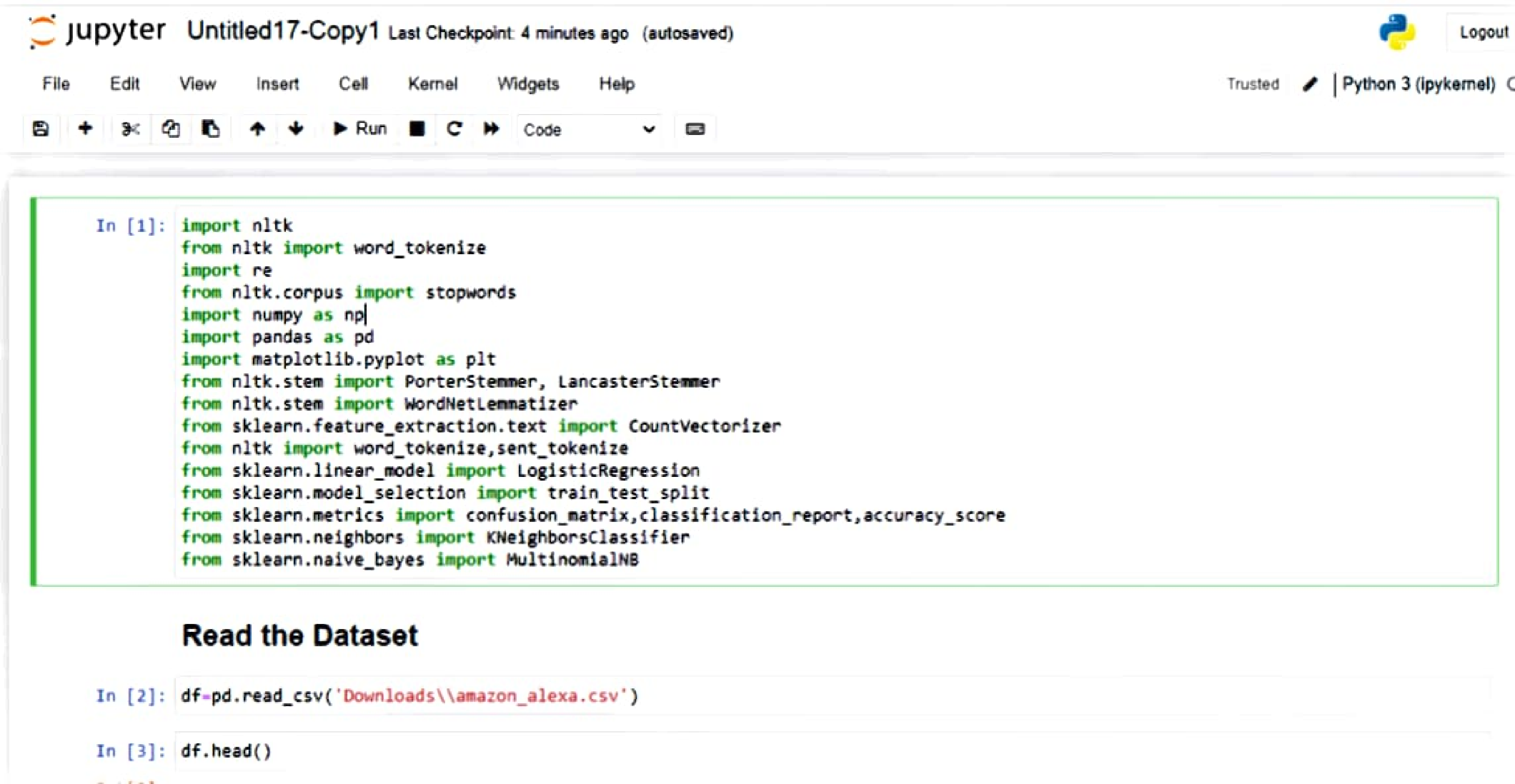


jupyter Untitled17-Copy1 Last Checkpoint: 4 minutes ago (autosaved)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code

```
In [1]: import nltk
        from nltk import word_tokenize
        import re
        from nltk.corpus import stopwords
        import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        from nltk.stem import PorterStemmer, LancasterStemmer
        from nltk.stem import WordNetLemmatizer
        from sklearn.feature_extraction.text import CountVectorizer
        from nltk import word_tokenize,sent_tokenize
        from sklearn.linear_model import LogisticRegression
        from sklearn.model_selection import train_test_split
        from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
        from sklearn.neighbors import KNeighborsClassifier
        from sklearn.naive_bayes import MultinomialNB
```

