In [2]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
import os
import warnings
warnings.filterwarnings('ignore')

from pandas.plotting import scatter_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import train_test_split
from sklearn.model_selection import KFold
from sklearn.model_selection import cross_val_score
from sklearn import metrics
import statsmodels.api as sm

from sklearn.datasets import fetch_20newsgroups
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import confusion_matrix, plot_confusion_matrix
```

In [9]:
```
salarydata_train = pd.read_csv('SalaryData_Train.csv')
salarydata_train
```

Out[9]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex
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

In [8]:
```
salarydata_test = pd.read_csv('SalaryData_Test.csv')
salarydata_test
```

Out[8]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	se
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

Exploratory data analysis

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

Out[5]: (30161, 14)

In [6]:
```
salarydata_test.shape
```

Out[6]: (15060, 14)

In [10]:
```
# preview the Training dataset

salarydata_train.head()
```

Out[10]:

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

In [11]:
```
# preview the Test dataset

salarydata_test.head()
```

Out[11]:

	age	workclass	education	educationno	maritalstatus	occupation	relationship	race	sex	ca
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	

View summary of Training dataset

In [12]:
```
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 [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 [14]:
```
salarydata_train.describe()
```

Out[14]:

	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 [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]:
```
#Finding the special characters in the data frame
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]:
```
#Finding the special characters in the data frame
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
```

Explore categorical variables

In [19]:
```
# find categorical variables

categorical = [var for var in salarydata_train.columns if salarydata_train[var].d

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', 'rac
e', 'sex', 'native', 'Salary']
```

In [20]:
```
# view the categorical variables

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

Explore problems within categorical variables

In [21]:
```
# check missing values in categorical variables
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]:
```
# view frequency counts of values in categorical variables

for var in categorical:

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

```
 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
 HS-grad         9840
 Some-college    6677
 Bachelors       5044
 Masters         1627
 Assoc-voc       1307
 11th            1048
 Assoc-acdm      1008
 10th             820
 7th-8th          557
 Prof-school      542
 9th              455
```

In [23]:
```
# view frequency distribution of categorical variables

for var in categorical:

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

```
 Private             0.738868
 Self-emp-not-inc    0.082855
 Local-gov           0.068532
 State-gov           0.042406
 Self-emp-inc        0.035609
 Federal-gov         0.031266
 Without-pay         0.000464
Name: workclass, dtype: float64
 HS-grad         0.326249
 Some-college    0.221379
 Bachelors       0.167236
 Masters         0.053944
 Assoc-voc       0.043334
 11th            0.034747
 Assoc-acdm      0.033421
 10th            0.027187
 7th-8th         0.018468
 Prof-school     0.017970
 9th             0.015086
```

In [24]:
```
# check labels in workclass variable

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]:
```
# check frequency distribution of values in workclass variable

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
```

Explore occupation variable

In [26]:
```
# check labels in occupation variable

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 [28]:
```
# check frequency distribution of values in occupation variable

salarydata_train.occupation.value_counts()
```

Out[28]:
```
 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
```

Explore native_country variable

In [29]:
```
# check labels in native_country variable

salarydata_train.native.unique()
```

Out[29]:
```
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 [30]:
```
# check frequency distribution of values in native_country variable

salarydata_train.native.value_counts()
```

Out[30]:
```
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
Italy                            68
China                            68
Dominican-Republic               67
Vietnam                          64
Guatemala                        63
Japan                            59
Columbia                         56
Poland                           56
Haiti                            42
Iran                             42
Taiwan                           42
Portugal                         34
Nicaragua                        33
Peru                             30
Greece                           29
Ecuador                          27
France                           27
Ireland                          24
Hong                             19
Trinadad&Tobago                  18
Cambodia                         18
Laos                             17
Thailand                         17
Yugoslavia                       16
Outlying-US(Guam-USVI-etc)       14
Hungary                          13
Honduras                         12
Scotland                         11
Name: native, dtype: int64
```

Number of labels: cardinality

In [31]:
```
# check for cardinality in categorical variables

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
```

Explore Numerical Variables

In [32]:
```
# find numerical variables

numerical = [var for var in salarydata_train.columns if salarydata_train[var].dty

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', 'capitallos
s', 'hoursperweek']
```

