# Assignment - Naive Bayes

1. Prepare a classification model using Naive Bayes for salary data

Data Description:

age -- age of a person

workclass-- A work class is a grouping of work

education-- Education of an individuals

maritalstatus -- Marital status of an individulas

occupation-- occupation of an individuals

relationship --

race -- Race of an Individual

sex -- Gender of an Individual

capitalgain -- profit received from the sale of an investment

capitalloss -- A decrease in the value of a capital asset

hoursperweek -- number of hours work per week

native -- Native of an individual

Salary -- salary of an individual

**Import libraries**

In [1]:

**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

**Import dataset**

In [2]:

salarydata\_train **=** pd**.**read\_csv('SalaryData\_Train.csv')

salarydata\_train**.**head()

Out[2]:

|  | **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 [3]:

salarydata\_test **=** pd**.**read\_csv('SalaryData\_Test.csv')

salarydata\_test**.**head()

Out[3]:

| **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 |

# Exploratory data analysis

In [4]:

salarydata\_train**.**shape

Out[4]:

(30161, 14)

We can see that there are 30161 instances and 14 attributes in the training data set.

In [5]:

salarydata\_test**.**shape

Out[5]:

(15060, 14)

We can see that there are 15060 instances and 14 attributes in the test data set.

# View top 5 rows of dataset

In [6]:

*# preview the Training dataset*

salarydata\_train**.**head()

Out[6]:

|  | **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 [7]:

*# preview the Test dataset*

salarydata\_test**.**head()

Out[7]:

|  | **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 |

# View summary of Training dataset

In [8]:

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 [9]:

salarydata\_train**.**describe()

Out[9]:

|  | **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 [10]:

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 [11]:

salarydata\_test**.**describe()

Out[11]:

|  | **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 [12]:

*#Finding the special characters in the data frame*

salarydata\_train**.**isin(['?'])**.**sum(axis**=**0)

Out[12]:

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 [13]:

*#Finding the special characters in the data frame*

salarydata\_test**.**isin(['?'])**.**sum(axis**=**0)

Out[13]:

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 [14]:

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 [15]:

*# find categorical variables*

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 [16]:

*# view the categorical variables*

salarydata\_train[categorical]**.**head()

Out[16]:

|  | **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 |

Summary of categorical variables

There are 9 categorical variables.

The categorical variables are given by workclass, education, maritalstatus, occupation, relationship, race, sex, native and Salary.

Salary is the target variable.

# Explore problems within categorical variables

In [17]:

*# check missing values in categorical variables*

salarydata\_train[categorical]**.**isnull()**.**sum()

Out[17]:

workclass 0

education 0

maritalstatus 0

occupation 0

relationship 0

race 0

sex 0

native 0

Salary 0

dtype: int64

We can see that there are no missing values in the categorical variables. I will confirm this further.

In [18]:

*# 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

12th 377

Doctorate 375

5th-6th 288

1st-4th 151

Preschool 45

Name: education, dtype: int64

Married-civ-spouse 14065

Never-married 9725

Divorced 4214

Separated 939

Widowed 827

Married-spouse-absent 370

Married-AF-spouse 21

Name: maritalstatus, dtype: int64

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

Husband 12463

Not-in-family 7726

Own-child 4466

Unmarried 3212

Wife 1406

Other-relative 888

Name: relationship, dtype: int64

White 25932

Black 2817

Asian-Pac-Islander 895

Amer-Indian-Eskimo 286

Other 231

Name: race, dtype: int64

Male 20380

Female 9781

Name: sex, dtype: int64

United-States 27504

Mexico 610

Philippines 188

Germany 128

Puerto-Rico 109

Canada 107

El-Salvador 100

India 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

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 [19]:

*# 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

12th 0.012500

Doctorate 0.012433

5th-6th 0.009549

1st-4th 0.005006

Preschool 0.001492

Name: education, dtype: float64

Married-civ-spouse 0.466331

Never-married 0.322436

Divorced 0.139717

Separated 0.031133

Widowed 0.027420

Married-spouse-absent 0.012267

Married-AF-spouse 0.000696

Name: maritalstatus, dtype: float64

Prof-specialty 0.133882

Craft-repair 0.133616

Exec-managerial 0.132356

Adm-clerical 0.123371

Sales 0.118829

Other-service 0.106495

Machine-op-inspct 0.065150

Transport-moving 0.052120

Handlers-cleaners 0.044760

Farming-fishing 0.032791

Tech-support 0.030238

Protective-serv 0.021352

Priv-house-serv 0.004741

Armed-Forces 0.000298

Name: occupation, dtype: float64

Husband 0.413216

Not-in-family 0.256159

Own-child 0.148072

Unmarried 0.106495

Wife 0.046616

Other-relative 0.029442

Name: relationship, dtype: float64

White 0.859786

Black 0.093399

Asian-Pac-Islander 0.029674

Amer-Indian-Eskimo 0.009482

Other 0.007659

Name: race, dtype: float64

Male 0.675707

Female 0.324293

Name: sex, dtype: float64

United-States 0.911906

Mexico 0.020225

Philippines 0.006233

Germany 0.004244

Puerto-Rico 0.003614

Canada 0.003548

El-Salvador 0.003316

India 0.003316

Cuba 0.003050

England 0.002851

Jamaica 0.002652

South 0.002354

Italy 0.002255

China 0.002255

Dominican-Republic 0.002221

Vietnam 0.002122

Guatemala 0.002089

Japan 0.001956

Columbia 0.001857

Poland 0.001857

Haiti 0.001393

Iran 0.001393

Taiwan 0.001393

Portugal 0.001127

Nicaragua 0.001094

Peru 0.000995

Greece 0.000962

Ecuador 0.000895

France 0.000895

Ireland 0.000796

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

In [20]:

*# check labels in workclass variable*

salarydata\_train**.**workclass**.**unique()

Out[20]:

array([' State-gov', ' Self-emp-not-inc', ' Private', ' Federal-gov',

' Local-gov', ' Self-emp-inc', ' Without-pay'], dtype=object)

In [21]:

*# check frequency distribution of values in workclass variable*

salarydata\_train**.**workclass**.**value\_counts()

Out[21]:

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 [22]:

*# check labels in occupation variable*

salarydata\_train**.**occupation**.**unique()

Out[22]:

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 [23]:

*# check frequency distribution of values in occupation variable*

salarydata\_train**.**occupation**.**value\_counts()

Out[23]:

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 [24]:

*# check labels in native\_country variable*

salarydata\_train**.**native**.**unique()

Out[24]:

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 [25]:

*# check frequency distribution of values in native\_country variable*

salarydata\_train**.**native**.**value\_counts()

Out[25]:

United-States 27504

Mexico 610

Philippines 188

Germany 128

Puerto-Rico 109

Canada 107

El-Salvador 100

India 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

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

# Number of labels: cardinality

In [26]:

*# 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 [27]:

*# find numerical variables*

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 [28]:

*# view the numerical variables*

salarydata\_train[numerical]**.**head()

Out[28]:

|  | **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 |

Summary of numerical variables

There are 5 numerical variables.

These are given by age, educationno, capitalgain, capitalloss and hoursperweek. All of the numerical variables are of discrete data type.

# Explore problems within numerical variables

In [29]:

*# check missing values in numerical variables*

salarydata\_train[numerical]**.**isnull()**.**sum()

Out[29]:

age 0

educationno 0

capitalgain 0

capitalloss 0

hoursperweek 0

dtype: int64

# Declare feature vector and target variable

In [30]:

X **=** salarydata\_train**.**drop(['Salary'], axis**=**1)

y **=** salarydata\_train['Salary']

# Split data into separate training and test set

In [31]:

*# 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\_state **=** 0)

In [32]:

*# check the shape of X\_train and X\_test*

X\_train**.**shape, X\_test**.**shape

Out[32]:

((21112, 13), (9049, 13))

# Feature Engineering

In [33]:

X\_train**.**dtypes

Out[33]:

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 [34]:

X\_test**.**dtypes

Out[34]:

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 [35]:

*# display categorical variables*

categorical **=** [col **for** col **in** X\_train**.**columns **if** X\_train[col]**.**dtypes **==** 'O']

categorical

Out[35]:

['workclass',

'education',

'maritalstatus',

'occupation',

'relationship',

'race',

'sex',

'native']

In [36]:

*# display numerical variables*

numerical **=** [col **for** col **in** X\_train**.**columns **if** X\_train[col]**.**dtypes **!=** 'O']

numerical

Out[36]:

['age', 'educationno', 'capitalgain', 'capitalloss', 'hoursperweek']

In [37]:

*# print percentage of missing values in the categorical variables in training set*

X\_train[categorical]**.**isnull()**.**mean()

Out[37]:

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 [38]:

*# 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 [39]:

*# 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 [40]:

*# check missing values in categorical variables in X\_train*

X\_train[categorical]**.**isnull()**.**sum()

Out[40]:

workclass 0

education 0

maritalstatus 0

occupation 0

relationship 0

race 0

sex 0

native 0

dtype: int64

In [41]:

*# check missing values in categorical variables in X\_test*

X\_test[categorical]**.**isnull()**.**sum()

Out[41]:

workclass 0

education 0

maritalstatus 0

occupation 0

relationship 0

race 0

sex 0

native 0

dtype: int64

In [42]:

*# check missing values in X\_train*

X\_train**.**isnull()**.**sum()

Out[42]:

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 [43]:

*# check missing values in X\_test*

X\_test**.**isnull()**.**sum()

Out[43]:

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 [44]:

*# print categorical variables*

categorical

Out[44]:

['workclass',

'education',

'maritalstatus',

'occupation',

'relationship',

'race',

'sex',

'native']

In [45]:

X\_train[categorical]**.**head()

Out[45]:

|  | **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 [46]:

*# import category encoders*

**import** category\_encoders **as** ce

In [48]:

*# encode remaining variables with one-hot encoding*

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 [49]:

X\_train**.**head()

Out[49]:

|  | **age** | **workclass\_1** | **workclass\_2** | **workclass\_3** | **workclass\_4** | **workclass\_5** | **workclass\_6** | **workclass\_7** | **education\_1** | **education\_2** | **...** | **native\_31** | **native\_32** | **native\_33** | **native\_34** | **native\_35** | **native\_36** | **native\_37** | **native\_38** | **native\_39** | **native\_40** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **8166** | 54 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **7138** | 21 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **437** | 30 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **5436** | 42 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **6541** | 37 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 102 columns

In [50]:

X\_train**.**shape

Out[50]:

(21112, 102)

We can see that from the initial 14 columns, we now have 102 columns.

In [51]:

X\_test**.**head()

Out[51]:

|  | **age** | **workclass\_1** | **workclass\_2** | **workclass\_3** | **workclass\_4** | **workclass\_5** | **workclass\_6** | **workclass\_7** | **education\_1** | **education\_2** | **...** | **native\_31** | **native\_32** | **native\_33** | **native\_34** | **native\_35** | **native\_36** | **native\_37** | **native\_38** | **native\_39** | **native\_40** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **25338** | 21 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **18840** | 21 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **8391** | 56 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **18258** | 43 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 1 | 0 | ... | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| **16669** | 53 | 0 | 0 | 0 | 1 | 0 | 0 | 0 | 0 | 0 | ... | 0 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |

5 rows × 102 columns

In [52]:

X\_test**.**shape

Out[52]:

(9049, 102)

We now have training and testing set ready for model building. Before that, we should map all the feature variables onto the same scale. It is called feature scaling.