Read the Dataset

```
In [2]: df=pd.read_csv('Downloads\\amazon_alexa.csv')
```

```
In [3]: df.head()
```

Out[3]:

	reviews	sentiment
0	Love my Echo!	1
1	Loved it!	1
2	Sometimes while playing a game, you can answer...	1
3	I have had a lot of fun with this thing. My 4 ...	1
4	Music	1

Handling Null Values(If any)

In [4]:
```
df.isnull().sum()
```

Out[4]:
```
reviews      0
sentiment    0
dtype: int64
```

Preprocess the Data

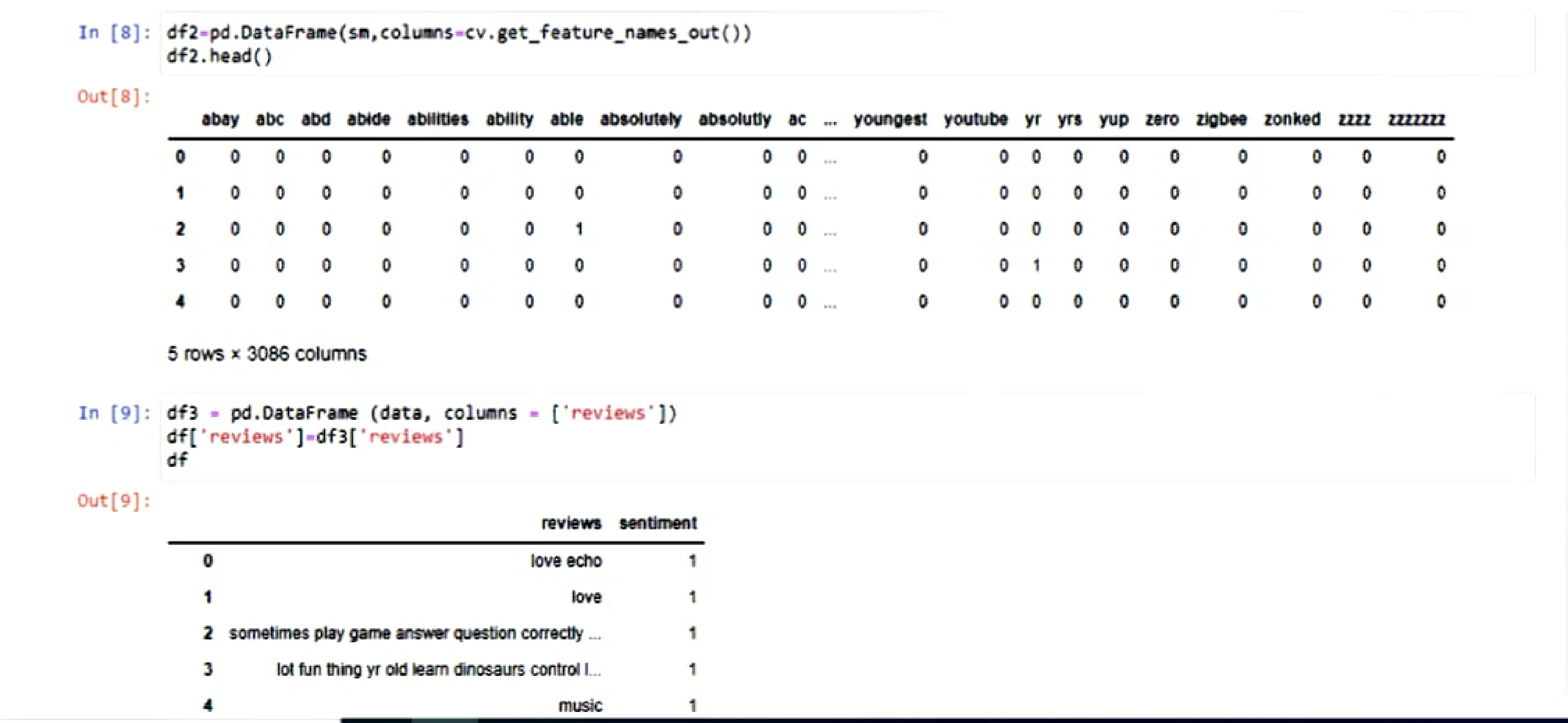
In [5]:
```
sw = stopwords.words('English')
lm = WordNetLemmatizer()
```

```
In [6]: data = []
        for i in df['reviews']:
            t = i.lower()                                 # lower case conversion
            t = re.sub('[^A-Za-z]',' ',t)                 # removing punctuation
            t = word_tokenize(t)                          # word tokenization
            t = [i for i in t if i not in sw]             # stop words removal
            t = [lm.lemmatize(i,pos="v") for i in t]      # lemmatization  # returns list of words
            t = " ".join(t)                               # Joining all the words
            data.append(t)
```

