5505 Assignment 5: Naive Bayes Learning

Shiva Chakravarthy Gollapudi
Student ID: 11468697

Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

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

Tools and Environment: Using python programming and Google Colab, I have explored the data.

Step 1: Load Libraries

Import the libraries that we use to load, explore data and build the model.

```
In [1]: import warnings
        warnings.filterwarnings('ignore')

In [2]: from IPython.core.interactiveshell import InteractiveShell
        InteractiveShell.ast_node_interactivity = 'all'

In [3]: import numpy as np
        import pandas as pd
        import matplotlib.pyplot as plt
        import seaborn as sns
        import time

        %matplotlib inline

        plt.style.use('fast')
        sns.set_style('whitegrid')
```

Step 2: Load the data from URL

```
# Load the data from the below URL
re = requests.get ('http://www.cse.chalmers.se/~richajo/dit862/data/all_sentiment_shuffled.txt')
re.encoding = 'utf-8'
data = re.text
```

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

Output:

```
data
```

```
'music neg 241.txt i bought this album because i loved the title song . it \'s such a great 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 and 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 that town . i \'l l get down off my soapbox now . but country music really needs to get back to it \'s roots and stop this pop nonsense . what country music really is and wha t it is considered to be by mainstream are two different things . \nmusic neg 544.txt i was misled and thought i was buying the entire cd and it contains on e song \nbooks neg 729.txt i have introduced many of my ell , ...'
```

Step 3: Data preprocessing

Each line includes topic, sentiment labels, document ids, and reviews. For our analysis reviews and sentiment labels will be sufficient. We can split these reviews and reformat into the dataframe.

```
# Split the data
reviews = []
topics = []
labels = []
doc_id = []

for line in data.splitlines():
  reviews.append(' '.join(token for token in line.split()[3:]))
  topics.append(line.split()[0])
  labels.append(line.split()[1])
  doc_id.append(line.split()[2])

df = pd.DataFrame(zip(topics, reviews, doc_id, labels), columns = ['topic', 'review', 'doc_id', 'label'])
df.head()
```

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

Finding the missing values:

```
# identifying null in dataframe
df.isnull().sum()
```

```
topic      0
review     0
doc_id     0
label      0
dtype: int64
```

There are no null values in the data frame.

Explore the dependent variable:

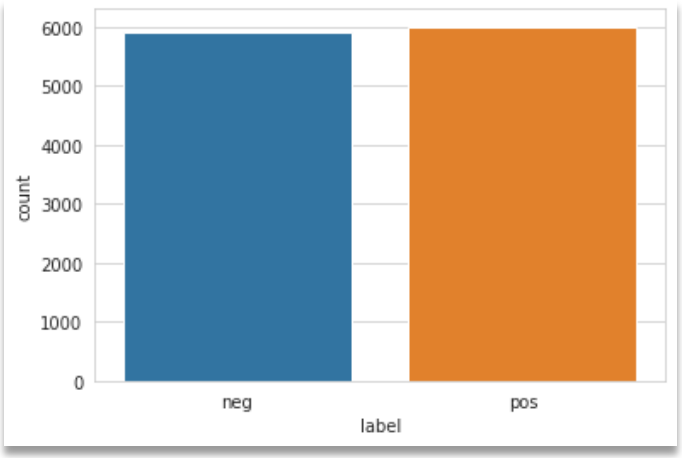
```
#Explore the dependent variable
count = df['label'].value_counts()
count
```

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

There are 6000 positive reviews and 5914 negative reviews.

Visualizing the dependent variable:

```
# Visualize the data
import seaborn as sns
sns.countplot(data = df, x = 'label')
plt.show();
```



Encoding the dependent variable into binary:

```
# Encoding the dependent labels into binary
df['label'] = np.where(df['label']=='pos', 1, 0)
df.head()
```

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

I have replaced the label column with binary, where positive review as 1 and negative as 0

Step 5: Text Cleaning

Text cleaning is a technique that developers use in a variety of domains. Depending on the goal of your project and where you get your data from, you may want to remove unwanted information, such as:

Punctuation and accents
Special characters
Numeric digits
Leading, ending, and vertical whitespace
HTML formatting

```
from stop_words import get_stop_words
stop_words = get_stop_words('en')
df['review'] = df['review'].apply( lambda x: ' '.join([x for x in str(x).split() if x not in stop_words]) ) # Removing stopwords, and numerics
df['review']=df['review'].str.replace('[^\w\s]','') # Removing punctuations and non-letter tokens
df['review'].head()
```

```
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: review, dtype: object
```

Step 6: Model Building

Navies Bayes: Naive Bayes methods are a set of supervised learning algorithms based on applying Bayes' theorem with the "naive" assumption of conditional independence between every pair of features given the value of the class variable.

Splitting the data into training and testing sets Here we will split the data into two parts: training set (80%), and test set (20%).

```
# Splitting the data into training (80%) and test set(20%)
from sklearn.model_selection import train_test_split
X = df['review']
y = df['label']
X_train, X_test, y_train, y_test = train_test_split (X, y, train_size = 0.8, random_state = 42)
print ('Shapes of X_train, y_train: ', X_train.shape, y_train.shape)
print ('Shapes of X_test, y_test: ', X_test.shape, y_test.shape)
```

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

Count Vectorizer: Convert a collection of text documents to a matrix of token counts.

Fitting the vectorizer model into the training set and test set and then returning document-term matrix of that set.

```
# For training set:
X_train_tf = vectorizer.fit_transform(X_train)
X_train_tf_dm = X_train_tf.toarray() # Converting  sparse matrix to a dense matrix
print ('Shapes of transformed X_train, y_train: ', X_train_tf_dm.shape, y_train.shape)

# For test set:
X_test_tf = vectorizer.transform(X_test)
X_test_tf_dm = X_test_tf.toarray() # Converting  sparse matrix to a dense matrix
print ('Shapes of transformed X_test, y_test: ', X_test_tf_dm.shape, y_test.shape)

print('Total number of features: ', len(vectorizer.get_feature_names()))
```

```
Shapes of transformed X_train, y_train:  (9531, 47686) (9531,)
Shapes of transformed X_test, y_test:  (2383, 47686) (2383,)
Total number of features:  47686
```

Multinomial NB classifier: It is a bit more complex but is the usual choice for text-based data. Here you assume that your data is distributed multinomially. Multinomial distribution is a generalized case of binomial, Bernoulli and categorical distributions depending on the number of trials (n) and the number of outcomes (k). If there are only 2 possible outcomes (k=2) and multiple trials (n>1) then your multinomial distribution becomes a binomial distribution.

```
# Create a naive bayes models for text classification
from sklearn.naive_bayes import MultinomialNB
mnb = MultinomialNB()

# Fit the model into the training set
mnb.fit(X_train_tf_dm, y_train)
```

K-fold cross validation:

In K-fold cv, training data is further split into K number of subsets, called folds, then iteratively fit the model k times, each time training the data on k-1 of the folds and evaluating on the kth fold. At the end, we average the performance on each of the folds to come up with final validation metrics for the model.

```
# Training and validating the model using k-fold cross validation
from sklearn.model_selection import cross_validate
cv = cross_validate (mnb, X_train_tf_dm, y_train, cv = 5)
print("accuracy score of 5-fold cross validation:\n", cv['test_score'])
print("cross validation accuracy mean score: \n", cv['test_score'].mean())
```

```
accuracy score of 5-fold cross validation:
 [0.81227058 0.80272823 0.77911857 0.795383   0.79590766]
cross validation accuracy mean score:
 0.79708160854333
```

We got cross validation accuracy mean score as 79.7%, which is not high. But still applied the model to make prediction on the test set.

```
# Applying the learned model to make prediction on the test set
y_pred = mnb.predict(X_test_tf_dm)
```

Step 7: Evaluating the selected model

We used accuracy score, confusion matrix, and ROC Curve to evaluate the model.

Accuracy Scores:

finding the accuracy scores for the test set

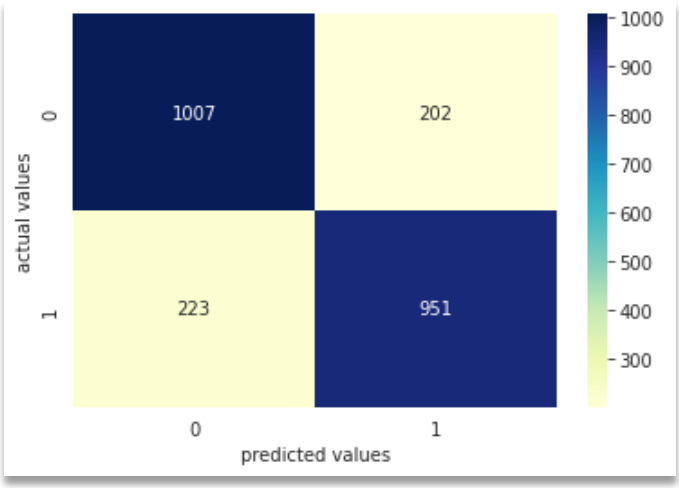
```
# Evaluating the model on the test set
print('Accuracy of the selected model in the test set : {:.4f}'.format(mnb.score(X_test_tf_dm, y_test)))
```

```
Accuracy of the selected model in the test set : 0.8217
```

Accuracy score is 82.1%

Confusion matrix: From the above confusion matrix, model correctly predicted 986 negative reviews, and 927 positive reviews but the model incorrectly predicts 248 positive reviews as negative and 222 negative reviews as positive.

```
# confusion matrix
from sklearn.metrics import confusion_matrix
conf_matrix = confusion_matrix(y_test, y_pred)
# Visualizing the confusion matrix
sns.heatmap(conf_matrix, cmap="YlGnBu", annot = True, fmt = '.0f')
plt.xlabel ('predicted values')
plt.ylabel ('actual values')
plt.show();
```



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.

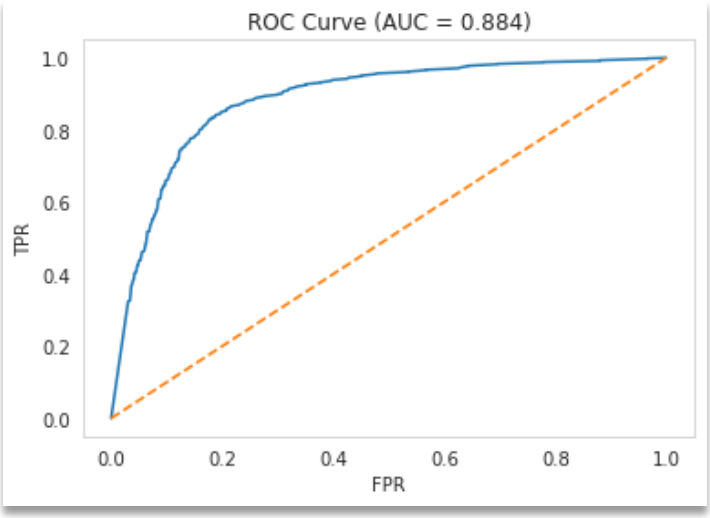
TPR: TPR is the probability that an actual positive will test positive.

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

FPR: FPR is the model mistakenly predicted the positive class

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

Visualizing the ROC curve:



The above AUC curve distinguishes the two classes positive (1) and negative (0).