Assignment 5: Naive Bayes Learning

Jalendar Reddy Maligireddy

11511290

1.About the model

Naive Bayes is a machine learning model that is used for big amounts of data; even if you are dealing with data that contains millions of records, Naive Bayes is the preferred strategy. When it comes to NLP tasks like sentiment analysis, it produces excellent results. It is a simple and quick categorization method.

P(A|B) = P(B|A).P(A)/P(B)

Where,

P(A): The probability of hypothesis H being true. This is known as the prior probability.

P(B): The probability of the evidence.

P(A|B): The probability of the evidence given that hypothesis is true.

P(B|A): The probability of the hypothesis given that the evidence is true.

A classifier is a machine learning model that distinguishes between different objects based on variable properties.

It is a classifier that is based on the Bayes theorem. Membership probabilities, such as the likelihood of data points belonging to a certain class, are predicted for each class.

The most likely class is designated as the most appropriate class. This is also referred to as Maximum A Posteriori (MAP).

Types of Naive Bayes Algorithms

1. Gaussian Nave Bayes: When characteristic values are continuous in nature, it is assumed that the values associated with each class are scattered according to the Gaussian distribution,

which is the Normal Distribution.

2. Multinomial Naive Bayes: Multinomial Naive Bayes is preferred for usage on multinomial distributed data. It is commonly used in NLP text classification. Each text classification event represents the presence of a word in a document.
3. Bernoulli Naive Bayes: Bernoulli Naive Bayes is employed when data is distributed using multivariate Bernoulli distributions. That is, several characteristics exist, but each is considered to have a binary value. As a result, features must be binary-valued.

Double-click (or enter) to edit

2.Dataset

This is a collection of customer reviews from six of the review topics used in the paper by Blitzer et al., (2007) mentioned above. The data has been formatted so that there is one review per line, and the texts have been split into separate words ("tokens") and lowercased.

url= http://www.cse.chalmers.se/~richajo/dit862/data/all_sentiment_shuffled.txt

Importing the required packages

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Requests allows you to send HTTP/1.1 requests extremely easily. There’s no need to manually add query strings to your URLs, or to form-encode your PUT & POST data — but nowadays, just use the json method!

```
import requests
```

Getting data from url and storing in data_txt

```
#Getting data from URL with request package
request_url = requests.get ('http://www.cse.chalmers.se/~richajo/dit862/data/all_sentiment_sh
data_txt = request_url.text
```

printing th data_txt

```
#Printing the data
data_txt
```

```
    'music neg 241.txt i bought this album because i loved the title song . it \'s such a g
    reat song , how bad can the rest of the album be , right ? well , the rest of the songs
    are just filler and are n\'t worth the money i paid for this . it \'s either shameless
    bubblegum or oversentimentalized depressing tripe . kenny chesney is a popular artist a
    nd as a result he is in the cookie cutter category of the nashville music scene . he
    \'s gotta pump out the albums so the record company can keep lining their pockets while
    the suckers out there keep buying this garbage to perpetuate more garbage coming out of
```

3.Data pre-processing

A line in the file is organized in columns:

• 0: topic category label (books, camera, dvd, health, music, or software)

• 1: sentiment polarity label (pos or neg)

• 2: document identifier

• 3 and on: the words in the document

```
# Initializing the columns
labels = []
category= []
document_id = []
words_review = []
```

Splitting the data_txt into required format with specified columns

```
# Collecting the data into the columns
for line in data_txt.splitlines():
  labels.append(line.split()[1])
  category.append(line.split()[0])
  document_id.append(line.split()[2])
  words_review.append(' '.join(token for token in line.split()[3:]))
```

Creating data frame with columns created

```
# Creating the dataframe with columns ('labels','category','document_id','words_review')
# ziping them
df = pd.DataFrame(zip(labels,category,document_id,words_review ), columns = ['labels','catego
df.head()
```

	labels	category	document_id	words_review
0	neg	music	241.txt	i bought this album because i loved the title ...
1	neg	music	544.txt	i was misled and thought i was buying the enti...
2	neg	books	729.txt	i have introduced many of my ell , high school...
3	pos	books	278.txt	anything you purchase in the left behind serie...
4	pos	dvd	840.txt	i loved these movies , and i cant wiat for the...

