Aadith Lasar

LY CORE-2

2203262

MACHINE LEARNING LAB-4

Title: Develop a Bayesian classifier IRIS dataset dataset

After completion of this experiment students will be able to:

1. To learn bayes theorem
2. To implement Bayesian classifier

Aim: Develop a Bayesian classifier IRIS dataset dataset

Theory:

Bayes Theorem

Bayes' Theorem is a way of finding a probability when we know certain other probabilities.

The formula is:

P(A|B) = *P(A) P(B|A)***P(B)**

Which tells us: how often A happens *given that B happens*, written **P(A|B)**,
When we know: how often B happens *given that A happens*, written **P(B|A)**
and how likely A is on its own, written **P(A)**
and how likely B is on its own, written **P(B)**

Bayes Classifier with example

In machine learning, **naïve Bayes classifiers** are a family of simple "probabilistic classifiers" based on applying Bayes' theorem with strong (naïve) independence assumptions between the features. They are among the simplest Bayesian network models.[1] But they could be coupled with Kernel density estimation and achieve higher accuracy levels.

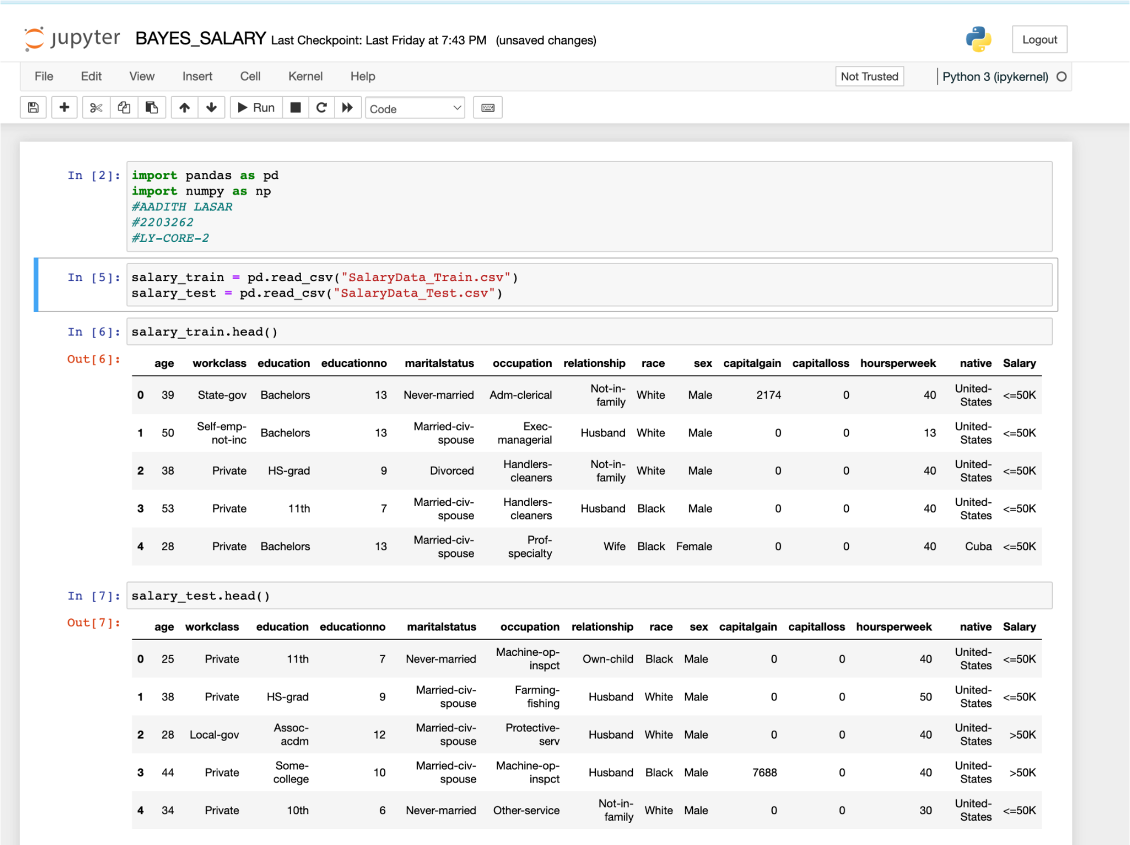
Naive Bayes is a simple technique for constructing classifiers: models that assign class labels to problem instances, represented as vectors of feature values, where the class labels are drawn from some finite set. There is not a single algorithm for training such classifiers, but a family of algorithms based on a common principle: all naive Bayes classifiers assume that the value of a
particular feature is independent of the value of any other feature, given the class variable.

For example, a fruit may be considered to be an apple if it is red, round, and about 10 cm in diameter. A naive Bayes classifier considers each of these features to contribute independently to the probability that this fruit is an apple, regardless of any possible correlations between the color, roundness, and diameter features.