Transforming the Words into Vectors

```
In [7]: cv=CountVectorizer()
        sm=cv.fit_transform(data).toarray()
        print(sm)
        print(cv.get_feature_names_out())
        print(len(cv.get_feature_names_out()))
```

```
[[0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 ...
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]
 [0 0 0 ... 0 0 0]]
['abay' 'abc' 'abd' ... 'zonked' 'zzzz' 'zzzzzzz']
3086
```



In [8]:
```
df2=pd.DataFrame(sm,columns=cv.get_feature_names_out())
df2.head()
```

Out[8]:

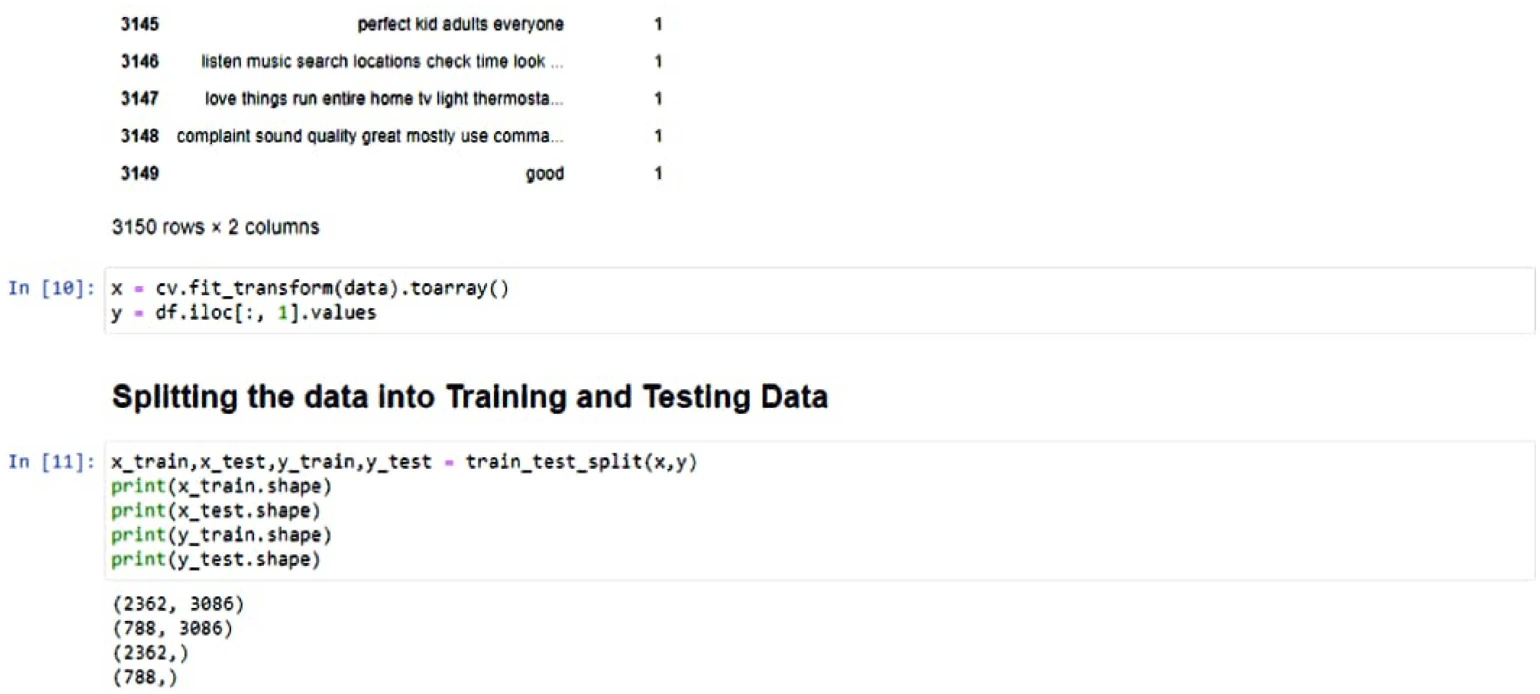
	abay	abc	abd	abide	abilities	ability	able	absolutely	absolutly	ac	...	youngest	youtube	yr	yrs	yup	zero	zigbee	zonked	zzzz	zzzzzzz
0	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	1	0	0	0	...	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

