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

In [5]:
```python
salarydata_train=pd.read_csv("C:\\Users\\Admin\\Downloads\\naive bayes\\SalaryData_Train.csv")
salarydata_train.head()
```

Out[5]:

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

In [7]:
```python
salarydata_test = pd.read_csv("C:\\Users\\Admin\\Downloads\\naive bayes\\SalaryData_Test.csv")
salarydata_test.head()
```

Out[7]:

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

In [8]:
```
salarydata_train.shape
```

Out[8]:
```
(30161, 14)
```

In [9]:
```
salarydata_test.shape
```

Out[9]:
```
(15060, 14)
```

In [6]:
```
salarydata_train.isnull().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 [11]:
```
salarydata_train.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 [12]:
```
salarydata_train.describe()
```

Out[12]:

	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 [13]:
```
salarydata_test.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 [15]:
```
salarydata_test.describe()
```

Out[15]:

	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 [16]:
```
salarydata_train.isin(['?']).sum(axis=0)
```

Out[16]:
```
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 [17]:
```
salarydata_test.isin(['?']).sum(axis=0)
```

Out[17]:
```
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 [18]:
```
print(salarydata_train[0:5])
```

```
   age         workclass   education  educationno       maritalstatus  \
0   39         State-gov   Bachelors           13       Never-married
1   50  Self-emp-not-inc   Bachelors           13  Married-civ-spouse
2   38           Private     HS-grad            9            Divorced
3   53           Private        11th            7  Married-civ-spouse
4   28           Private   Bachelors           13  Married-civ-spouse

          occupation   relationship    race     sex  capitalgain  \
0       Adm-clerical  Not-in-family   White    Male         2174
1    Exec-managerial        Husband   White    Male            0
2  Handlers-cleaners  Not-in-family   White    Male            0
3  Handlers-cleaners        Husband   Black    Male            0
4     Prof-specialty           Wife   Black  Female            0

   capitalloss  hoursperweek         native  Salary
0            0            40  United-States   <=50K
1            0            13  United-States   <=50K
2            0            40  United-States   <=50K
3            0            40  United-States   <=50K
4            0            40           Cuba   <=50K
```

In [19]:
```
categorical = [var for var in salarydata_train.columns if salarydata_train[var].dtype=='O']

print('There are {} categorical variables\n'.format(len(categorical)))

print('The categorical variables are :\n\n', categorical)
```

```
There are 9 categorical variables

The categorical variables are :

 ['workclass', 'education', 'maritalstatus', 'occupation', 'relationship', 'race', 'sex', 'native', 'Salary']
```

In [20]:
```
salarydata_train[categorical].head()
```

Out[20]:

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

In [21]:
```
salarydata_train[categorical].isnull().sum()
```

Out[21]:
```
workclass        0
education        0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
native           0
Salary           0
dtype: int64
```

In [22]:
```
for var in categorical:

    print(salarydata_train[var].value_counts())
```

```
 Nicaragua                         33
 Peru                              30
 Greece                            29
 France                            27
 Ecuador                           27
 Ireland                           24
 Hong                              19
 Cambodia                          18
 Trinadad&Tobago                   18
 Laos                              17
 Thailand                          17
 Yugoslavia                        16
 Outlying-US(Guam-USVI-etc)        14
 Hungary                           13
 Honduras                          12
 Scotland                          11
Name: native, dtype: int64
 <=50K    22653
 >50K      7508
Name: Salary, dtype: int64
```

In [23]:
```
for var in categorical:

    print(salarydata_train[var].value_counts()/np.float(len(salarydata_train)))
```

```
 Hong                          0.000630
 Cambodia                      0.000597
 Trinadad&Tobago               0.000597
 Laos                          0.000564
 Thailand                      0.000564
 Yugoslavia                    0.000530
 Outlying-US(Guam-USVI-etc)    0.000464
 Hungary                       0.000431
 Honduras                      0.000398
 Scotland                      0.000365
Name: native, dtype: float64
 <=50K    0.751069
 >50K     0.248931
Name: Salary, dtype: float64

