Label.fit_transform(Naive_bayes_Train_Data['race
Naive_bayes_Train_Data['sex'] = Label.fit_transform(Naive_bayes_Train_Data['sex']
Naive_bayes_Train_Data['native'] = Label.fit_transform(Naive_bayes_Train_Data['na
Naive_bayes_Train_Data['Salary'] = Label.fit_transform(Naive_bayes_Train_Data['Sa
```

In [31]:
```
Naive_bayes_Train_Data
```

Out[31]:

	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 [32]:
```
Naive_bayes_Train_Data.dtypes
```

Out[32]:
```
age             int64
workclass       int64
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 [34]:
```
X = Naive_bayes_Train_Data.drop(['Salary'], axis=1)
y = Naive_bayes_Train_Data['Salary']
```

In [35]:
```
X
```

Out[35]:

class	education	educationno	maritalstatus	occupation	relationship	race	sex	capitalgain	capitall
5	9	13	4	0	1	4	1	2174	
4	9	13	2	3	0	4	1	0	
2	11	9	0	5	1	4	1	0	
2	1	7	2	5	0	2	1	0	
2	9	13	2	9	5	2	0	0	
...	...	...	...	...	...	...	...	...	
2	7	12	2	12	5	4	0	0	
2	11	9	2	6	0	4	1	0	
2	11	9	6	0	4	4	0	0	
2	11	9	4	0	3	4	1	0	
3	11	9	2	3	5	4	0	15024	

olumns

In [36]:
```
y
```

Out[36]:
```
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 [37]:
```
from sklearn.model_selection import train_test_split
```

In [57]:
```
X_train, X_test, y_train, y_test = train_test_split(X,y, test_size=0.20, random_s
```

In [58]:
```
X_train.shape, y_train.shape
```

Out[58]:
```
((24128, 13), (24128,))
```

In [59]:
```
X_test.shape, y_test.shape
```

Out[59]:
```
((6033, 13), (6033,))
```

6. Model training | Testing | Evaluation

In [60]:
```
from sklearn.naive_bayes import MultinomialNB
nb_Classifier = MultinomialNB()
```

In [62]:
```
nb_Classifier.fit(X_train, y_train)
MultinomialNB()
```

Out[62]:
```
MultinomialNB()
```

In [63]:
```
y_pred = nb_Classifier.predict(X_test)
y_pred
```

Out[63]:
```
array([0, 0, 0, ..., 0, 0, 0])
```

In [64]:
```
from sklearn.metrics import accuracy_score, precision_score,confusion_matrix, cla
```

In [71]:
```
print("Accuracy Score:", round(accuracy_score(y_test, y_pred),4))
```

```
Accuracy Score: 0.7721
```

In [73]:
```
print("Confusion Matrix:", confusion_matrix(y_test, y_pred))
```

```
Confusion Matrix: [[4358  199]
 [1176  300]]
```

In [81]:
```
print("Classification Report:",  classification_report(y_test, y_pred))
```

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

           0       0.79      0.96      0.86      4557
           1       0.60      0.20      0.30      1476

    accuracy                           0.77      6033
   macro avg       0.69      0.58      0.58      6033
weighted avg       0.74      0.77      0.73      6033
```

In [75]:
```
print("Precision Scoree:", round(precision_score(y_test, y_pred),4))
```

```
Precision Scoree: 0.6012
```

7. Model Deployment

In [90]:
```
from pickle import dump
```

In [91]:
```
dump(nb_Classifier, open('lognb_Classifier.plktrain', 'wb'))
```

In [92]:
```
from pickle import load
```

In [93]:
```
nb_Classifier_pickle = load(open('lognb_Classifier.plktrain', 'rb'))
```

In [94]:
```
Pickle_pred = nb_Classifier_pickle.predict(X_test)
```

Conclusion:

Classification model using Naive Bayes for salary data train data with model accuracy 77%