5 rows × 3086 columns

In [9]:
```
df3 = pd.DataFrame (data, columns = ['reviews'])
df['reviews']=df3['reviews']
df
```

Out[9]:

	reviews	sentiment
0	love echo	1
1	love	1
2	sometimes play game answer question correctly ...	1
3	lot fun thing yr old learn dinosaurs control l...	1
4	music	1



3145	perfect kid adults everyone	1
3146	listen music search locations check time look ...	1
3147	love things run entire home tv light thermosta...	1
3148	complaint sound quality great mostly use comma...	1
3149	good	1

3150 rows × 2 columns

```
In [10]: x = cv.fit_transform(data).toarray()
         y = df.iloc[:, 1].values
```

Splitting the data into Training and Testing Data

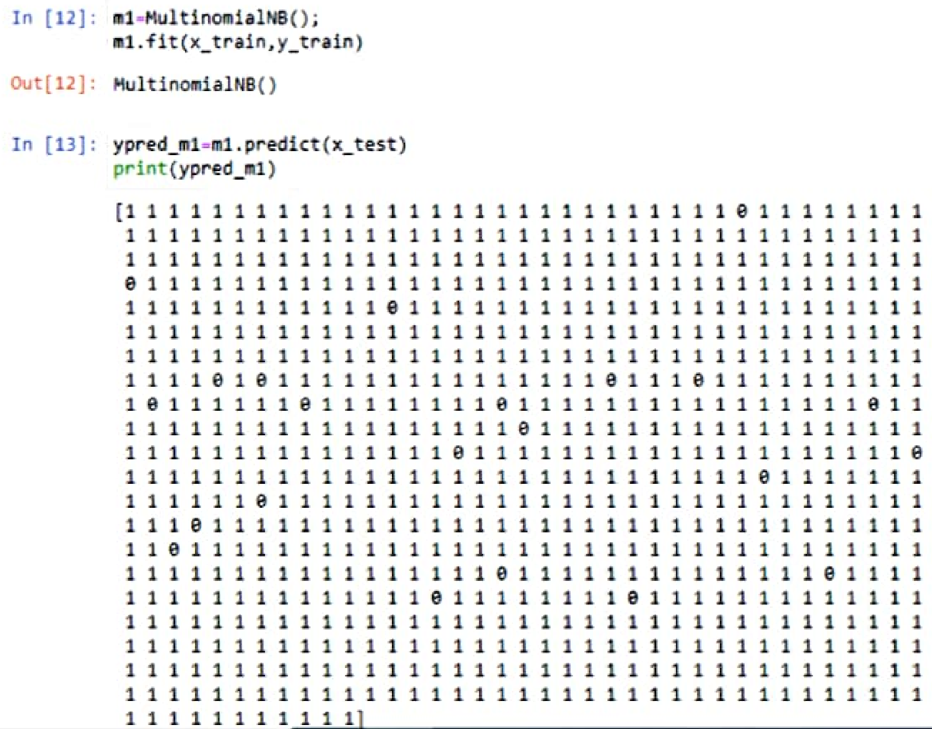
```
In [11]: x_train,x_test,y_train,y_test = train_test_split(x,y)
         print(x_train.shape)
         print(x_test.shape)
         print(y_train.shape)
         print(y_test.shape)
```

```
(2362, 3086)
(788, 3086)
(2362,)
(788,)
```

Applying Multinomial Naive Bayes Classification

```
In [12]: m1=MultinomialNB();
         m1.fit(x_train,y_train)

Out[12]: MultinomialNB()
```