Checking any null values in the data

```
# Checking the null values in the dataframe
df.isnull().sum()
```

```
     labels          0
     category        0
     document_id     0
     words_review    0
     dtype: int64
```

4.Data visualizations

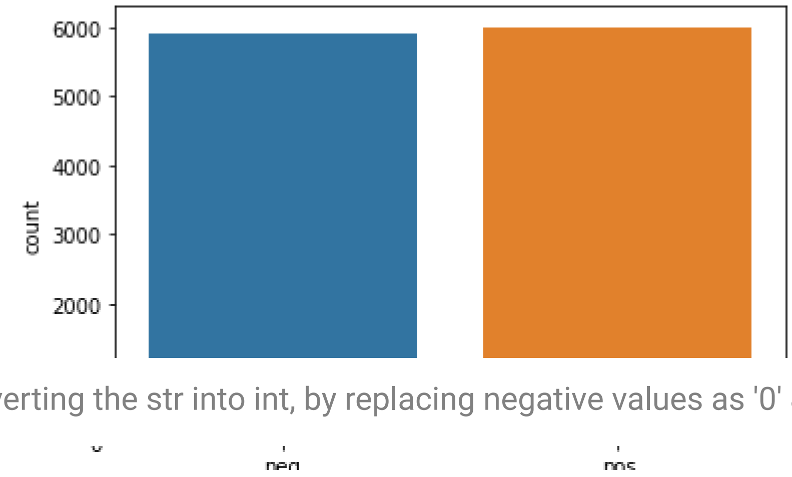
```
#counting the total number different values in the lables column
count_labels = df['labels'].value_counts()


#Printing the count
count_labels
```

```
     pos    6000
     neg    5914
     Name: labels, dtype: int64
```

Creating a plot using seaborn, for the negative & postive values in the labels column

```
# creating a plot for the count values with lables for negative & postive
sns.countplot(data = df, x = 'labels')
plt.show();
```



converting the str into int, by replacing negative values as '0' & postive as '1'

```
#assiging neg as '0' & pos as '1'
df['labels'] = np.where(df['labels']!='pos', 0, 1)
```

```
#printing the df
df
```

	labels	category	document_id	words_review
0	0	music	241.txt	i bought this album because i loved the title ...
1	0	music	544.txt	i was misled and thought i was buying the enti...
2	0	books	729.txt	i have introduced many of my ell , high school...
3	1	books	278.txt	anything you purchase in the left behind serie...
4	1	dvd	840.txt	i loved these movies , and i cant wiat for the...
...	...	...	...	...
11909	0	dvd	53.txt	the story here dose n't matter . the main char...
11910	1	software	70.txt	i liked everything about this product except i...
11911	1	camera	864.txt	this flash is the perfect back-up for a studio...
11912	0	health	264.txt	i had boughten this as a gift which turned out...
11913	0	health	83.txt	the pedometer arrive held prisoner in a diffic...

11914 rows × 4 columns

5.Splitting the data

The Bayes theorem is applied with the "naive" assumption of conditional independence between every pair of features given the value of the class variable in Navies Bayes approaches.

```
# Without cleaning data
```

Importing the train_test_split module from sklearn package

```
from sklearn.model_selection import train_test_split
```

Declaring the words to x & labels to y

```
# word_review to X value
X = df['words_review']
# labels to y values
y = df['labels']
```

splitting the data with test size as 20% and random state as 42

```
# creating test and train data using train_test_split model with train size 80% & test size 2
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.2, random_state = 42
```

```
# Getting the shapes of train & test
print ('The shapes of X_train, y_train: ', X_train.shape, y_train.shape)
print ('The shapes of X_test, y_test: ', X_test.shape, y_test.shape)

     The shapes of X_train, y_train:  (9531,) (9531,)
     The shapes of X_test, y_test:  (2383,) (2383,)
```

6.Applying the model

Creating matrix of token count from collection of text document

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(lowercase = False)
```

Fitting the vectorizer model for training data

```
# the vectorizer model into the training set
```

```
X_train_vec_tfrm = vectorizer.fit_transform(X_train)
#Creating dense matrix for train data
X_train_vec_tfrm_den_mtrx = X_train_vec_tfrm.toarray()
```