C:\Users\Admin\AppData\Local\Temp/ipykernel_432/217981199.py:3: DeprecationWarning: `np.float` is a deprecat
ed alias for the builtin `float`. To silence this warning, use `float` by itself. Doing this will not modify
any behavior and is safe. If you specifically wanted the numpy scalar type, use `np.float64` here.
Deprecated in NumPy 1.20; for more details and guidance: https://numpy.org/devdocs/release/1.20.0-notes.html
#deprecations (https://numpy.org/devdocs/release/1.20.0-notes.html#deprecations)
```

In [24]:
```
salarydata_train.workclass.unique()
```

Out[24]:
```
array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',
       ' Local-gov', ' Self-emp-inc', ' Without-pay'], dtype=object)
```

In [25]:
```
salarydata_train.workclass.value_counts()
```

Out[25]:
```
 Private             22285
 Self-emp-not-inc     2499
 Local-gov            2067
 State-gov            1279
 Self-emp-inc         1074
 Federal-gov           943
 Without-pay            14
Name: workclass, dtype: int64
```

In [26]:
```
salarydata_train.occupation.unique()
```

Out[26]:
```
array([' Adm-clerical', ' Exec-managerial', ' Handlers-cleaners',
       ' Prof-specialty', ' Other-service', ' Sales', ' Transport-moving',
       ' Farming-fishing', ' Machine-op-inspct', ' Tech-support',
       ' Craft-repair', ' Protective-serv', ' Armed-Forces',
       ' Priv-house-serv'], dtype=object)
```

In [27]:
```
salarydata_train.occupation.value_counts()
```

Out[27]:
```
 Prof-specialty       4038
 Craft-repair         4030
 Exec-managerial      3992
 Adm-clerical         3721
 Sales                3584
 Other-service        3212
 Machine-op-inspct    1965
 Transport-moving     1572
 Handlers-cleaners    1350
 Farming-fishing       989
 Tech-support          912
 Protective-serv       644
 Priv-house-serv       143
 Armed-Forces            9
Name: occupation, dtype: int64
```

In [28]:
```
salarydata_train.native.unique()
```

Out[28]:
```
array([' United-States', ' Cuba', ' Jamaica', ' India', ' Mexico',
       ' Puerto-Rico', ' Honduras', ' England', ' Canada', ' Germany',
       ' Iran', ' Philippines', ' Poland', ' Columbia', ' Cambodia',
       ' Thailand', ' Ecuador', ' Laos', ' Taiwan', ' Haiti', ' Portugal',
       ' Dominican-Republic', ' El-Salvador', ' France', ' Guatemala',
       ' Italy', ' China', ' South', ' Japan', ' Yugoslavia', ' Peru',
       ' Outlying-US(Guam-USVI-etc)', ' Scotland', ' Trinadad&Tobago',
       ' Greece', ' Nicaragua', ' Vietnam', ' Hong', ' Ireland',
       ' Hungary'], dtype=object)
```

In [29]:
```
salarydata_train.native.value_counts()
```

Out[29]:
```
United-States                 27504
Mexico                          610
Philippines                     188
Germany                         128
Puerto-Rico                     109
Canada                          107
India                           100
El-Salvador                     100
Cuba                             92
England                          86
Jamaica                          80
South                            71
China                            68
Italy                            68
Dominican-Republic               67
Vietnam                          64
Guatemala                        63
Japan                            59
Poland                           56
Columbia                         56
Iran                             42
Taiwan                           42
Haiti                            42
Portugal                         34
Nicaragua                        33
Peru                             30
Greece                           29
France                           27
Ecuador                          27
Ireland                          24
Hong                             19
Cambodia                         18
Trinadad&Tobago                  18
Laos                             17
Thailand                         17
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         12
Scotland                         11
Name: native, dtype: int64
```

In [30]:
```
for var in categorical:

    print(var, ' contains ', len(salarydata_train[var].unique()), ' labels')
```

```
workclass  contains  7  labels
education  contains  16  labels
maritalstatus  contains  7  labels
occupation  contains  14  labels
relationship  contains  6  labels
race  contains  5  labels
sex  contains  2  labels
native  contains  40  labels
Salary  contains  2  labels
```

In [31]:
```
numerical = [var for var in salarydata_train.columns if salarydata_train[var].dtype!='O']

print('There are {} numerical variables\n'.format(len(numerical)))

print('The numerical variables are :', numerical)
```

```
There are 5 numerical variables

The numerical variables are : ['age', 'educationno', 'capitalgain', 'capitalloss', 'hoursperweek']
```

In [32]:
```
salarydata_train[numerical].head()
```

Out[32]:

	age	educationno	capitalgain	capitalloss	hoursperweek
0	39	13	2174	0	40
1	50	13	0	0	13
2	38	9	0	0	40
3	53	7	0	0	40
4	28	13	0	0	40

In [33]:
```
salarydata_train[numerical].isnull().sum()
```

Out[33]:
```
age             0
educationno     0
capitalgain     0
capitalloss     0
hoursperweek    0
dtype: int64
```

In [34]:
```
X = salarydata_train.drop(['Salary'], axis=1)

y = salarydata_train['Salary']
```

In [35]:
```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.3, random_state = 0)
```

In [36]:
```
X_train.shape, X_test.shape
```

Out[36]:
```
((21112, 13), (9049, 13))
```

In [37]:
```
X_train.dtypes
```

Out[37]:
```
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
dtype: object
```

In [38]:
```
X_test.dtypes
```

Out[38]:
```
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
dtype: object
```

In [39]:
```
categorical = [col for col in X_train.columns if X_train[col].dtypes == 'O']

categorical
```

Out[39]:
```
['workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native']
```

In [40]:
```
numerical = [col for col in X_train.columns if X_train[col].dtypes != 'O']

numerical
```

Out[40]:
```
['age', 'educationno', 'capitalgain', 'capitalloss', 'hoursperweek']
```

In [41]:
```
X_train[categorical].isnull().mean()
```

Out[41]:
```
workclass        0.0
education        0.0
maritalstatus    0.0
occupation       0.0
relationship     0.0
race             0.0
sex              0.0
native           0.0
dtype: float64
```

In [42]:
```
for col in categorical:
    if X_train[col].isnull().mean()>0:
        print(col, (X_train[col].isnull().mean()))
```

In [44]:
```
X_train[categorical].isnull().sum()
```

Out[44]:
```
workclass        0
education        0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
native           0
dtype: int64
```

In [45]:
```
X_test[categorical].isnull().sum()
```

Out[45]:
```
workclass        0
education        0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
native           0
dtype: int64
```

In [46]:
```
X_train.isnull().sum()
```

Out[46]:
```
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
dtype: int64
```

In [47]:
```
X_test.isnull().sum()
```

Out[47]:
```
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
dtype: int64
```

In [48]:
```
categorical
```

Out[48]:
```
['workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native']
```

In [49]:
```
X_train[categorical].head()
```

Out[49]:

	workclass	education	maritalstatus	occupation	relationship	race	sex	native
8166	Local-gov	Some-college	Married-civ-spouse	Protective-serv	Husband	White	Male	United-States
7138	Private	Some-college	Never-married	Other-service	Own-child	White	Male	United-States
437	Private	HS-grad	Never-married	Transport-moving	Not-in-family	White	Male	United-States
5436	Private	HS-grad	Married-civ-spouse	Craft-repair	Husband	White	Male	United-States
6541	Self-emp-not-inc	HS-grad	Married-civ-spouse	Tech-support	Husband	White	Male	United-States

In [50]:
```
!pip install category_encoders
```