Code

```
import pandas as pd

from sklearn.naive_bayes import GaussianNB
df_=pd.read_csv('IRIS.csv')
df_=df_.dropna(axis=1,how='any')

from sklearn.preprocessing import StandardScaler from
sklearn.model_selection import train_test_split from
sklearn.neural_network import MLPClassifier from
sklearn.preprocessing import LabelEncoder

le = LabelEncoder()

le.fit(["Iris-setosa", "Iris-versicolor", "Iris-viginica"])
le.transform(["Iris-setosa", "Iris-versicolor", "Iris-viginica"])


x=df_.iloc[:,0:3]


obj=StandardScaler()
x_=obj.fit_transform(x)
y=df_['species']

X_train, X_test, y_train, y_test = train_test_split(x_, y, test_size=0.4,rando m_state=42)

X_validate, X_test1, y_validate, y_test1=train_test_split(X_test,y_test,test_s
ize=0.5,random_state=42)

bayes=GaussianNB(priors=None)
bayes.fit(X_train,y_train)
print(bayes.score(X_validate,y_validate))
print(bayes.score(X_test1,y_test1))
```

Results:-


jupyter BAYES_SALARY Last Checkpoint: Last Friday at 7:43 PM (unsaved changes)
Logout
File
Edit
View
Insert
Cell
Kernel
Help
Not Trusted
Python 3 (ipykernel)
Run
Code

```
In [2]: import pandas as pd
        import numpy as np
        #AADITH LASAR
        #2203262
        #LY-CORE-2
```

```
In [5]: salary_train = pd.read_csv("SalaryData_Train.csv")
        salary_test = pd.read_csv("SalaryData_Test.csv")
```

```
In [6]: salary_train.head()
```

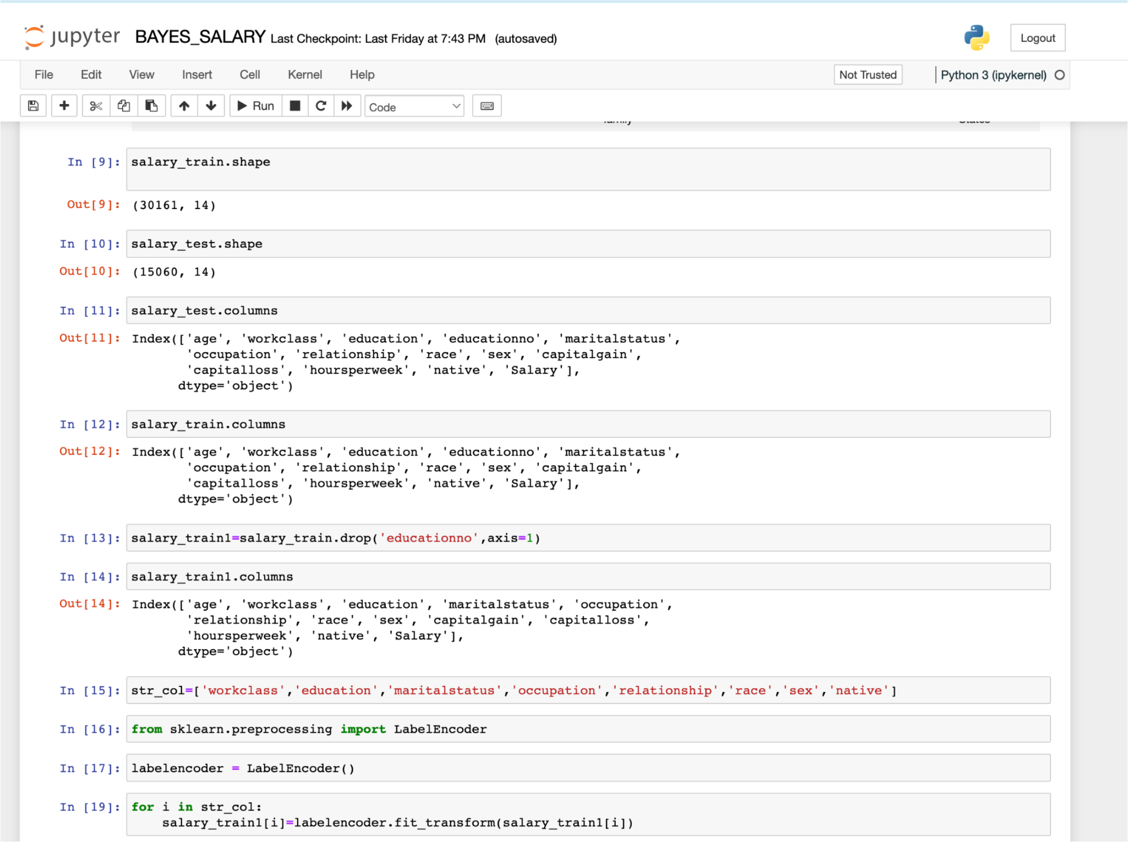
Out[6]:

	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

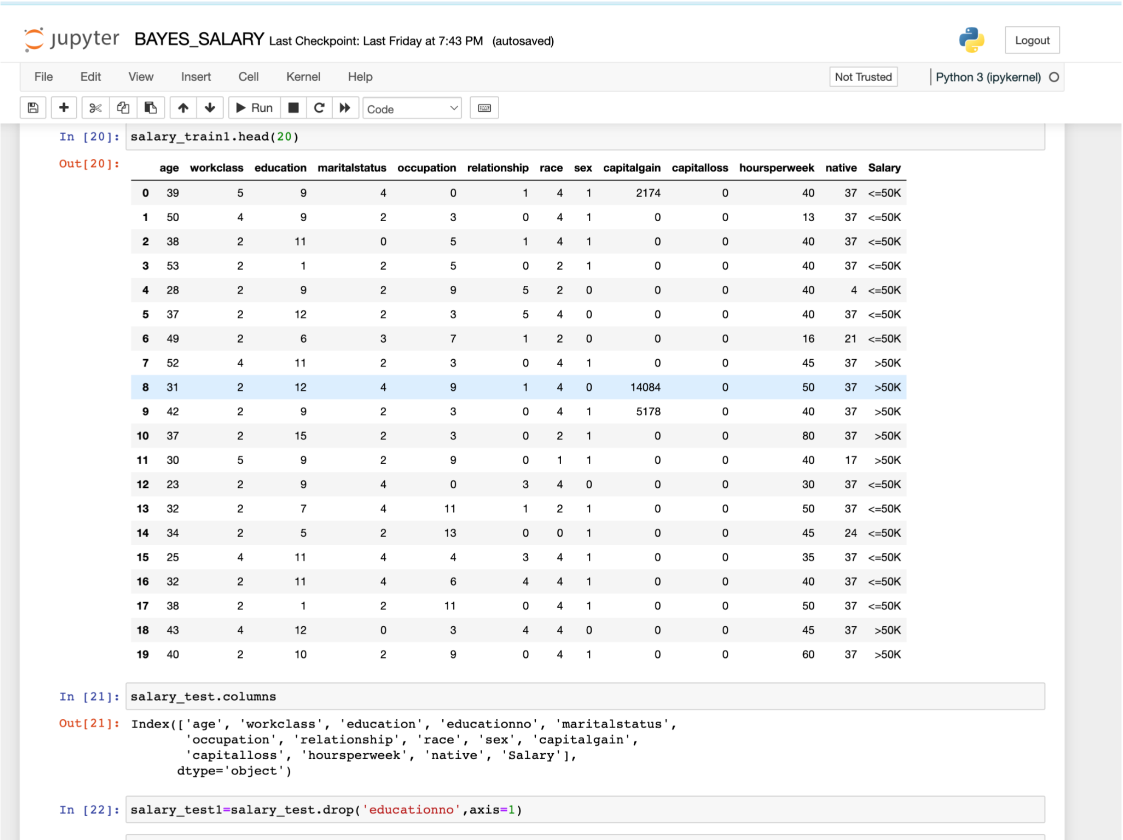
```
In [7]: salary_test.head()
```

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



```
In [9]: salary_train.shape

Out[9]: (30161, 14)

In [10]: salary_test.shape
Out[10]: (15060, 14)

In [11]: salary_test.columns
Out[11]: Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
                'occupation', 'relationship', 'race', 'sex', 'capitalgain',
                'capitalloss', 'hoursperweek', 'native', 'Salary'],
               dtype='object')

In [12]: salary_train.columns
Out[12]: Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
                'occupation', 'relationship', 'race', 'sex', 'capitalgain',
                'capitalloss', 'hoursperweek', 'native', 'Salary'],
               dtype='object')

In [13]: salary_train1=salary_train.drop('educationno',axis=1)

In [14]: salary_train1.columns
Out[14]: Index(['age', 'workclass', 'education', 'maritalstatus', 'occupation',
                'relationship', 'race', 'sex', 'capitalgain', 'capitalloss',
                'hoursperweek', 'native', 'Salary'],
               dtype='object')

In [15]: str_col=['workclass','education','maritalstatus','occupation','relationship','race','sex','native']

In [16]: from sklearn.preprocessing import LabelEncoder

In [17]: labelencoder = LabelEncoder()

In [19]: for i in str_col:
             salary_train1[i]=labelencoder.fit_transform(salary_train1[i])
```