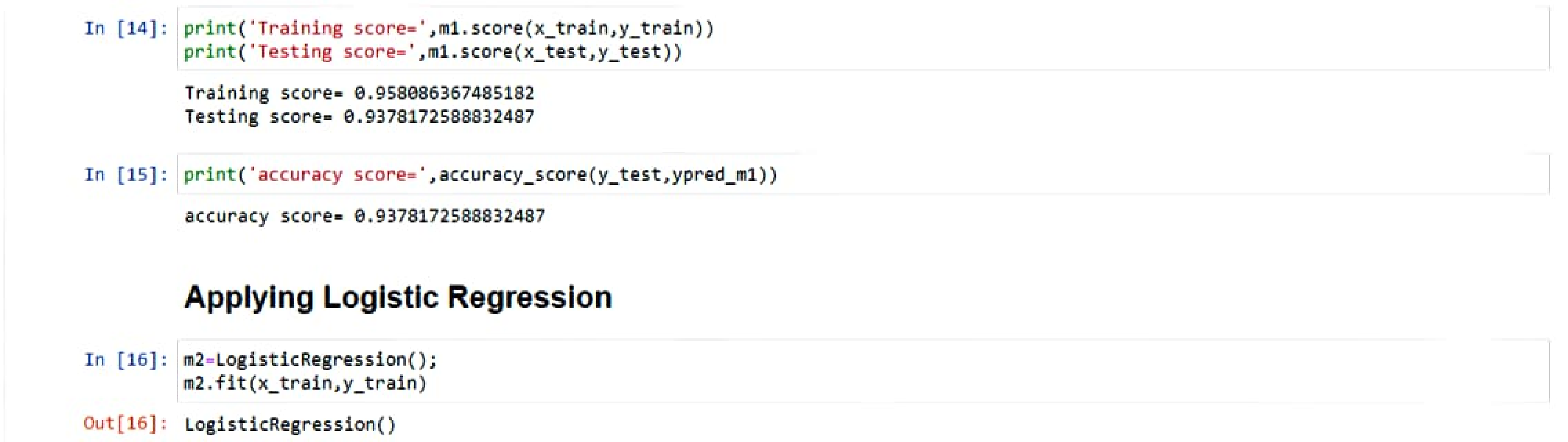


```
In [12]: m1=MultinomialNB();
         m1.fit(x_train,y_train)

Out[12]: MultinomialNB()

In [13]: ypred_m1=m1.predict(x_test)
         print(ypred_m1)

[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1
 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1]
```



```
In [14]: print('Training score=',m1.score(x_train,y_train))
         print('Testing score=',m1.score(x_test,y_test))

         Training score= 0.958086367485182
         Testing score= 0.9378172588832487
```

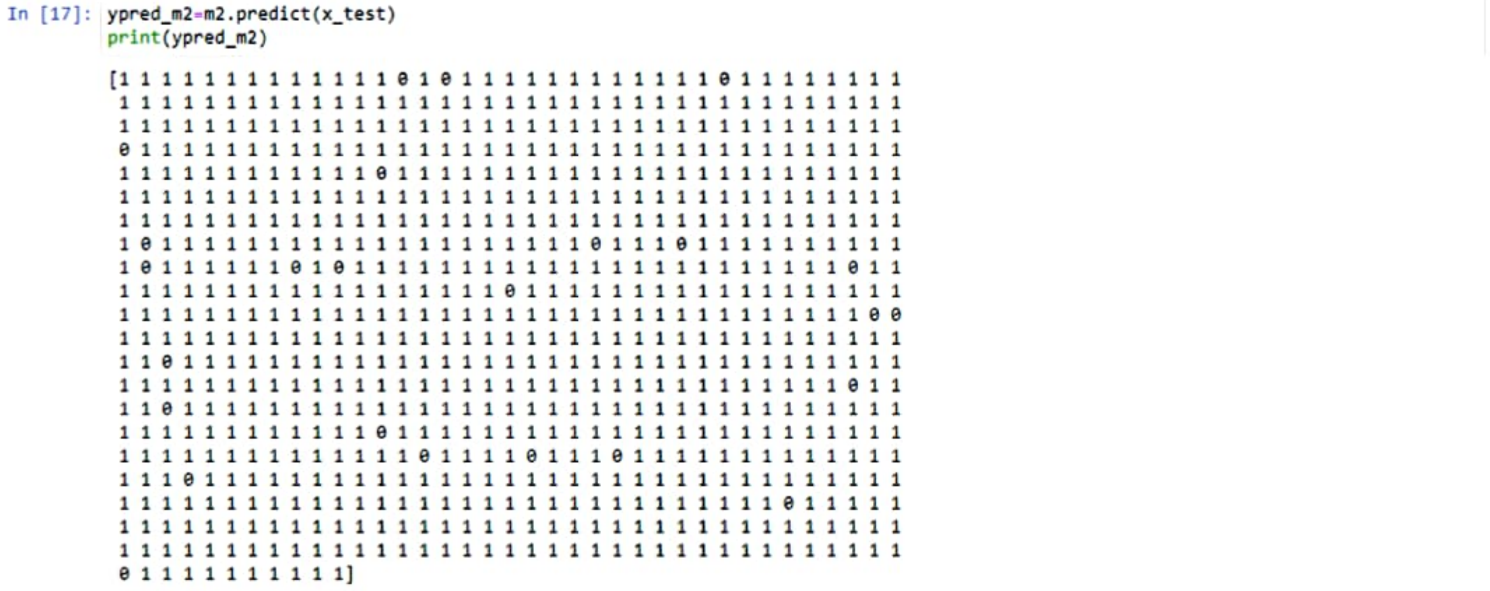
```
In [15]: print('accuracy score=',accuracy_score(y_test,ypred_m1))

         accuracy score= 0.9378172588832487
```