```
Requirement already satisfied: category_encoders in c:\users\admin\anaconda3\lib\site-packages (2.4.0)
Requirement already satisfied: statsmodels>=0.9.0 in c:\users\admin\anaconda3\lib\site-packages (from category
_encoders) (0.12.2)
Requirement already satisfied: numpy>=1.14.0 in c:\users\admin\anaconda3\lib\site-packages (from category_enco
ders) (1.20.3)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\users\admin\anaconda3\lib\site-packages (from catego
ry_encoders) (0.24.2)
Requirement already satisfied: pandas>=0.21.1 in c:\users\admin\anaconda3\lib\site-packages (from category_enc
oders) (1.3.4)
Requirement already satisfied: patsy>=0.5.1 in c:\users\admin\anaconda3\lib\site-packages (from category_encod
ers) (0.5.2)
Requirement already satisfied: scipy>=1.0.0 in c:\users\admin\anaconda3\lib\site-packages (from category_encod
ers) (1.7.1)
Requirement already satisfied: pytz>=2017.3 in c:\users\admin\anaconda3\lib\site-packages (from pandas>=0.21.1
->category_encoders) (2021.3)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\admin\anaconda3\lib\site-packages (from pand
as>=0.21.1->category_encoders) (2.8.2)
Requirement already satisfied: six in c:\users\admin\anaconda3\lib\site-packages (from patsy>=0.5.1->category_
encoders) (1.16.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\admin\anaconda3\lib\site-packages (from scikit
-learn>=0.20.0->category_encoders) (2.2.0)
Requirement already satisfied: joblib>=0.11 in c:\users\admin\anaconda3\lib\site-packages (from scikit-learn>=
0.20.0->category_encoders) (1.1.0)
```

In [51]:
```
import category_encoders as ce
```

In [52]:
```
encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'maritalstatus', 'occupation', 'relationship',
                                 'race', 'sex', 'native'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

In [53]:
```
X_train.head()
```

Out[53]:

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	education_1	education_2	...	r
8166	54	1	0	0	0	0	0	0	1	0	...	
7138	21	0	1	0	0	0	0	0	1	0	...	
437	30	0	1	0	0	0	0	0	0	1	...	
5436	42	0	1	0	0	0	0	0	0	1	...	
6541	37	0	0	1	0	0	0	0	0	1	...	

5 rows × 102 columns

In [55]:
```
X_train.shape
```

Out[55]: (21112, 102)

In [56]:
```
X_test.head()
```

Out[56]:

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	education_1	education_2	...
25338	21	0	1	0	0	0	0	0	0	1	...
18840	21	0	1	0	0	0	0	0	0	0	...
8391	56	0	1	0	0	0	0	0	0	0	...
18258	43	1	0	0	0	0	0	0	1	0	...
16669	53	0	0	0	1	0	0	0	0	0	...

5 rows × 102 columns

In [57]:
```
X_test.shape
```

Out[57]: (9049, 102)

In [58]:
```
cols = X_train.columns
```

In [59]:
```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

In [60]:
```
X_train = pd.DataFrame(X_train, columns=[cols])
```

In [61]:
```
X_test = pd.DataFrame(X_test, columns=[cols])
```

In [62]:
```
X_train.head()
```

Out[62]:

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workclass_7	education_1	education_2	...
0	0.894737	1.0	-1.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
1	-0.842105	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	0.0	...
2	-0.368421	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...
3	0.263158	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	1.0	...
4	0.000000	0.0	-1.0	1.0	0.0	0.0	0.0	0.0	0.0	1.0	...

5 rows × 102 columns

In [63]:
```
from sklearn.naive_bayes import GaussianNB

gnb = GaussianNB()

gnb.fit(X_train, y_train)
```

Out[63]: GaussianNB()

In [64]:
```
y_pred = gnb.predict(X_test)

y_pred
```

Out[64]:
```
array([' <=50K', ' <=50K', ' <=50K', ..., ' <=50K', ' <=50K', ' >50K'],
      dtype='<U6')
```

In [65]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score: {0:0.4f}'. format(accuracy_score(y_test, y_pred)))
```

```
Model accuracy score: 0.7995
```

In [66]:
```
y_pred_train = gnb.predict(X_train)

y_pred_train
```

Out[66]:
```
array([' >50K', ' <=50K', ' <=50K', ..., ' <=50K', ' >50K', ' <=50K'],
      dtype='<U6')
```