# Feature Scaling

In [53]:

cols **=** X\_train**.**columns

In [54]:

**from** sklearn.preprocessing **import** RobustScaler

scaler **=** RobustScaler()

X\_train **=** scaler**.**fit\_transform(X\_train)

X\_test **=** scaler**.**transform(X\_test)

In [55]:

X\_train **=** pd**.**DataFrame(X\_train, columns**=**[cols])

In [56]:

X\_test **=** pd**.**DataFrame(X\_test, columns**=**[cols])

In [57]:

X\_train**.**head()

Out[57]:

|  | **age** | **workclass\_1** | **workclass\_2** | **workclass\_3** | **workclass\_4** | **workclass\_5** | **workclass\_6** | **workclass\_7** | **education\_1** | **education\_2** | **...** | **native\_31** | **native\_32** | **native\_33** | **native\_34** | **native\_35** | **native\_36** | **native\_37** | **native\_38** | **native\_39** | **native\_40** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | 0.894737 | 1.0 | -1.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| **1** | -0.842105 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 1.0 | 0.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 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 | 0.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 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 | 0.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 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 | 0.0 | 0.0 | 1.0 | ... | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

5 rows × 102 columns

We now have X\_train dataset ready to be fed into the Gaussian Naive Bayes classifier.

# Model training

In [58]:

*# 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[58]:

GaussianNB()

# Predict the results

In [59]:

y\_pred **=** gnb**.**predict(X\_test)

y\_pred

Out[59]:

array([' <=50K', ' <=50K', ' <=50K', ..., ' <=50K', ' <=50K', ' >50K'],

dtype='<U6')

# Check accuracy score

In [60]:

**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

Here, y\_test are the true class labels and y\_pred are the predicted class labels in the test-set.

# Compare the train-set and test-set accuracy

In [61]:

y\_pred\_train **=** gnb**.**predict(X\_train)

y\_pred\_train

Out[61]:

array([' >50K', ' <=50K', ' <=50K', ..., ' <=50K', ' >50K', ' <=50K'],

dtype='<U6')

In [62]:

print('Training-set accuracy score: {0:0.4f}'**.** format(accuracy\_score(y\_train, y\_pred\_train)))

Training-set accuracy score: 0.8023

# Check for overfitting and underfitting

In [63]:

*# 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

The training-set accuracy score is 0.8023 while the test-set accuracy to be 0.7995. These two values are quite comparable. So, there is no sign of overfitting

# Compare model accuracy with null accuracy

In [64]:

*# check class distribution in test set*

y\_test**.**value\_counts()

Out[64]:

<=50K 6798

>50K 2251

Name: Salary, dtype: int64

In [65]:

*# check null accuracy score*

null\_accuracy **=** (7407**/**(7407**+**2362))

print('Null accuracy score: {0:0.4f}'**.** format(null\_accuracy))

Null accuracy score: 0.7582

We can see that our model accuracy score is 0.8023 but null accuracy score is 0.7582. So, we can conclude that our Gaussian Naive Bayes Classification model is doing a very good job in predicting the class labels.

# Confusion matrix

In [66]:

*# 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 [67]:

*# visualize confusion matrix with seaborn heatmap*

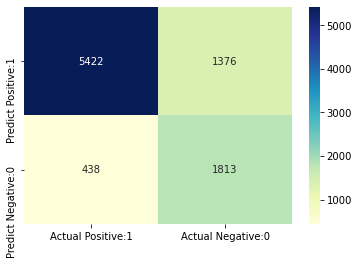
cm\_matrix **=** pd**.**DataFrame(data**=**cm, columns**=**['Actual Positive:1', 'Actual Negative:0'],

index**=**['Predict Positive:1', 'Predict Negative:0'])

sns**.**heatmap(cm\_matrix, annot**=True**, fmt**=**'d', cmap**=**'YlGnBu')

Out[67]:

<matplotlib.axes.\_subplots.AxesSubplot at 0x22ef3f1d3a0>



# Classification metrices

In [68]:

**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 [69]:

TP **=** cm[0,0]

TN **=** cm[1,1]

FP **=** cm[0,1]

FN **=** cm[1,0]

In [70]:

*# 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 [71]:

*# 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 [72]:

*# print precision score*

precision **=** TP **/** float(TP **+** FP)

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

Precision : 0.7976

**Recall**

In [73]:

recall **=** TP **/** float(TP **+** FN)

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

Recall or Sensitivity : 0.9253

True Positive Rate

True Positive Rate is synonymous with Recall.