Applying Logistic Regression

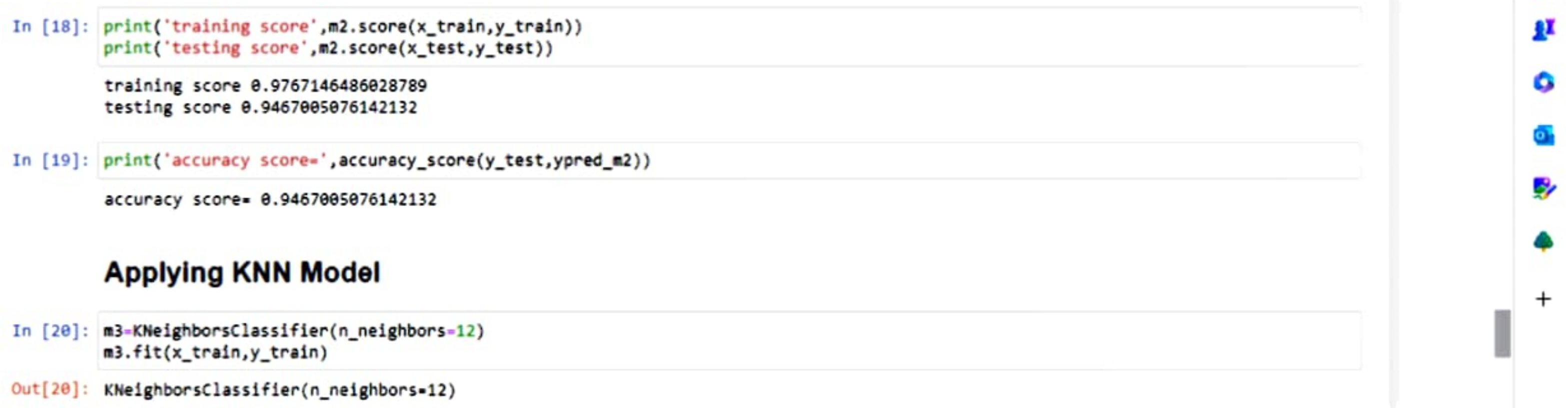
```
In [16]: m2=LogisticRegression();
         m2.fit(x_train,y_train)

Out[16]: LogisticRegression()
```



```
In [17]: ypred_m2=m2.predict(x_test)
         print(ypred_m2)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1
 1 0 1 1 1 1 1 1 0 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1
 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 0 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 0 1 1 1 1 1 1 1 1 1 1]
```



```
In [18]: print('training score',m2.score(x_train,y_train))
         print('testing score',m2.score(x_test,y_test))

         training score 0.9767146486028789
         testing score 0.9467005076142132

In [19]: print('accuracy score=',accuracy_score(y_test,ypred_m2))

         accuracy score= 0.9467005076142132
```