```
# Printing the shapes
# the shape for tranformed X_train data
print ('The shape of transformed X_train: ', X_train_vec_tfrm_den_mtrx.shape)
# the shape for transformed y_train data
print ('The shape of transformed y_train: ', y_train.shape)
```

```
The shape of transformed X_train:  (9531, 42109)
The shape of transformed y_train:  (9531,)
```

Fitting the vectorizer model for test data

```
# the vectorizer model into the test set
X_test_vec_tfrm = vectorizer.transform(X_test)
#Creating dense matrix for test data
X_test_vec_tfrm_den_mtrx = X_test_vec_tfrm.toarray()
# the shape of transformed X_test data
print ('The shape of transformed X_test: ', X_test_vec_tfrm_den_mtrx.shape)
# the shape of transformed y_test data
print ('The shape of transformed y_test: ', y_test.shape)
```

```
The shape of transformed X_test:  (2383, 42109)
The shape of transformed y_test:  (2383,)
```

```
# Printing the length of vectorizer feature names
print('The total no. of features: ', len(vectorizer.get_feature_names()))
```

```
The total no. of features:  42109
/usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
  warnings.warn(msg, category=FutureWarning)
```

Creating a Naive Bayes models forr the text classififcation

```
# Importing the multinomial navie bayes module from sklearn
from sklearn.naive_bayes import MultinomialNB
M_nav_bys = MultinomialNB()
```

```
# Fitting the model into the training set
M_nav_bys.fit(X_train_vec_tfrm_den_mtrx, y_train)
```

```
MultinomialNB()
```

```
# Training and validating the model using k-fold cross validation
from sklearn.model_selection import cross_validate
cv = cross_validate (M_nav_bys, X_train_vec_tfrm_den_mtrx, y_train, cv = 8)
```

7.Prediction results and test scores

Printing the accuracy score for cv=8 & mean

```
print("the accuracy score of 8-fold cross validation:", cv['test_score'])
```

```
the accuracy score of 8-fold cross validation: [0.80536913 0.81291946 0.8011745  0.7959
 0.82115869 0.79429051]
```

```
print("the cross validation accuracy mean score: ", cv['test_score'].mean())
```

```
the cross validation accuracy mean score:  0.801908116804445
```

```
# the learnt model to make prediction on the test set
y_pred = M_nav_bys.predict(X_test_vec_tfrm_den_mtrx)
```

```
# Evaluating the model on the test set
print('the Accuracy of model in the test set : {:.3f}'.format(M_nav_bys.score(X_test_vec_tfrm
```

```
the Accuracy of model in the test set : 0.822
```

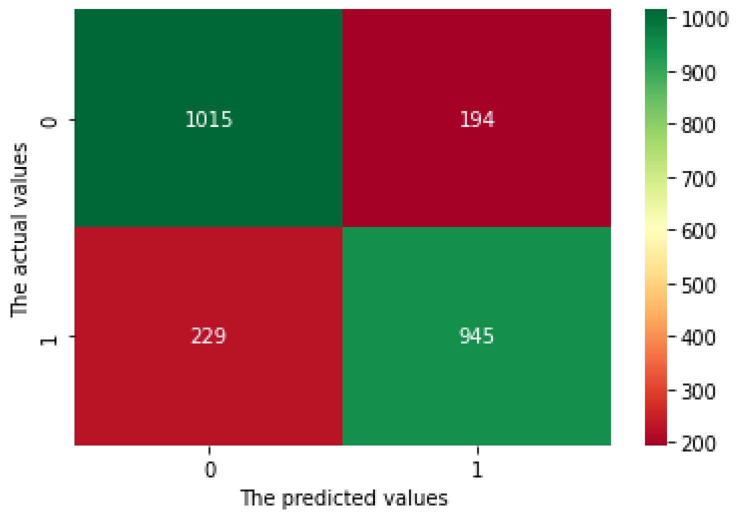
8.Metrics

The confusion matrix for y_test & y_train

```
# confusion matrix
from sklearn.metrics import confusion_matrix
confn_matrix = confusion_matrix(y_test, y_pred)
```

Creating a image for confusion matrix heat map using seaborn package