In [33]:
```
# view the numerical variables

salarydata_train[numerical].head()
```

Out[33]:

	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

Explore problems within numerical variables

In [34]:
```
# check missing values in numerical variables

salarydata_train[numerical].isnull().sum()
```

Out[34]:
```
age             0
educationno     0
capitalgain     0
capitalloss     0
hoursperweek    0
dtype: int64
```

Declare feature vector and target variable

In [35]:
```
X = salarydata_train.drop(['Salary'], axis=1)

y = salarydata_train['Salary']
```

Split data into separate training and test set

In [36]:
```
# split X and y into training and testing sets

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
```

In [37]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

Out[37]:
```
((21112, 13), (9049, 13))
```

Feature Engineering

In [38]:
```
X_train.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 [40]:
```
X_test.dtypes
```

Out[40]:
```
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 [41]:
```
# display categorical variables

categorical = [col for col in X_train.columns if X_train[col].dtypes == 'O']

categorical
```

Out[41]:
```
['workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native']
```

In [42]:
```
# display numerical variables

numerical = [col for col in X_train.columns if X_train[col].dtypes != 'O']

numerical
```

Out[42]:
```
['age', 'educationno', 'capitalgain', 'capitalloss', 'hoursperweek']
```

In [43]:
```
# print percentage of missing values in the categorical variables in training set

X_train[categorical].isnull().mean()
```

Out[43]:
```
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 [44]:
```
# print categorical variables with missing data

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

In [45]:
```
# impute missing categorical variables with most frequent value

for df2 in [X_train, X_test]:
    df2['workclass'].fillna(X_train['workclass'].mode()[0], inplace=True)
    df2['occupation'].fillna(X_train['occupation'].mode()[0], inplace=True)
    df2['native'].fillna(X_train['native'].mode()[0], inplace=True)
```

In [46]:
```
# check missing values in categorical variables in X_train

X_train[categorical].isnull().sum()
```

Out[46]:
```
workclass        0
education        0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
native           0
dtype: int64
```

In [47]:
```
# check missing values in categorical variables in X_test

X_test[categorical].isnull().sum()
```

Out[47]:
```
workclass        0
education        0
maritalstatus    0
occupation       0
relationship     0
race             0
sex              0
native           0
dtype: int64
```

In [48]:
```
# check missing values in X_train

X_train.isnull().sum()
```

Out[48]:
```
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 [49]:
```
# check missing values in X_test

X_test.isnull().sum()
```

Out[49]:
```
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
```

Encode categorical variables

In [50]:
```
# print categorical variables

categorical
```

Out[50]:
```
['workclass',
 'education',
 'maritalstatus',
 'occupation',
 'relationship',
 'race',
 'sex',
 'native']
```

In [51]:
```
X_train[categorical].head()
```

Out[51]:

	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 [52]:
```
!pip install category_encoders
```

```
Collecting category_encoders
  Downloading category_encoders-2.3.0-py2.py3-none-any.whl (82 kB)
Requirement already satisfied: pandas>=0.21.1 in c:\programdata\anaconda3\lib\s
ite-packages (from category_encoders) (1.2.4)
Requirement already satisfied: numpy>=1.14.0 in c:\programdata\anaconda3\lib\si
te-packages (from category_encoders) (1.20.1)
Requirement already satisfied: scikit-learn>=0.20.0 in c:\programdata\anaconda3
\lib\site-packages (from category_encoders) (0.24.1)
Requirement already satisfied: scipy>=1.0.0 in c:\programdata\anaconda3\lib\sit
e-packages (from category_encoders) (1.6.2)
Requirement already satisfied: patsy>=0.5.1 in c:\programdata\anaconda3\lib\sit
e-packages (from category_encoders) (0.5.1)
Requirement already satisfied: statsmodels>=0.9.0 in c:\programdata\anaconda3\l
ib\site-packages (from category_encoders) (0.12.2)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\programdata\anacond
a3\lib\site-packages (from pandas>=0.21.1->category_encoders) (2.8.1)
Requirement already satisfied: pytz>=2017.3 in c:\programdata\anaconda3\lib\sit
e-packages (from pandas>=0.21.1->category_encoders) (2021.1)
Requirement already satisfied: six in c:\programdata\anaconda3\lib\site-package
s (from patsy>=0.5.1->category_encoders) (1.15.0)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\anaconda3
\lib\site-packages (from scikit-learn>=0.20.0->category_encoders) (2.1.0)
Requirement already satisfied: joblib>=0.11 in c:\programdata\anaconda3\lib\sit
e-packages (from scikit-learn>=0.20.0->category_encoders) (1.0.1)
Installing collected packages: category-encoders
Successfully installed category-encoders-2.3.0
```

In [53]:
```
# import category encoders

import category_encoders as ce
```

In [54]:
```
# encode remaining variables with one-hot encoding

encoder = ce.OneHotEncoder(cols=['workclass', 'education', 'maritalstatus', 'occu
                                 'race', 'sex', 'native'])

X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

In [55]:
```
X_train.head()
```

Out[55]:

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workcl
8166	54	1	0	0	0	0	0	
7138	21	0	1	0	0	0	0	
437	30	0	1	0	0	0	0	
5436	42	0	1	0	0	0	0	
6541	37	0	0	1	0	0	0	

5 rows × 102 columns

In [56]:
```
X_train.shape
```

Out[56]:
```
(21112, 102)
```

In [57]:
```
X_test.head()
```

Out[57]:

	age	workclass_1	workclass_2	workclass_3	workclass_4	workclass_5	workclass_6	workc
25338	21	0	1	0	0	0	0	
18840	21	0	1	0	0	0	0	
8391	56	0	1	0	0	0	0	
18258	43	1	0	0	0	0	0	
16669	53	0	0	0	1	0	0	

5 rows × 102 columns

In [58]:
```
X_test.shape
```

Out[58]:
```
(9049, 102)
```

Feature Scaling

In [59]:
```
cols = X_train.columns
```

In [60]:
```
from sklearn.preprocessing import RobustScaler

scaler = RobustScaler()

X_train = scaler.fit_transform(X_train)

X_test = scaler.transform(X_test)
```

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

In [62]:
```
X_test = pd.DataFrame(X_test, columns=[cols])
```

In [63]:
```
X_train.head()
```

Out[63]:

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

5 rows × 102 columns

Model training

In [64]:
```
# train a Gaussian Naive Bayes classifier on the training set
from sklearn.naive_bayes import GaussianNB


# instantiate the model
gnb = GaussianNB()


# fit the model
gnb.fit(X_train, y_train)
```

Out[64]:
```
GaussianNB()
```

Predict the results

In [65]:
```
y_pred = gnb.predict(X_test)

y_pred
```

Out[65]:
```
array([' <=50K', ' <=50K', ' <=50K', ..., ' <=50K', ' <=50K', ' >50K'],
      dtype='<U6')
```

Check accuracy score

In [66]:
```
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
```

Compare the train-set and test-set accuracy

In [67]:
```
y_pred_train = gnb.predict(X_train)

y_pred_train
```

Out[67]:
```
array([' >50K', ' <=50K', ' <=50K', ..., ' <=50K', ' >50K', ' <=50K'],
      dtype='<U6')
```

In [68]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_p
```

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

Check for overfitting and underfitting

In [69]:
```
# print the scores on training and test set

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
```

Compare model accuracy with null accuracy

In [70]:
```
# check class distribution in test set

y_test.value_counts()
```

Out[70]:
```
 <=50K    6798
 >50K     2251
Name: Salary, dtype: int64
```

In [71]:
```
# check null accuracy score

null_accuracy = (7407/(7407+2362))

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

```
Null accuracy score: 0.7582
```

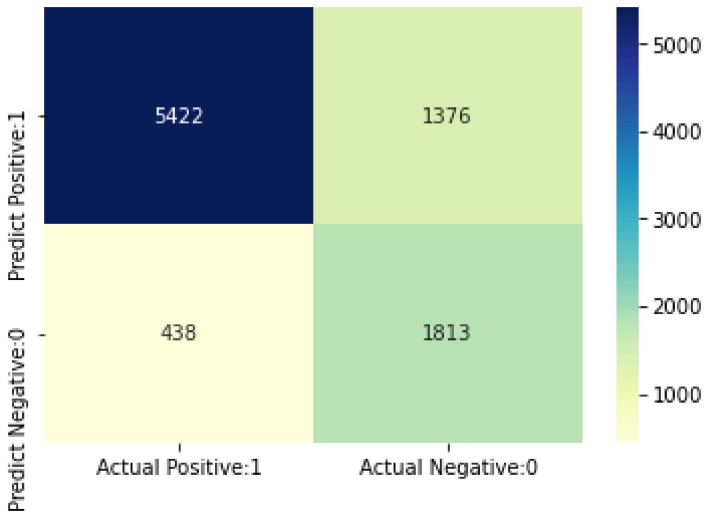
Confusion matrix

In [72]:
```
# Print the Confusion Matrix and slice it into four pieces

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]:
```
# visualize confusion matrix with seaborn heatmap

cm_matrix = pd.DataFrame(data=cm, columns=['Actual Positive:1', 'Actual Negative:
                                           index=['Predict Positive:1', 'Predict Negative:0

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

Out[73]: <AxesSubplot:>



Classification metrices

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
```

Classification accuracy

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

In [76]:
```
# print classification accuracy

classification_accuracy = (TP + TN) / float(TP + TN + FP + FN)

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

```
Classification accuracy : 0.7995
```

Classification error

In [77]:
```
# print classification error

classification_error = (FP + FN) / float(TP + TN + FP + FN)

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

```
Classification error : 0.2005
```

Precision

In [78]:
```
# print precision score

precision = TP / float(TP + FP)


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

```
Precision : 0.7976
```

Recall

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
```

False Positive Rate

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
```

Calculate class probabilities

In [82]:
```
# print the first 10 predicted probabilities of two classes- 0 and 1

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

y_pred_prob
```

Out[82]:
```
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 [83]:
```
# store the probabilities in dataframe

y_pred_prob_df = pd.DataFrame(data=y_pred_prob, columns=['Prob of - <=50K', 'Prob

y_pred_prob_df
```

Out[83]:

	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

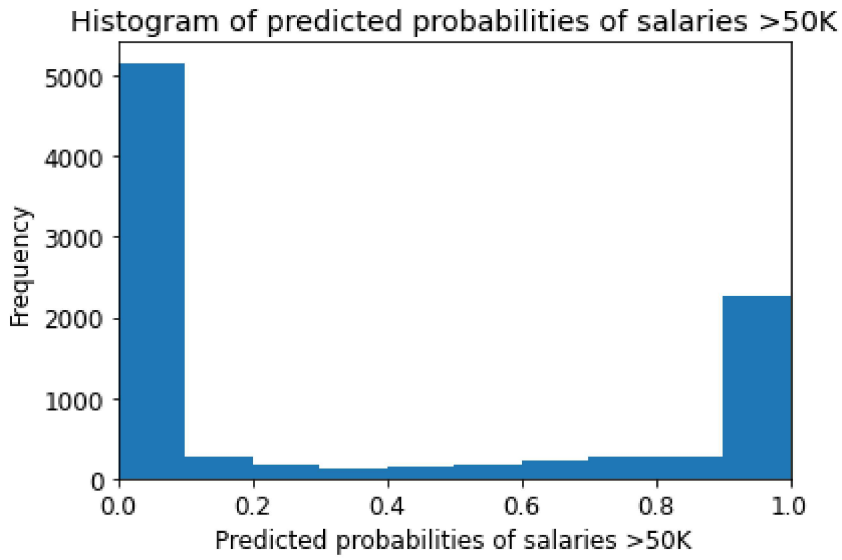
In [84]:
```
# print the first 10 predicted probabilities for class 1 - Probability of >50K

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

Out[84]:
```
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 [85]:
```
# store the predicted probabilities for class 1 - Probability of >50K