Applying KNN Model

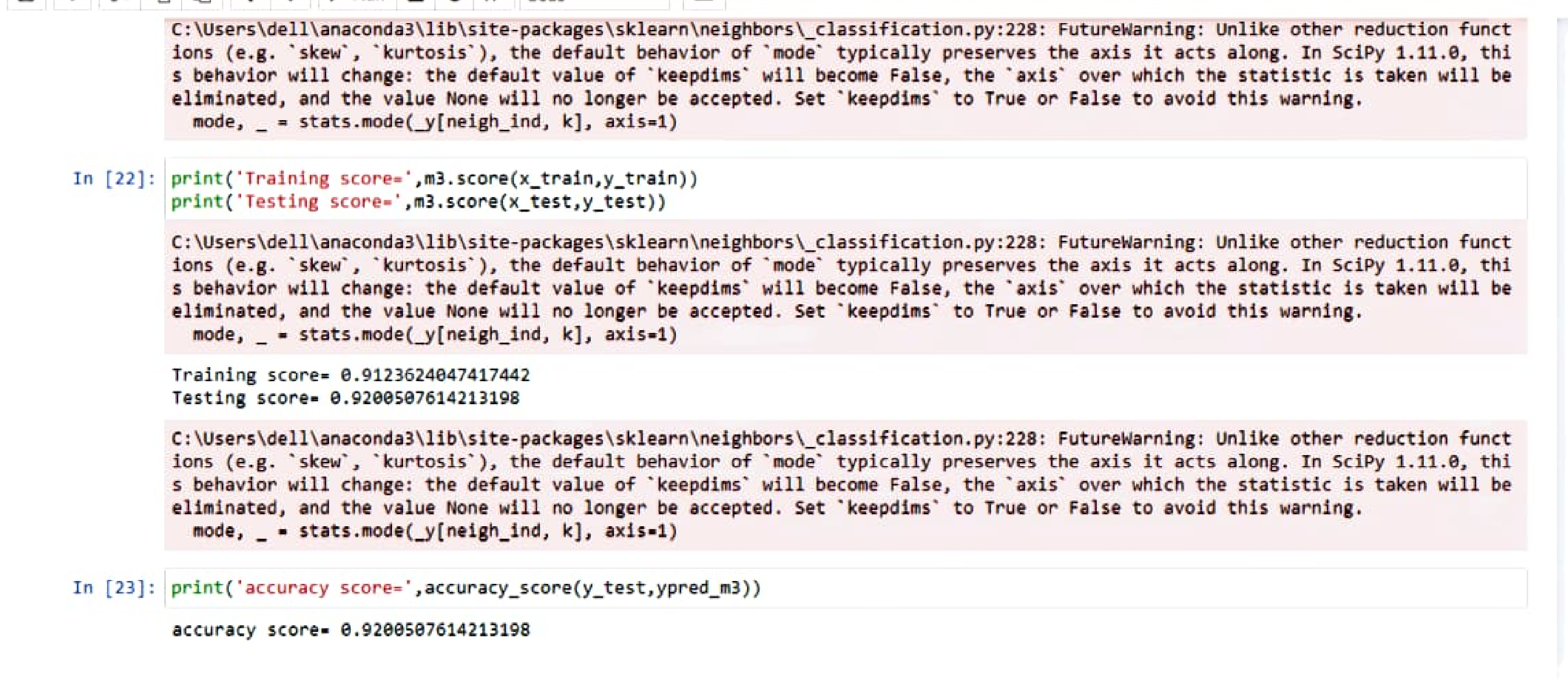
```
In [20]: m3=KNeighborsClassifier(n_neighbors=12)
         m3.fit(x_train,y_train)

Out[20]: KNeighborsClassifier(n_neighbors=12)
```

```
In [21]: ypred_m3=m3.predict(x_test)
         print(ypred_m3)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1]
```

```
C:\Users\dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction funct
ions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, thi
s behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be
eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
```



```
C:\Users\dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction funct
ions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, thi
s behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be
eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
In [22]: print('Training score=',m3.score(x_train,y_train))
         print('Testing score=',m3.score(x_test,y_test))
```

```
C:\Users\dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction funct
ions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, thi
s behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be
eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)

Training score= 0.9123624047417442
Testing score= 0.9200507614213198

C:\Users\dell\anaconda3\lib\site-packages\sklearn\neighbors\_classification.py:228: FutureWarning: Unlike other reduction funct
ions (e.g. `skew`, `kurtosis`), the default behavior of `mode` typically preserves the axis it acts along. In SciPy 1.11.0, thi
s behavior will change: the default value of `keepdims` will become False, the `axis` over which the statistic is taken will be
eliminated, and the value None will no longer be accepted. Set `keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