```
# Visualizing the confusion matrix
sns.heatmap(confn_matrix, cmap="RdYlGn", annot = True, fmt = '.0f')
plt.xlabel ('The predicted values')
plt.ylabel ('The actual values')
plt.show();
```



From the above confusion matrix, model correctly predicted 1015 negative reviews, and 945 positive reviews but the model incorrectly predicts 194 positive reviews as negative and 229 negative reviews as positive.

9. Conclusion

With the ROC curve

An ROC (Receiver Operating Characteristic Curve) is a graph showing the performance of a classification model at all classification thresholds.

Higher the AUC is, the better the model performs. The AUC is expected to greater than 0.5, or over left-top part compared to the baseline.

TPR: TPR is the probability that an actual positive will test positive.

```
TPR = TP/P = TP/(TP+FN)
```

FPR: FPR is the model mistakenly predicted the positive class

```
FPR = FP/N = FP/ (FP+TN)
```

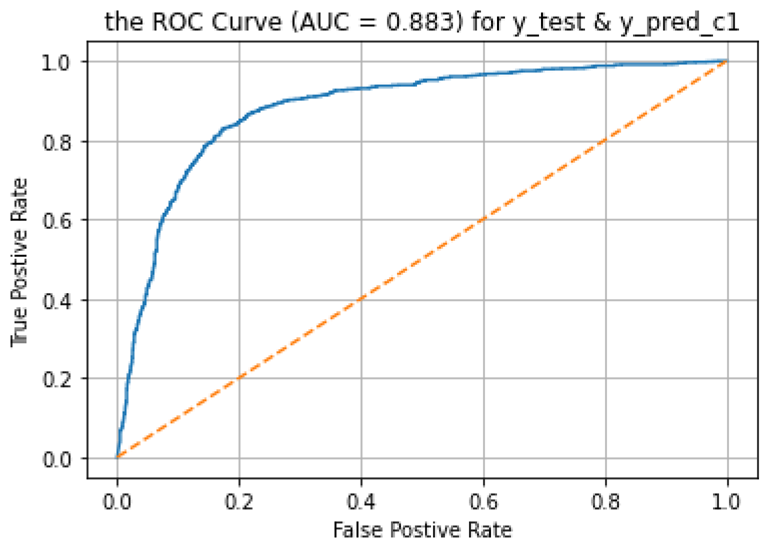
```
# the ROC curve
from sklearn.metrics import roc_curve, roc_auc_score


y_pred_prob_c1  = M_nav_bys.predict_proba (X_test_vec_tfrm_den_mtrx)[:,1]
```

```
false_pstve_rte, true_pstve_rte, threshold = roc_curve (y_test, y_pred_prob_c1, pos_label = 1
roc_auc_score = roc_auc_score (y_test, y_pred_prob_c1)
print('The ROC AUC score for y_test & y_pred_c1: ', roc_auc_score)
```

```
     The ROC AUC score for y_test & y_pred_c1:  0.8826793089308888
```

```
# Visualizing the ROC Curve
plt.plot (false_pstve_rte, true_pstve_rte)
plt.plot([0,1], '--')
plt.xlabel ('False Postive Rate')
plt.ylabel('True Postive Rate')
plt.title ("the ROC Curve (AUC = {:.3f}) for y_test & y_pred_c1".format(roc_auc_score))
plt.grid()
plt.show();
```



It is plotted between the true positive rate and the false positive rate at different thresholds.

The ROC curve area was found to be 0.883.

```
# with cleaning the data
```

```
!pip install stop_words
```

```
     Collecting stop_words
       Downloading stop-words-2018.7.23.tar.gz (31 kB)
     Building wheels for collected packages: stop-words
       Building wheel for stop-words (setup.py) ... done
       Created wheel for stop-words: filename=stop_words-2018.7.23-py3-none-any.whl size=329
       Stored in directory: /root/.cache/pip/wheels/fb/86/b2/277b10b1ce9f73ce15059bf6975d454
     Successfully built stop-words
     Installing collected packages: stop-words
     Successfully installed stop-words-2018.7.23
```