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

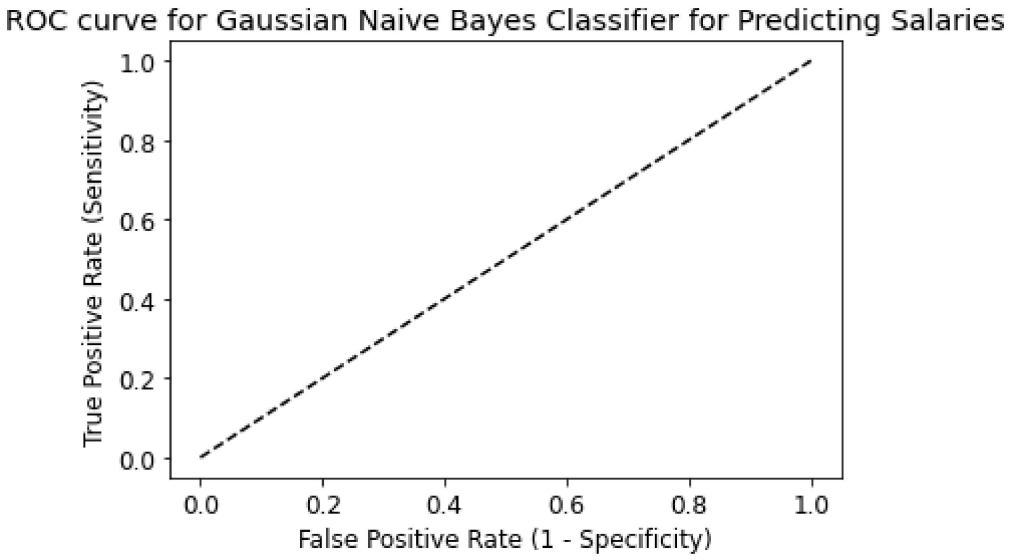
In [87]:
```
# plot histogram of predicted probabilities


# adjust the font size
plt.rcParams['font.size'] = 12


# plot histogram with 10 bins
plt.hist(y_pred1, bins = 10)


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

# set the x-axis limit
plt.xlim(0,1)


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

Out[87]:
```
Text(0, 0.5, 'Frequency')
```



ROC - AUC

In [88]:
```
# plot ROC Curve

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()
```



In [89]:
```
# compute ROC AUC

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
```

Interpretation

In [90]:
```
# calculate cross-validated ROC AUC

from sklearn.model_selection import cross_val_score

Cross_validated_ROC_AUC = cross_val_score(gnb, X_train, y_train, cv=5, scoring='r

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

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

k-Fold Cross Validation

In [91]:
```
# Applying 10-Fold Cross Validation

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 [92]:
```python
# compute Average cross-validation score

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

```
Average cross-validation score: 0.8018

Exception in callback BaseSelectorEventLoop._read_from_self()
handle: <Handle BaseSelectorEventLoop._read_from_self()>
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\asyncio\events.py", line 81, in _run
    self._context.run(self._callback, *self._args)
  File "C:\ProgramData\Anaconda3\lib\asyncio\selector_events.py", line 120, in
_read_from_self
    data = self._ssock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
Exception in callback BaseSelectorEventLoop._read_from_self()
handle: <Handle BaseSelectorEventLoop._read_from_self()>
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\asyncio\events.py", line 81, in _run
    self._context.run(self._callback, *self._args)
  File "C:\ProgramData\Anaconda3\lib\asyncio\selector_events.py", line 120, in
_read_from_self
    data = self._ssock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
Exception in callback BaseSelectorEventLoop._read_from_self()
handle: <Handle BaseSelectorEventLoop._read_from_self()>
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\asyncio\events.py", line 81, in _run
    self._context.run(self._callback, *self._args)
  File "C:\ProgramData\Anaconda3\lib\asyncio\selector_events.py", line 120, in
_read_from_self
    data = self._ssock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
Exception in callback BaseSelectorEventLoop._read_from_self()
handle: <Handle BaseSelectorEventLoop._read_from_self()>
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\asyncio\events.py", line 81, in _run
    self._context.run(self._callback, *self._args)
  File "C:\ProgramData\Anaconda3\lib\asyncio\selector_events.py", line 120, in
_read_from_self
    data = self._ssock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
Exception in callback BaseSelectorEventLoop._read_from_self()
handle: <Handle BaseSelectorEventLoop._read_from_self()>
Traceback (most recent call last):
  File "C:\ProgramData\Anaconda3\lib\asyncio\events.py", line 81, in _run
    self._context.run(self._callback, *self._args)
  File "C:\ProgramData\Anaconda3\lib\asyncio\selector_events.py", line 120, in
_read_from_self
    data = self._ssock.recv(4096)
ConnectionResetError: [WinError 10054] An existing connection was forcibly clos
ed by the remote host
```

In [ ]: