Natural Language Processing Course

CA#2 (Naïve Bayes) Report

Parnian Fazel – 810198516

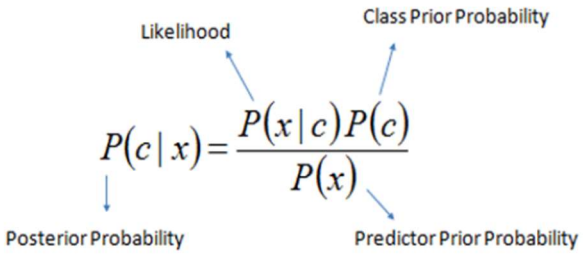
In this computer assignment, we are to classify two datasets using **Naïve Bayes**. Initially, we classify Spam and non-Spam SMS on a dataset. Then, we train a classification tool to distinguish between deceptive and truthful text. Naive bayes is a classification technique based on **Bayes' Theorem** with an assumption of **independence** among predictors. Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature. Also, we get familiar with the **bag of words** method, new libraries like **sklearn** and **nltk**, new methods to normalize text, how to modify our model in order to get better results and how to evaluate our model.

Bag of Words

A **bag-of-words** is a representation of text that describes the occurrence of words within a document. It is called a "bag" of words because, any information about the **order** or structure of words in the document is discarded. The model is only concerned with whether known words occur in the document, not where in the document. So, every word is **independent** from its **position,** and we can ignore the sequence of each word.

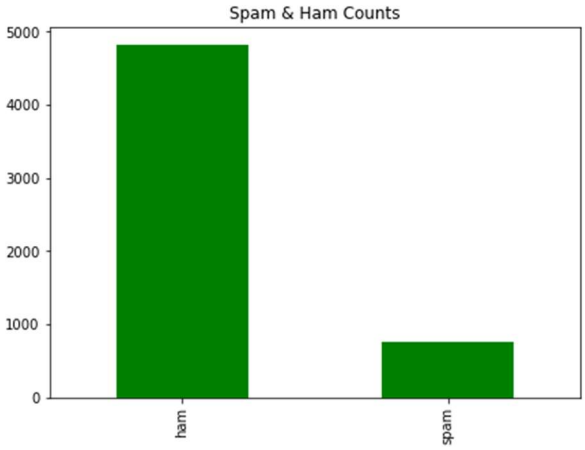
The **features** in this model are the **frequencies** of each word in each class. Hence, as the frequency of a word in class `c` gets higher the probability of choosing class `c` for the that word gets higher.

Consider the given image which is ne `Naive Bayes` formula:



$$P(c|X) = P(x_1|c) \times P(x_2|c) \times \cdots \times P(x_n|c) \times P(c)$$

Spam SMS detection (Applying Naïve Bayes on the first dataset)



Here we can observe that the dataset is **imbalanced**.

Pre-process:

- **Tokenizing** by word_tokenize() in nltk library.
- **Normalizing** and convert all letters to lower case by lower(). This is important because we are using BOW model and naïve bayes and two words "Congratulations" and "congratulations" are now different so, this can hep us to detect same words better and therefore classify better.
- **Removing stop words** like prepositions detecting by nltk_stopwords(). Stop words are words like prepositions which has a high frequency in all classes to be detected so they are frequent in all classes and do not help us to classify hence, we remove them.
- **Lemmatizing** tokens by WordNetLemmatizer() in nltk. Lemmatizaion and stemming are an important part of this pre-processing. Words like "join", "joined" and "joins" are not that different and can be considered as one words "join" to take the meaning and classify better. They Reduce the inflectional forms of each word into a common base or root.

  Stemming algorithms work by cutting off the end or the beginning of the word on the other hand, Lemmatization takes into consideration the morphological analysis of the words. Lemmatization is more **accurate** than stemming but **slower**. Stemming is preferred when the **meaning** of the word is not important for analysis but when the meaning should take into account, we should use lemmatization. Here I employer lemmatization; I got slightly better result by lemmatization than stemmer.

In this part I did not remove punctuations since punctuations effect on classifying. For instance, "!" usually occurs in spam messages.

Feature Extraction:

Feature engineering is one of the most important steps in machine learning. It is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If we can use suitable features and feed them to our model then the model will be able to understand the sentence better. For detecting spam and ham in this dataset I extracted these features(based on my experiments and also searching through internet and this website[1]):

- **Frequency of words:** we use bag of words model. We use the assumption of independency and count the number of words in a record.
- **Message length**: mostly spam messages are longer than non-spam messages. So, this length could be an important feature to extract.
- **Number of uppercased words:** mostly spam messages have uppercased words to get your attention and make you see some keywords such as "FREE" to get curious and read the whole message.
- **URL presence:** usually spam messages contain a URL to invite receivers to checkout that link. In non-spam messages this happens rarely.
- **Phone number presence:** normally we see a phone number in a spam message to invite the user to call to get more information.
- **Currency sign presence:** in a spam message we usually see a price declaration and it is mostly followed by a currency sign like "$"
- **Number of spam keywords:** there are some keywords like "congratulations" or "free" which mainly occur in spam massages. I prepared a list of these words in a "spam_words.txt".

	(Legitimate message)	4 info" *(Spam message)*
URLs	No	No
Punctuation marks	Yes	Yes
Mathematical Symbols	No	Yes
Special symbols	No	Yes
Emoji symbols	Yes	No
Uppercased words	No	Yes
Phone Number	No	No
Special Keywords	No	Yes
Message Length	50	137
Number of Words	11	26

List of extracted features for spam and ham messages

[1] https://www.researchgate.net/figure/List-of-extracted-features-for-spam-and-ham-messages_tbl2_340607093

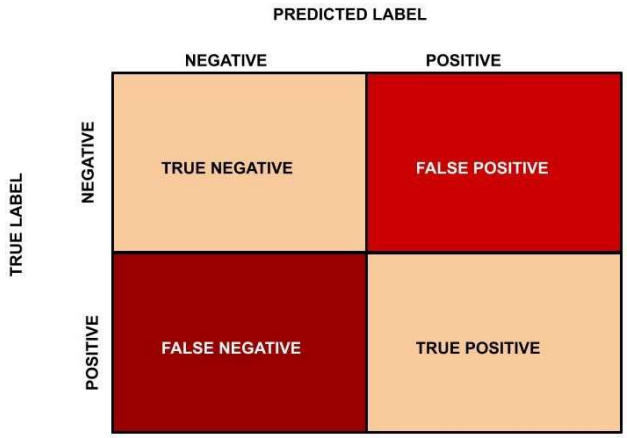
In the first dataset, applying preprocess does not affect drastically on accuracy because the dataset is biased and we still get a accuracy of 98% without applying preprocessing steps on our model!

Training the model

I employed **MultinomialNB()** in **sklearn** library to train naïve bayes classifier. To use this classifier first we should clean dataset and apply preprocessing step then, after extracting features of dataset I split 80% train and 20% test data in the given dataframe train_test_split() in sklearn library. Before training the model, I used **MinMaxScaler()** in sklearn.preprocessing which is used to transform features by scaling each feature to a given range. Normalization involves adjusting values that exist on different scales into a **common scale**, allowing them to be more readily compared.

Evaluation

Here is the confusion matrix labels which we can map it into the formulas below of accuracy, recall, precision and F1 score.



$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$F1\ Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

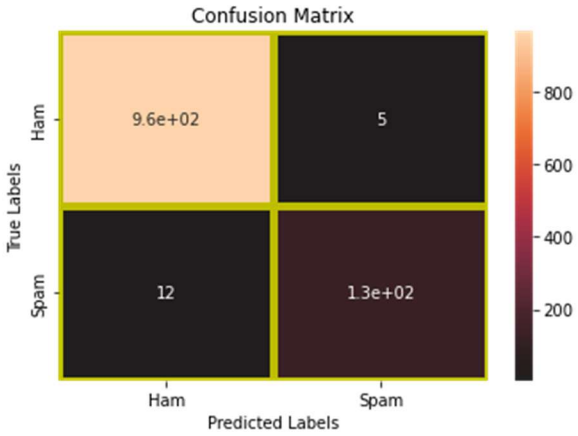
Precision & Recall:

- **Precision**: *Out of all the examples the classifier labeled as positive, what fraction were correct?*
- **Recall:** *Out of all the positive examples there were, what fraction did the classifier pick up?*

F1 score: F1 Score is needed when you want to seek a balance between Precision and Recall. This is the harmonic mean of the two fractions. F1-measure weights precision and recall equally. This measure is approximately the average of the two when they are close but it's not exactly the mean because it punishes extreme values. Precision gives us a measure of the relevant data points, but recall gives a measure of how accurately our model is able to identify the relevant data.

Evaluation on first dataset results:

```
Accuracy: 0.9847533632286996
Recall: 0.9172413793103448
Precision: 0.9637681159420289
F1 score: 0.939929328621908
```



As the matrix shows the **true positives** and **true negatives** are high. The results show that precision is greater that recall. When precision is higher than recall means our model is more **peaky**.

As the plot has shown, the data distribution of ham and spam records are imbalanced. This causes bias toward the dataset with more records (Here is Ham). The classification reports:

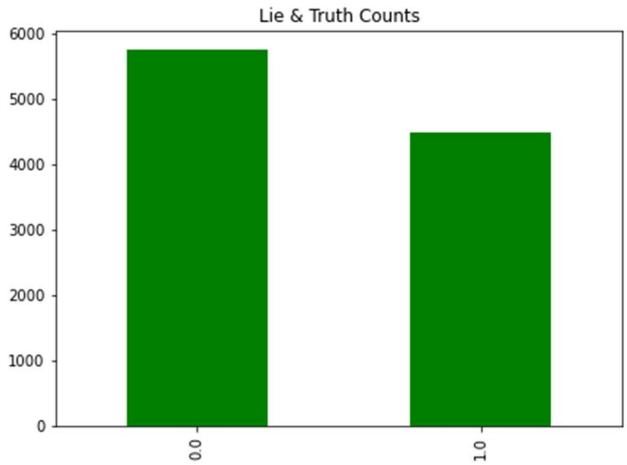
	precision	recall	f1-score	support
0	0.99	0.99	0.99	970
1	0.96	0.92	0.94	145
accuracy			0.98	1115
macro avg	0.98	0.96	0.97	1115
weighted avg	0.98	0.98	0.98	1115

As we can see the scores on 0(ham) is so better than 1(spam) because the dataset is imbalanced. **Accuracy** can be a misleading metric for **imbalanced** data sets. Our model can classify Ham better that spam. Here there are 970 records on Ham and 145 records on spam class in test data (Imbalanced). The F1 score is a better metric to evaluate our which can be calculated by precision and recall (formula given above).

Accuracy, no doubt, is an important metric to consider but it does **not always** give the full picture. For example, consider here we have 970 Ham and 145 Spam, and we have a model that classifies each message as Ham. Then the accuracy of the model is 970/1115 = 86%, meaning that we have a highly *accurate* model, but we know it is not!

Sentimental LIAR detection (Applying Naïve Bayes on the second dataset)

Texts are in the “statement” column and its label is based on the people's vote are in the “label” column. Here we are to consider “barely true” labels, “fire pants” and “FALSE’ as **lie**, and “half-true”, “mostly-true” and “TRUE” labels as the **truth**. Here we may use the sentiment columns as well(like “sentiment_score”, “joy”, “fear” and etc.

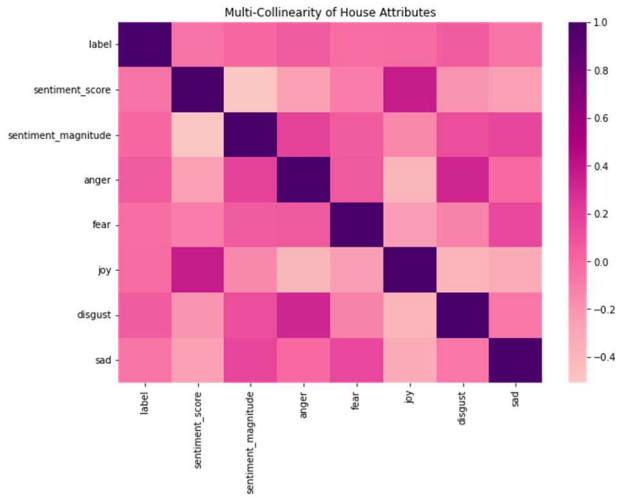


Pre-process:

- **Tokenizing** by word_tokenize() in nltk library.
- **Normalizing** and convert all letters to lower case by lower(), and removing punctuations.
- **Removing stop words** like prepositions detecting by nltk_stopwords().
- **Lemmatizing** tokens by WordNetLemmatizer() in nltk.

Feature Extraction:

Feature engineering is one of the most important steps in machine learning. It is the process of using domain knowledge of the data to create features that make machine learning algorithms work. If we can use suitable features and feed them to our model then the model will be able to understand the sentence better. Beside text we can use other sentiment columns as features. I employed correlation to detect relation between "label" and other features. Here is the plot of **correlation matrix**:



So, the raw dataset of sentiment columns can not be helpful to classify better as there are no column which are highly correlated with label column.

For detecting lie and truth in this dataset I extracted these features(extracting features of this dataset was a little bit tricky but as a searched and read some papers, I extracted features based on this paper[2] and this link[3])

- **Frequency of words:** we use bag of words model. We use the assumption of independency and count the number of words in a record.
- **Number of intensifier words**: intensifiers like "very", "surely and etc can be a feature to detect liying.
- **Number of words in OTHER class(like she, he, …):** according to the paper, desceptive messages has more of these words.

[2] https://dl.acm.org/doi/pdf/10.5555/1667583.1667679
[3] https://research.signal-ai.com/assets/Deception_Detection_with_NLP.pdf

- **Number of words in Self class(like I, mine, …):** according to the paper, truthful messages has more of these words.
- **Positivity or negativity of sentiment_score:** we can see whether a sentiment is positive or negative.
- **Range of 5 sentiments("fear", "joy", "anger", "disgust", "sad"):** the values of these 5 columns are a number between zero and one. Because of the fact we are looking for conditional probabilities in naïve bayes, the occurrence probability of a exact number is zero, so I considered a range for these columns. This range is [0, medianOfColumn) and [medianOfColumn, 1]. If value of a sentiment for a record is less than median the value is set to 0 and if not, the value is set to 1. Now we can extract some features of these columns which can be used by naïve bayes classifier. Here are the median of these columns in train dataset:

    - `anger median = 0.128523`
    - `fear median = 0.10621549999999999`
    - `joy median = 0.128076`
    - `disgust median = 0.1584215`
    - `sad median = 0.28799450000000004`

Class	Score	Sample words
		Deceptive Text
METAPH	1.71	god, die, sacred, mercy, sin, dead, hell, soul, lord, sins
YOU	1.53	you, thou
OTHER	1.47	she, her, they, his, them, him, herself, himself, themselves
HUMANS	1.31	person, child, human, baby, man, girl, humans, individual, male, person, adult
CERTAIN	1.24	always, all, very, truly, completely, totally
		Truthful Text
OPTIM	0.57	best, ready, hope, accepts, accept, determined, accepted, won, super
I	0.59	I, myself, mine
FRIENDS	0.63	friend, companion, body
SELF	0.64	our, myself, mine, ours
INSIGHT	0.65	believe, think, know, see, understand, found, thought, feels, admit

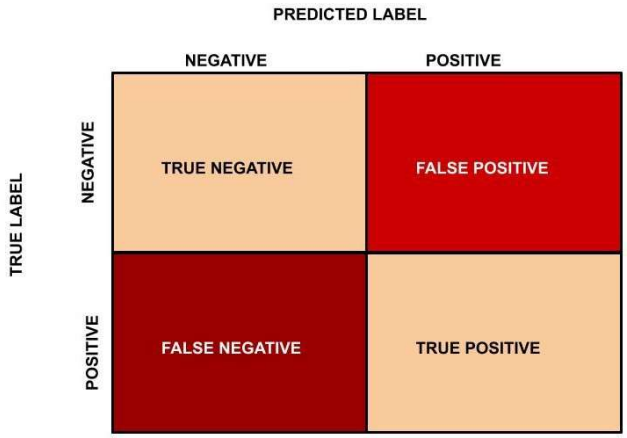
Table 4: Dominant word classes in deceptive text, along with sample words.

Training the model

I employed **MultinomialNB()** in **sklearn** library to train naïve bayes classifier. To use this classifier first we should clean dataset and apply preprocessing step then, after extracting features of dataset I split train and test data in the given dataframe train_test_split() in sklearn library. Before training the model, I used **MinMaxScaler()** in sklearn.preprocessing which is used to transform features by scaling each feature to a given range. Normalization involves adjusting values that exist on different scales into a **common scale**, allowing them to be more readily compared.

Evaluation

Here is the confusion matrix labels which we can map it into the formulas below of accuracy, recall, precision and F1 score.



$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN}$$

$$Recall = \frac{TP}{TP + FN}$$

$$Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN}$$

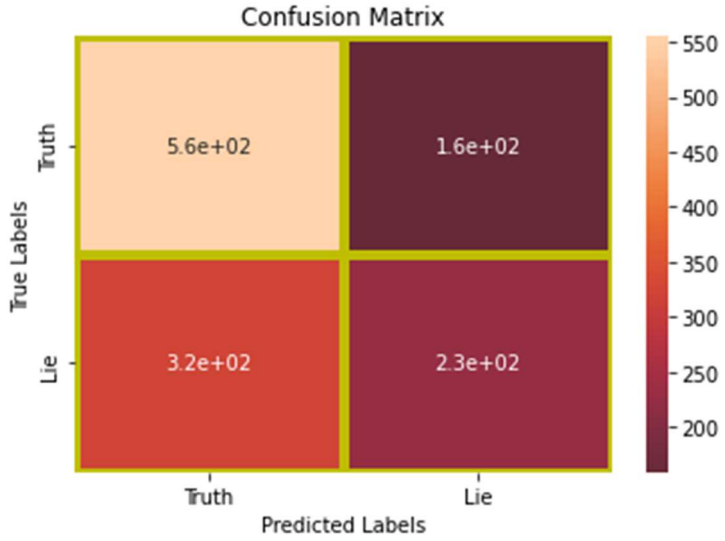
Precision & Recall:

- **Precision**: *Out of all the examples the classifier labeled as positive, what fraction were correct?*
- **Recall:** *Out of all the positive examples there were, what fraction did the classifier pick up?*

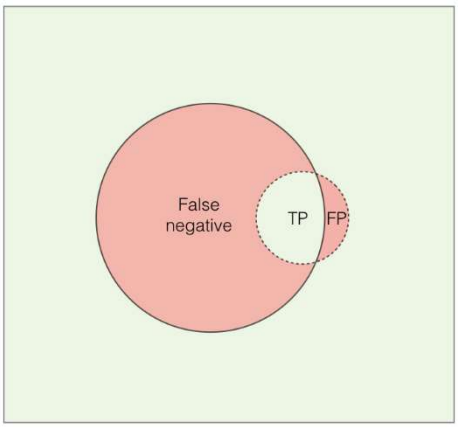
F1 score: F1 Score is needed when you want to seek a balance between Precision and Recall. This is the harmonic mean of the two fractions. F1-measure weights precision and recall equally. This measure is approximately the average of the two when they are close but it's not exactly the mean because it punishes extreme values.

Evaluation on second dataset results:

```
Accuracy: 0.6179952644041041
Recall: 0.41229656419529837
Precision: 0.5891472868217055
F1 score: 0.4851063829787234
```



Here we have "**low recall high precision**". Our classifier is very **picky** and, does not think many things are "truth". All the texts it thinks are "truth", are really "truth". However, it also misses a lot of actual "truth" because, it is so very picky. We have low recall, but a very high precision. Low recall, high precision can be show as:



Recall and precision are not a good metric individually, but F1 score which a harmonic mean of these two metrics is actually a very good metric to evaluate out model.

Here is the classification report of the model:

	precision	recall	f1-score	support
0.0	0.63	0.78	0.70	714
1.0	0.59	0.41	0.49	553
accuracy			0.62	1267
macro avg	0.61	0.59	0.59	1267
weighted avg	0.61	0.62	0.60	1267

Usually, it is better to get high recall and high precision, but it's not always true. Sometimes it's more important to correctly classify your samples than getting all of them (**high precision**). Sometimes it's more important to get all of them (**high recall**) than getting all of them correct. Also, we can see that the confusion matrix shows that the precision, recall and f1-score of truth(0) is higher that lie(1), this could be because there are 1266 more records in truth class than lie class, so the model could learn the truth class better and is biased.

```
train value counts:
 0.0    5751
1.0    4485
Name: label, dtype: int64
```

Conclusion

In this assignment I used Naive Bayes Classifier to classify the spam and ham messages. I got familiar with the model of bag-of-words and using it in naïve bayes model. I got deeper in the naive bayes formula and how it functions and. I employed nltk and skelarn libraries to pre-process and train my model. I got familiar with some popular ways to evaluate my model and assess how it works. Now I can use Naive Bayes Classifier for classifying data when I want to get a fast and not bad results!

Helpful Links which helped me doing this assignment:

- https://www.guru99.com/pos-tagging-chunking-nltk.html
- https://dl.acm.org/doi/pdf/10.5555/1667583.1667679
- https://medium.com/@klintcho/explaining-precision-and-recall-c770eb9c69e9
- https://towardsdatascience.com/is-accuracy-everything-96da9afd540d

*** for running the code make sure you have these files near the two notebook files:

- ✓ Dataset #1: spam.csv
- ✓ Dataset #2: train_final.csv, test_final.csv
- ✓ Spam_words.txt, intensifiers.txt