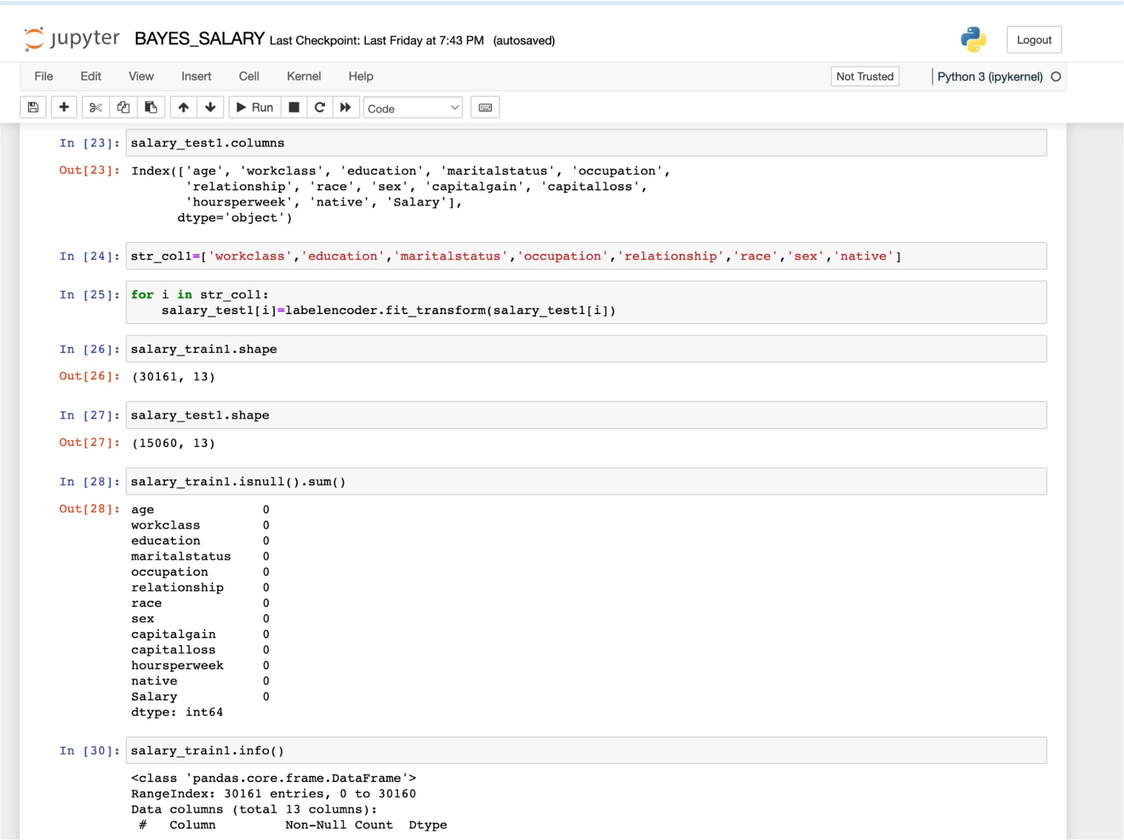


```
In [20]: salary_train1.head(20)
```

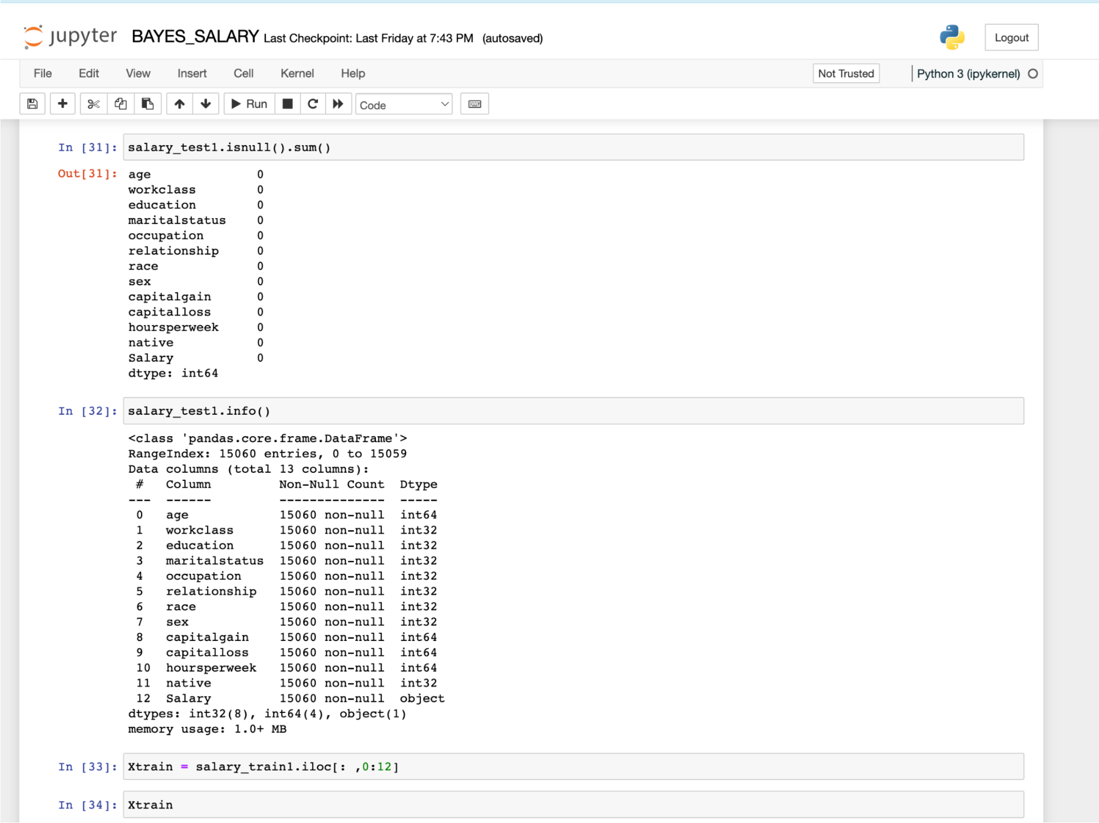
Out[20]:

	age	workclass	education	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native	Salary
0	39	5	9	4	0	1	4	1	2174	0	40	37	<=50K
1	50	4	9	2	3	0	4	1	0	0	13	37	<=50K
2	38	2	11	0	5	1	4	1	0	0	40	37	<=50K
3	53	2	1	2	5	0	2	1	0	0	40	37	<=50K
4	28	2	9	2	9	5	2	0	0	0	40	4	<=50K
5	37	2	12	2	3	5	4	0	0	0	40	37	<=50K
6	49	2	6	3	7	1	2	0	0	0	16	21	<=50K
7	52	4	11	2	3	0	4	1	0	0	45	37	>50K
8	31	2	12	4	9	1	4	0	14084	0	50	37	>50K
9	42	2	9	2	3	0	4	1	5178	0	40	37	>50K
10	37	2	15	2	3	0	2	1	0	0	80	37	>50K
11	30	5	9	2	9	0	1	1	0	0	40	17	>50K
12	23	2	9	4	0	3	4	0	0	0	30	37	<=50K
13	32	2	7	4	11	1	2	1	0	0	50	37	<=50K
14	34	2	5	2	13	0	0	1	0	0	45	24	<=50K
15	25	4	11	4	4	3	4	1	0	0	35	37	<=50K
16	32	2	11	4	6	4	4	1	0	0	40	37	<=50K
17	38	2	1	2	11	0	4	1	0	0	50	37	<=50K
18	43	4	12	0	3	4	4	0	0	0	45	37	>50K
19	40	2	10	2	9	0	4	1	0	0	60	37	>50K

```
In [21]: salary_test.columns

Out[21]: Index(['age', 'workclass', 'education', 'educationno', 'maritalstatus',
                'occupation', 'relationship', 'race', 'sex', 'capitalgain',
                'capitalloss', 'hoursperweek', 'native', 'Salary'],
               dtype='object')

In [22]: salary_test1=salary_test.drop('educationno',axis=1)
```



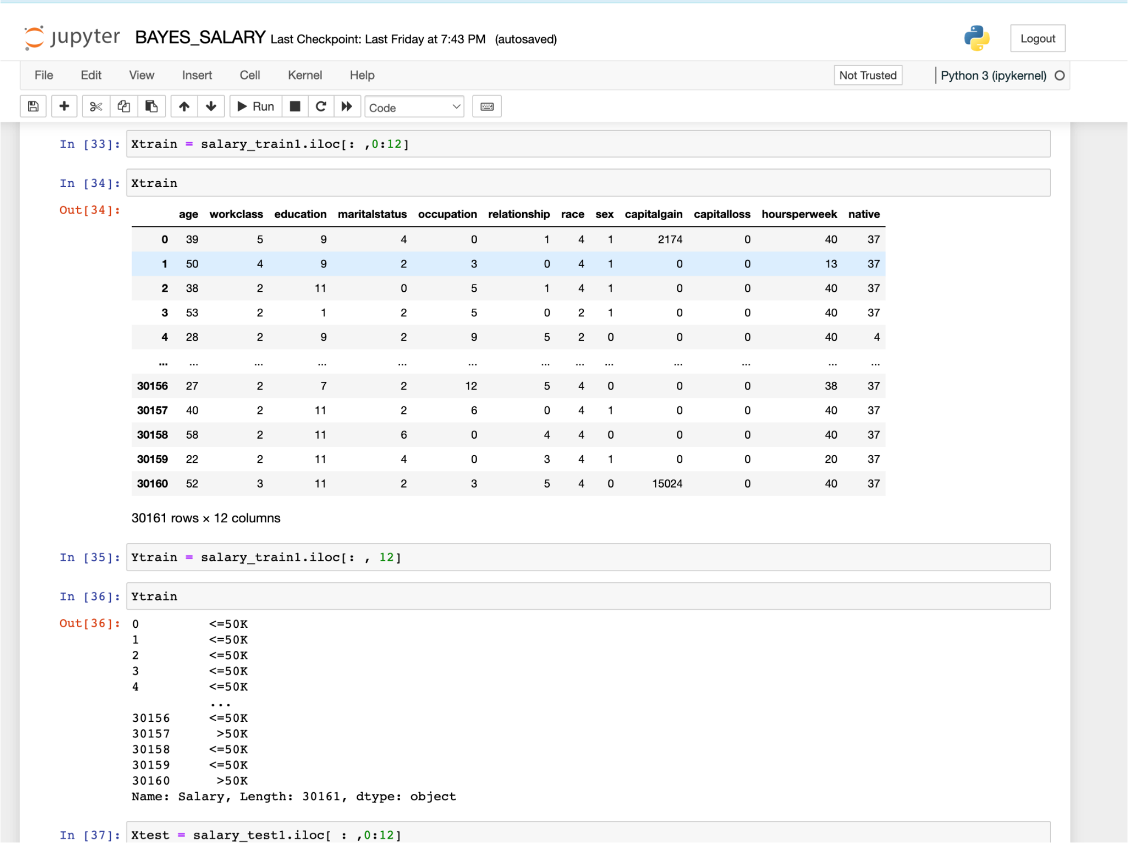
```
In [23]: salary_test1.columns

Out[23]: Index(['age', 'workclass', 'education', 'maritalstatus', 'occupation',
                'relationship', 'race', 'sex', 'capitalgain', 'capitalloss',
                'hoursperweek', 'native', 'Salary'],
               dtype='object')

In [24]: str_coll=['workclass','education','maritalstatus','occupation','relationship','race','sex','native']

In [25]: for i in str_coll:
             salary_test1[i]=labelencoder.fit_transform(salary_test1[i])

In [26]: salary_train1.shape

Out[26]: (30161, 13)

In [27]: salary_test1.shape

Out[27]: (15060, 13)

In [28]: salary_train1.isnull().sum()

Out[28]: age              0
         workclass        0
         education        0
         maritalstatus    0
         occupation       0
         relationship     0
         race             0
         sex              0
         capitalgain      0
         capitalloss      0
         hoursperweek     0
         native           0
         Salary           0
         dtype: int64

In [30]: salary_train1.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 30161 entries, 0 to 30160
         Data columns (total 13 columns):
          #   Column         Non-Null Count  Dtype
```


jupyter BAYES_SALARY Last Checkpoint: Last Friday at 7:43 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel) ○
Run
Code

```
In [31]: salary_test1.isnull().sum()

Out[31]: age              0
         workclass        0
         education        0
         maritalstatus    0
         occupation       0
         relationship     0
         race             0
         sex              0
         capitalgain      0
         capitalloss      0
         hoursperweek     0
         native           0
         Salary           0
         dtype: int64
```

```
In [32]: salary_test1.info()

         <class 'pandas.core.frame.DataFrame'>
         RangeIndex: 15060 entries, 0 to 15059
         Data columns (total 13 columns):
          #   Column         Non-Null Count  Dtype
         ---  ------         --------------  -----
          0   age            15060 non-null  int64
          1   workclass      15060 non-null  int32
          2   education      15060 non-null  int32
          3   maritalstatus  15060 non-null  int32
          4   occupation     15060 non-null  int32
          5   relationship   15060 non-null  int32
          6   race           15060 non-null  int32
          7   sex            15060 non-null  int32
          8   capitalgain    15060 non-null  int64
          9   capitalloss    15060 non-null  int64
          10  hoursperweek   15060 non-null  int64
          11  native         15060 non-null  int32
          12  Salary         15060 non-null  object
         dtypes: int32(8), int64(4), object(1)
         memory usage: 1.0+ MB
```

```
In [33]: Xtrain = salary_train1.iloc[: ,0:12]

In [34]: Xtrain
```


jupyter BAYES_SALARY Last Checkpoint: Last Friday at 7:43 PM (autosaved)
Logout
File Edit View Insert Cell Kernel Help
Not Trusted
Python 3 (ipykernel) ○
Run
Code

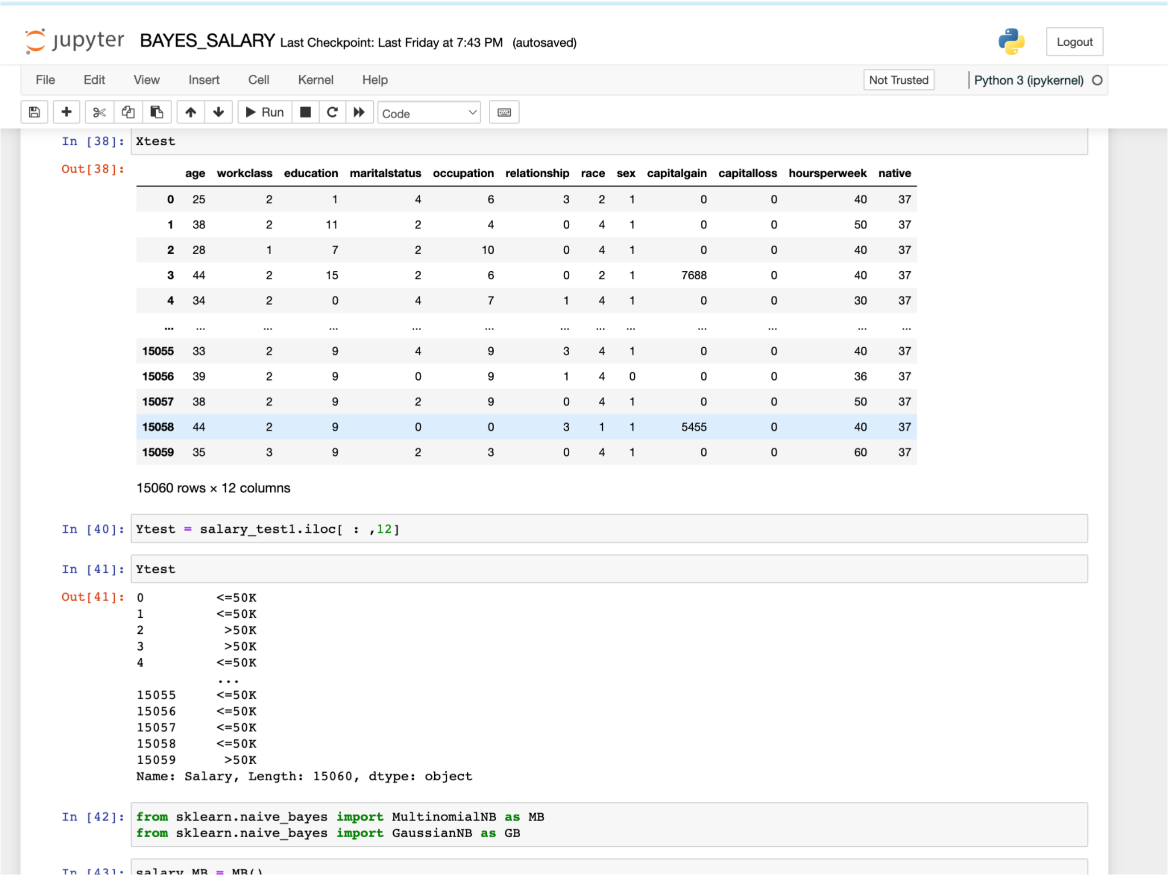
```
In [33]: Xtrain = salary_train1.iloc[: ,0:12]

In [34]: Xtrain
```

Out[34]:

	age	workclass	education	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native
0	39	5	9	4	0	1	4	1	2174	0	40	37
1	50	4	9	2	3	0	4	1	0	0	13	37
2	38	2	11	0	5	1	4	1	0	0	40	37
3	53	2	1	2	5	0	2	1	0	0	40	37
4	28	2	9	2	9	5	2	0	0	0	40	4
...	...	...	...	...	...	...	...	...	...	...	...	...
30156	27	2	7	2	12	5	4	0	0	0	38	37
30157	40	2	11	2	6	0	4	1	0	0	40	37
30158	58	2	11	6	0	4	4	0	0	0	40	37
30159	22	2	11	4	0	3	4	1	0	0	20	37
30160	52	3	11	2	3	5	4	0	15024	0	40	37

30161 rows × 12 columns

```
In [35]: Ytrain = salary_train1.iloc[: , 12]

In [36]: Ytrain

Out[36]: 0         <=50K
         1         <=50K
         2         <=50K
         3         <=50K
         4         <=50K
                   ...
         30156     <=50K
         30157      >50K
         30158     <=50K
         30159     <=50K
         30160      >50K
         Name: Salary, Length: 30161, dtype: object

In [37]: Xtest = salary_test1.iloc[ : ,0:12]
```



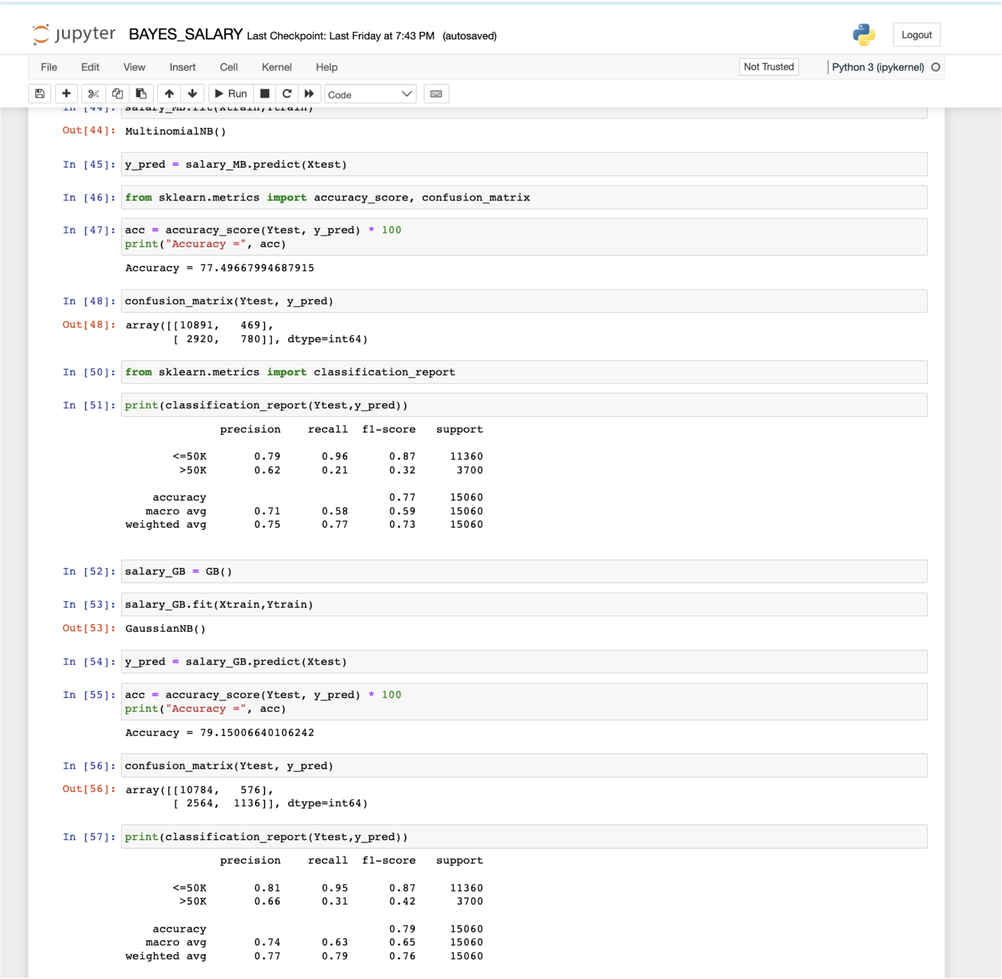
BAYES_SALARY Last Checkpoint: Last Friday at 7:43 PM (autosaved)

```
In [38]: Xtest
Out[38]:
```

	age	workclass	education	maritalstatus	occupation	relationship	race	sex	capitalgain	capitalloss	hoursperweek	native
0	25	2	1	4	6	3	2	1	0	0	40	37
1	38	2	11	2	4	0	4	1	0	0	50	37
2	28	1	7	2	10	0	4	1	0	0	40	37
3	44	2	15	2	6	0	2	1	7688	0	40	37
4	34	2	0	4	7	1	4	1	0	0	30	37
...	...	...	...	...	...	...	...	...	...	...	...	...
15055	33	2	9	4	9	3	4	1	0	0	40	37
15056	39	2	9	0	9	1	4	0	0	0	36	37
15057	38	2	9	2	9	0	4	1	0	0	50	37
15058	44	2	9	0	0	3	1	1	5455	0	40	37
15059	35	3	9	2	3	0	4	1	0	0	60	37

15060 rows × 12 columns

```
In [40]: Ytest = salary_test1.iloc[ : ,12]

In [41]: Ytest

Out[41]: 0         <=50K
         1         <=50K
         2          >50K
         3          >50K
         4         <=50K
                   ...
         15055     <=50K
         15056     <=50K
         15057     <=50K
         15058     <=50K
         15059      >50K
         Name: Salary, Length: 15060, dtype: object

In [42]: from sklearn.naive_bayes import MultinomialNB as MB
         from sklearn.naive_bayes import GaussianNB as GB

In [43]: salary_MB = MB()
```



BAYES_SALARY Last Checkpoint: Last Friday at 7:43 PM (autosaved)

```
In [44]: salary_MB.fit(Xtrain,Ytrain)
Out[44]: MultinomialNB()

In [45]: y_pred = salary_MB.predict(Xtest)

In [46]: from sklearn.metrics import accuracy_score, confusion_matrix

In [47]: acc = accuracy_score(Ytest, y_pred) * 100
         print("Accuracy =", acc)

         Accuracy = 77.49667994687915

In [48]: confusion_matrix(Ytest, y_pred)

Out[48]: array([[10891,   469],
                [ 2920,   780]], dtype=int64)

In [50]: from sklearn.metrics import classification_report

In [51]: print(classification_report(Ytest,y_pred))

                       precision    recall  f1-score   support

                <=50K       0.79      0.96      0.87     11360
                 >50K       0.62      0.21      0.32      3700

             accuracy                           0.77     15060
            macro avg       0.71      0.58      0.59     15060
         weighted avg       0.75      0.77      0.73     15060


In [52]: salary_GB = GB()

In [53]: salary_GB.fit(Xtrain,Ytrain)

Out[53]: GaussianNB()

In [54]: y_pred = salary_GB.predict(Xtest)

In [55]: acc = accuracy_score(Ytest, y_pred) * 100
         print("Accuracy =", acc)

         Accuracy = 79.15006640106242

In [56]: confusion_matrix(Ytest, y_pred)

Out[56]: array([[10784,   576],
                [ 2564,  1136]], dtype=int64)

In [57]: print(classification_report(Ytest,y_pred))

                       precision    recall  f1-score   support

                <=50K       0.81      0.95      0.87     11360
                 >50K       0.66      0.31      0.42      3700

             accuracy                           0.79     15060
            macro avg       0.74      0.63      0.65     15060
         weighted avg       0.77      0.79      0.76     15060
```

Conclusion:

Thus we have successfully completed the implementation of Naïve Bayes Gaussian Classifier.