In [67]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train)))
```

```
Training-set accuracy score: 0.8023
```

In [68]:
```
print('Training set score: {:.4f}'.format(gnb.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(gnb.score(X_test, y_test)))
```

```
Training set score: 0.8023
Test set score: 0.7995
```

In [69]:
```
y_test.value_counts()
```

Out[69]:
```
 <=50K    6798
 >50K     2251
Name: Salary, dtype: int64
```

In [70]:
```
null_accuracy = (7407/(7407+2362))

print('Null accuracy score: {0:0.4f}'. format(null_accuracy))
```

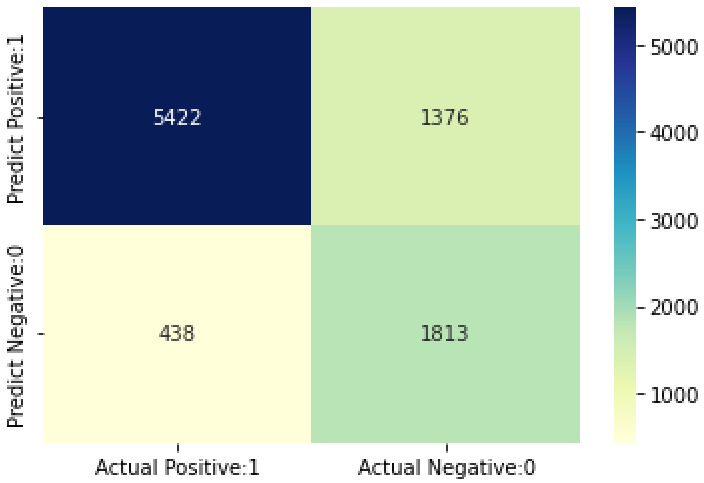
```
Null accuracy score: 0.7582
```

In [71]:
```
from sklearn.metrics import confusion_matrix

cm = confusion_matrix(y_test, y_pred)

print('Confusion matrix\n\n', cm)

print('\nTrue Positives(TP) = ', cm[0,0])

print('\nTrue Negatives(TN) = ', cm[1,1])

print('\nFalse Positives(FP) = ', cm[0,1])

print('\nFalse Negatives(FN) = ', cm[1,0])
```

```
Confusion matrix

 [[5422 1376]
 [ 438 1813]]

True Positives(TP) =  5422

True Negatives(TN) =  1813

False Positives(FP) =  1376

False Negatives(FN) =  438
```

In [73]:
```
import seaborn as sns
cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:0'],
                                 index=['Predict Positive:1', 'Predict Negative:0'])

sns.heatmap(cm_matrix, annot=True, fmt='d', cmap='YlGnBu')
```

Out[73]: <AxesSubplot:>



In [74]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

       <=50K       0.93      0.80      0.86      6798
        >50K       0.57      0.81      0.67      2251

    accuracy                           0.80      9049
   macro avg       0.75      0.80      0.76      9049
weighted avg       0.84      0.80      0.81      9049
```

In [75]:
```
TP = cm[0,0]
TN = cm[1,1]
FP = cm[0,1]
FN = cm[1,0]
```

In [76]:
```
classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

print('Classification accuracy : {0:0.4f}'.format(classification_accuracy))
```

```
Classification accuracy : 0.7995
```

In [77]:
```
classification_error = (FP + FN) / float(TP + TN + FP + FN)

print('Classification error : {0:0.4f}'.format(classification_error))
```

```
Classification error : 0.2005
```

In [78]:
```
precision = TP / float(TP + FP)


print('Precision : {0:0.4f}'.format(precision))
```

```
Precision : 0.7976
```

In [79]:
```
recall = TP / float(TP + FN)

print('Recall or Sensitivity : {0:0.4f}'.format(recall))
```

```
Recall or Sensitivity : 0.9253
```