```
from stop_words import get_stop_words
stop_words = get_stop_words('en')
# Removing stopwords, and numerics
df['words_review'] = df['words_review'].apply( lambda x: ' '.join([x for x in str(x).split()
# Removing punctuations and non-letter tokens
df['words_review']=df['words_review'].str.replace('[^\w\s]','')
df['words_review'].head()
```

```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:6: FutureWarning: The defau

0    bought album loved title song  s great song  b...
1    misled thought buying entire cd contains one song
2    introduced many ell  high school students lois...
3    anything purchase left behind series excellent...
4    loved movies  cant wiat third one  funny  suit...
Name: words_review, dtype: object
```

```
from sklearn.model_selection import train_test_split
```

```
# word_review to X value
X = df['words_review']
# labels to y values
y = df['labels']
```

```
# creating test and train data using train_test_split model with train size 80% & test size 2
X_train, X_test, y_train, y_test = train_test_split (X, y, test_size = 0.2, random_state = 42
```

```
# Getting the shapes of train & test
print (' the shapes of X_train, y_train: ', X_train.shape, y_train.shape)
print ('the shapes of X_test, y_test: ', X_test.shape, y_test.shape)
```

```
 the shapes of X_train, y_train:  (9531,) (9531,)
the shapes of X_test, y_test:  (2383,) (2383,)
```

```
from sklearn.feature_extraction.text import CountVectorizer
vectorizer = CountVectorizer(lowercase = False)
```

```
# the vectorizer model into the training set
X_train_vec_tfrm = vectorizer.fit_transform(X_train)
#Creating dense matrix for train data
X_train_vec_tfrm_den_mtrx = X_train_vec_tfrm.toarray()
```

```
# Printing the shapes
# the shape for tranformed X_train data
print ('The shape of transformed X_train: ', X_train_vec_tfrm_den_mtrx.shape)
# the shape for transformed y_train data
print ('The shape of transformed y_train: ', y_train.shape)

     The shape of transformed X_train:  (9531, 47692)
     The shape of transformed y_train:  (9531,)
```

```
# the vectorizer model into the test set
X_test_vec_tfrm = vectorizer.transform(X_test)
#Creating dense matrix for test data
X_test_vec_tfrm_den_mtrx = X_test_vec_tfrm.toarray()
# the shape of transformed X_test data
print ('The shape of transformed X_test: ', X_test_vec_tfrm_den_mtrx.shape)
# the shape of transformed y_test data
print ('The shape of transformed y_test: ', y_test.shape)

     The shape of transformed X_test:  (2383, 47692)
     The shape of transformed y_test:  (2383,)
```

```
# Printing the length of vectorizer feature names
print('The total no. of features: ', len(vectorizer.get_feature_names()))

     The total no. of features:  47692
     /usr/local/lib/python3.7/dist-packages/sklearn/utils/deprecation.py:87: FutureWarning: F
       warnings.warn(msg, category=FutureWarning)
```

```
# Importing the multinomial navie bayes module from sklearn
from sklearn.naive_bayes import MultinomialNB
M_nav_bys = MultinomialNB()
```

```
# Fitting the model into the training set
M_nav_bys.fit(X_train_vec_tfrm_den_mtrx, y_train)

     MultinomialNB()
```

```
# Training and validating the model using k-fold cross validation
from sklearn.model_selection import cross_validate
cv = cross_validate (M_nav_bys, X_train_vec_tfrm_den_mtrx, y_train, cv = 8)
```

```
print("the accuracy score of 8-fold cross validation:", cv['test_score'])

     the accuracy score of 8-fold cross validation: [0.80285235 0.8045302  0.80201342 0.7984
      0.80940386 0.78757347]
```

```
print("the cross validation accuracy mean score: ", cv['test_score'].mean())

     the cross validation accuracy mean score:  0.7981308358550425
```

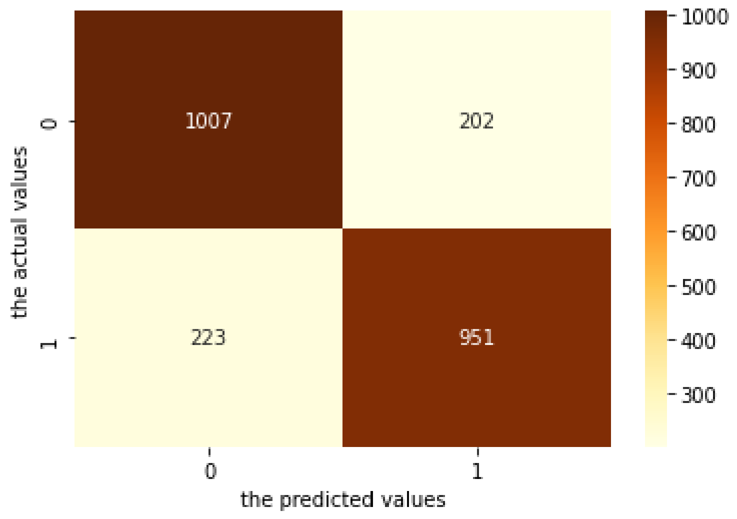
```
# the learnt model to make prediction on the test set
y_pred = M_nav_bys.predict(X_test_vec_tfrm_den_mtrx)
```

```
# Evaluating the model on the test set
print('the Accuracy of model in the test set : {:.3f}'.format(M_nav_bys.score(X_test_vec_tfrm

     the Accuracy of model in the test set : 0.822
```

```
# confusion matrix
from sklearn.metrics import confusion_matrix
confn_matrix = confusion_matrix(y_test, y_pred)
```

```
# Visualizing the confusion matrix
sns.heatmap(confn_matrix, cmap="YlOrBr", annot = True, fmt = '.0f')
plt.xlabel ('the predicted values')
plt.ylabel ('the actual values')
plt.show();
```

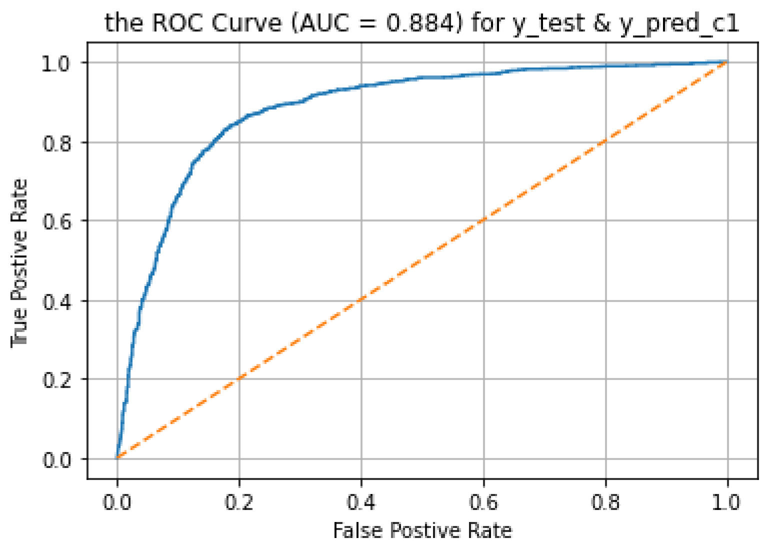


```
# the ROC curve
from sklearn.metrics import roc_curve, roc_auc_score
```

```
y_pred_prob_c1  = M_nav_bys.predict_proba (X_test_vec_tfrm_den_mtrx)[:,1]
false_pstve_rte, true_pstve_rte, threshold = roc_curve (y_test, y_pred_prob_c1, pos_label = 1
roc_auc_score = roc_auc_score (y_test, y_pred_prob_c1)
print('The ROC AUC for y_test & y_pred_c1: ', roc_auc_score)
```

```
The ROC AUC for y_test & y_pred_c1:  0.8840799342805167
```

```
# Visualizing the ROC Curve
plt.plot (false_pstve_rte, true_pstve_rte)
plt.plot([0,1], '--')
plt.xlabel ('False Postive Rate')
plt.ylabel('True Postive Rate')
plt.title ("the ROC Curve (AUC = {:.3f}) for y_test & y_pred_c1".format(roc_auc_score))
plt.grid()
plt.show();
```



It is plotted between the true positive rate and the false positive rate at different thresholds.

The ROC curve area was found to be 0.884.

✓ 0s completed at 6:08 PM