```
In [23]: print('accuracy score=',accuracy_score(y_test,ypred_m3))
```

```
accuracy score= 0.9200507614213198
```

Computing Confusion Matrix and Classification Report

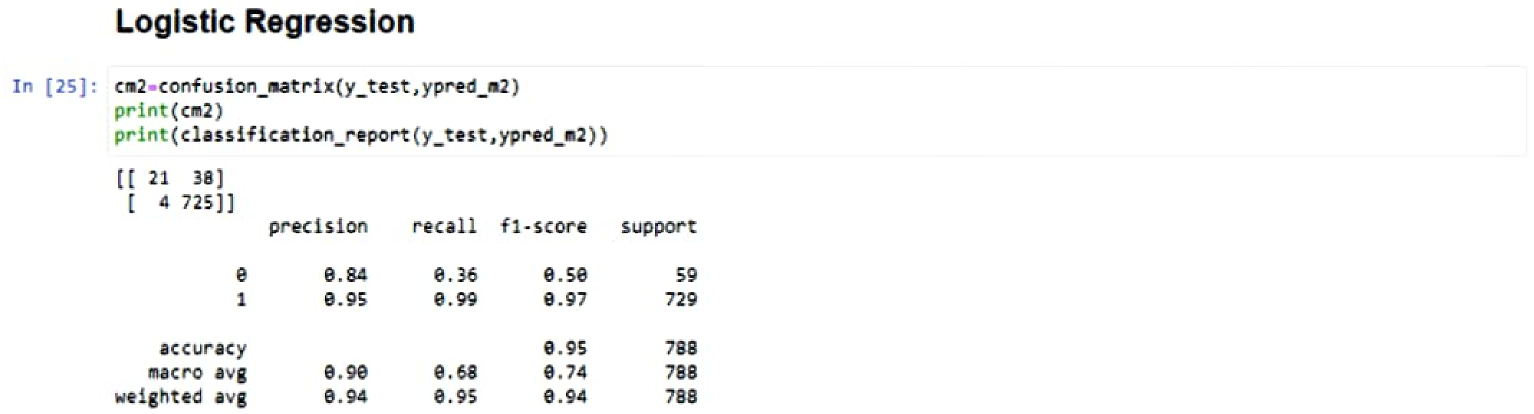
Multinomial Naive Bayes model

In [24]:
```
cm1=confusion_matrix(y_test,ypred_m1)
print(cm1)
print(classification_report(y_test,ypred_m1))
```

```
[[ 16  43]
 [  6 723]]
              precision    recall  f1-score   support

           0       0.73      0.27      0.40        59
           1       0.94      0.99      0.97       729

    accuracy                           0.94       788
   macro avg       0.84      0.63      0.68       788
weighted avg       0.93      0.94      0.92       788
```



Logistic Regression

In [25]:
```
cm2=confusion_matrix(y_test,ypred_m2)
print(cm2)
print(classification_report(y_test,ypred_m2))
```

```
[[ 21  38]
 [  4 725]]
              precision    recall  f1-score   support

           0       0.84      0.36      0.50        59
           1       0.95      0.99      0.97       729

    accuracy                           0.95       788
   macro avg       0.90      0.68      0.74       788
weighted avg       0.94      0.95      0.94       788
```

KNN Model

```
In [26]: cm3=confusion_matrix(y_test,ypred_m3)
         print(cm3)
         print(classification_report(y_test,ypred_m3))
```

```
[[  0  59]
 [  4 725]]
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        59
           1       0.92      0.99      0.96       729

    accuracy                           0.92       788
   macro avg       0.46      0.50      0.48       788
weighted avg       0.86      0.92      0.89       788
```

Best Accuracy Model

From the above Models, the Logistic Regression Model has the best accuracy