In [80]:
```
true_positive_rate = TP / float(TP + FN)


print('True Positive Rate : {0:0.4f}'.format(true_positive_rate))
```

```
True Positive Rate : 0.9253
```

In [81]:
```
false_positive_rate = FP / float(FP + TN)


print('False Positive Rate : {0:0.4f}'.format(false_positive_rate))
```

```
False Positive Rate : 0.4315
```

In [82]:
```
specificity = TN / (TN + FP)

print('Specificity : {0:0.4f}'.format(specificity))
```

```
Specificity : 0.5685
```

In [83]:
```
y_pred_prob = gnb.predict_proba(X_test)[0:10]

y_pred_prob
```

Out[83]:
```
array([[9.99955511e-01, 4.44887598e-05],
       [9.95935549e-01, 4.06445120e-03],
       [8.63901480e-01, 1.36098520e-01],
       [9.99999906e-01, 9.37239455e-08],
       [8.80888343e-02, 9.11911166e-01],
       [9.99562896e-01, 4.37103927e-04],
       [5.34482750e-06, 9.99994655e-01],
       [6.28497161e-01, 3.71502839e-01],
       [5.46536963e-04, 9.99453463e-01],
       [9.99999570e-01, 4.30495598e-07]])
```

In [84]:
```
y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Prob of - <=50K', 'Prob of - >50K'])

y_pred_prob_df
```

Out[84]:

	Prob of - <=50K	Prob of - >50K
0	0.999956	4.448876e-05
1	0.995936	4.064451e-03
2	0.863901	1.360985e-01
3	1.000000	9.372395e-08
4	0.088089	9.119112e-01
5	0.999563	4.371039e-04
6	0.000005	9.999947e-01
7	0.628497	3.715028e-01
8	0.000547	9.994535e-01
9	1.000000	4.304956e-07

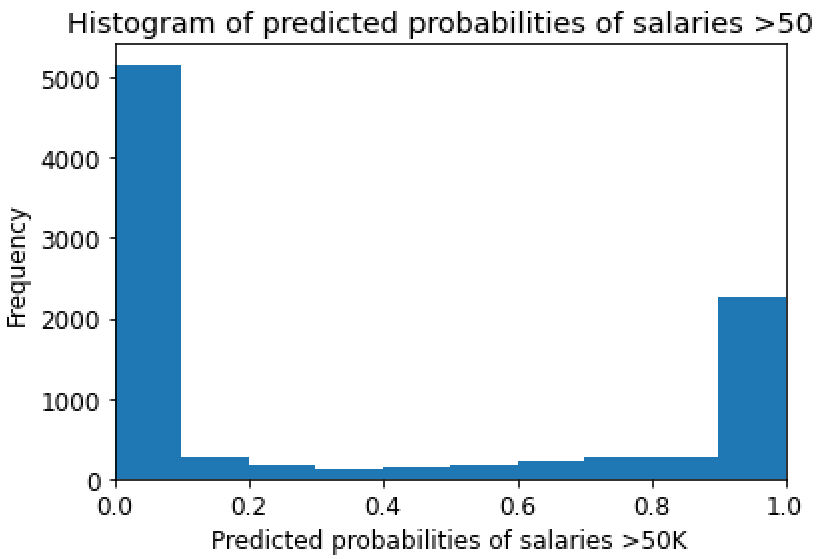
In [85]:
```
gnb.predict_proba(X_test)[0:10, 1]
```

Out[85]:
```
array([4.44887598e-05, 4.06445120e-03, 1.36098520e-01, 9.37239455e-08,
       9.11911166e-01, 4.37103927e-04, 9.99994655e-01, 3.71502839e-01,
       9.99453463e-01, 4.30495598e-07])
```

In [86]:
```
y_pred1 = gnb.predict_proba(X_test)[:, 1]
```

In [89]:
```
import matplotlib.pyplot as plt
plt.rcParams['font.size'] = 12


plt.hist(y_pred1, bins = 10)

plt.title('Histogram of predicted probabilities of salaries >50K')


plt.xlim(0,1)

plt.xlabel('Predicted probabilities of salaries >50K')
plt.ylabel('Frequency')
```