In [74]:

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 [75]:

false\_positive\_rate **=** FP **/** float(FP **+** TN)

print('False Positive Rate : {0:0.4f}'**.**format(false\_positive\_rate))

False Positive Rate : 0.4315

**Specificity**

In [76]:

specificity **=** TN **/** (TN **+** FP)

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

Specificity : 0.5685

**Calculate class probabilities**

In [77]:

*# 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[77]:

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]])

Observations

* In each row, the numbers sum to 1.
* There are 2 columns which correspond to 2 classes - <=50K and >50K.
* \* Class 0 => <=50K - Class that a person makes less than equal to 50K.
* \* Class 1 => >50K - Class that a person makes more than 50K.
* Importance of predicted probabilities
  + We can rank the observations by probability of whether a person makes less than or equal to 50K or more than 50K.
* predict\_proba process
  + Predicts the probabilities
  + Choose the class with the highest probability
* Classification threshold level
  + There is a classification threshold level of 0.5.
  + Class 0 => <=50K - probability of salary less than or equal to 50K is predicted if probability < 0.5.
  + Class 1 => >50K - probability of salary more than 50K is predicted if probability > 0.5.

In [78]:

*# store the probabilities in dataframe*

y\_pred\_prob\_df **=** pd**.**DataFrame(data**=**y\_pred\_prob, columns**=**['Prob of - <=50K', 'Prob of - >50K'])

y\_pred\_prob\_df

Out[78]:

|  | **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 [79]:

*# print the first 10 predicted probabilities for class 1 - Probability of >50K*

gnb**.**predict\_proba(X\_test)[0:10, 1]

Out[79]:

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 [80]:

*# store the predicted probabilities for class 1 - Probability of >50K*

y\_pred1 **=** gnb**.**predict\_proba(X\_test)[:, 1]

In [81]:

*# 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)

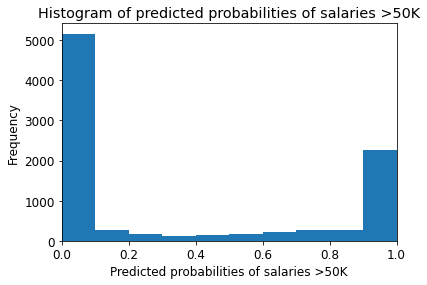
*# set the title*

plt**.**xlabel('Predicted probabilities of salaries >50K')

plt**.**ylabel('Frequency')

Out[81]:

Text(0, 0.5, 'Frequency')



**ROC - AUC**

In [82]:

*# 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--' )

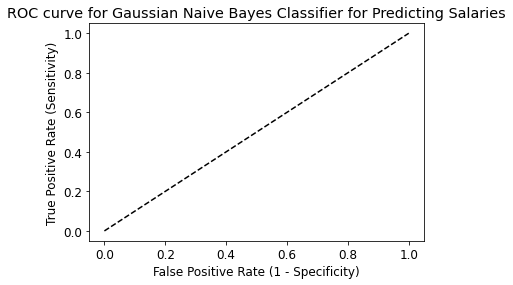
plt**.**rcParams['font.size'] **=** 12

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 [83]:

*# 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 [84]:

*# 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**=**'roc\_auc')**.**mean()

print('Cross validated ROC AUC : {:.4f}'**.**format(Cross\_validated\_ROC\_AUC))

Cross validated ROC AUC : 0.8923

# k-Fold Cross Validation

In [85]:

*# 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 [86]:

*# compute Average cross-validation score*

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

Average cross-validation score: 0.8018