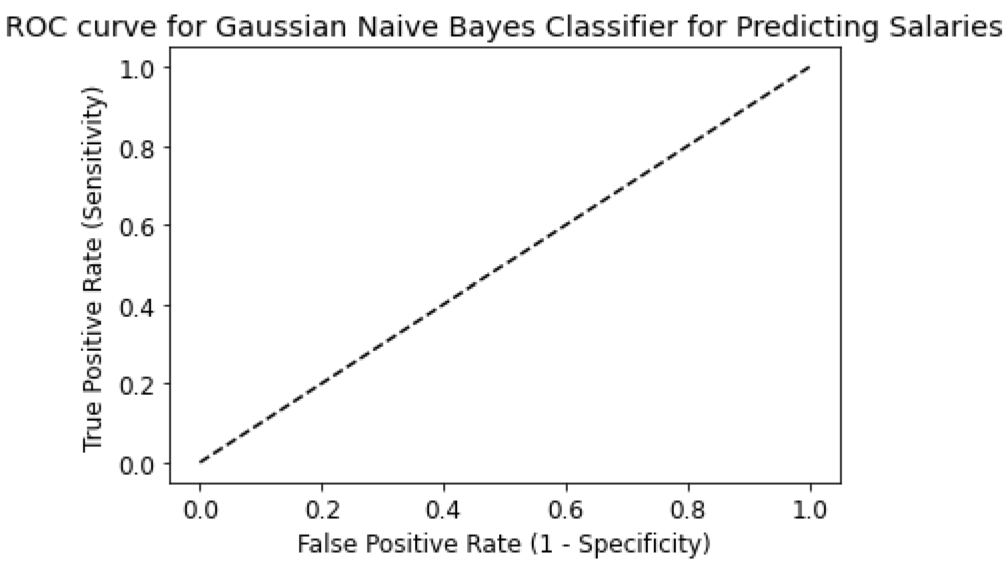
Out[89]:
```
Text(0, 0.5, 'Frequency')
```



In [90]:

```
from sklearn.metrics import roc_curve

fpr, tpr, thresholds = roc_curve(y_test, y_pred1, pos_label = '>50K')

plt.figure(figsize=(6,4))

plt.plot(fpr, tpr, linewidth=2)

plt.plot([0,1], [0,1], 'k--' )

plt.rcParams['font.size'] = 12

plt.title('ROC curve for Gaussian Naive Bayes Classifier for Predicting Salaries')

plt.xlabel('False Positive Rate (1 - Specificity)')

plt.ylabel('True Positive Rate (Sensitivity)')

plt.show()
```

```
C:\Users\Admin\anaconda3\lib\site-packages\sklearn\metrics\_ranking.py:949: UndefinedMetricWarning: No positiv
e samples in y_true, true positive value should be meaningless
  warnings.warn("No positive samples in y_true, "
```



In [91]:
```
from sklearn.metrics import roc_auc_score

ROC_AUC = roc_auc_score(y_test, y_pred1)

print('ROC AUC : {:.4f}'.format(ROC_AUC))
```

```
ROC AUC : 0.8902
```

In [92]:
```
from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(gnb, X_train, y_train, cv=5, scoring='roc_auc').mean()

print('Cross validated ROC AUC : {:.4f}'.format(Cross_validated_ROC_AUC))
```

```
Cross validated ROC AUC : 0.8923
```

In [93]:
```
from sklearn.model_selection import cross_val_score

scores = cross_val_score(gnb, X_train, y_train, cv = 10, scoring='accuracy')

print('Cross-validation scores:{}'.format(scores))
```

```
Cross-validation scores:[0.81676136 0.79829545 0.79014685 0.81288489 0.80388441 0.79062056
 0.80767409 0.7925154  0.79630507 0.80909522]
```

In [94]:
```
print('Average cross-validation score: {:.4f}'.format(scores.mean()))
```

```
Average cross-validation score: 0.8018
```

In [ ]: