Text Analysis: Naïve Bayes Algorithm using Python JupyterLab
A. S. Thanuja Nishadi

Faculty of Graduate Studies,
University of Colombo, Sri Lanka
thanuja.nishadi@gmail.com
DOI: 10.29322/IJSRP.9.11.2019.p9515
http://dx.doi.org/10.29322/IJSRP.9.11.2019.p9515

Abstract—The use of mobile phones has skyrocketed during the last decade hence it has generated bulk of junk messages. However, emerging machine learning algorithms are one such solutions which are used to detect and filter spam messages recently. Naïve Bayes is a powerful yet simple classification machine learning algorithm which used in predictive modelling. Thus, Naïve Bayes categorize as four forms of its implementations; however, Multinomial Naïve Bayes is used in this study. Further, it has pre-processed the data using bag – of words (BoW) using tokenizing data with Naïve Bayes algorithm. This study will explore the process of spam filtering using Naïve Bayes classifier and further predict the classification of new text as ham or spam. In addition to that, the data analysis is carried out in Python-Jupyter Lab which is the next-generation open source web-based user interface.

Index Terms—Machine Learning, Analysis of Algorithms, Spam Filtering, Naïve Bayes Algorithm, Multinomial Naïve Bayes, Classification Algorithm, Bag-Of-Words

I. INTRODUCTION

The volumes of Short Messages (SMS) are growing fast due to its features of rapidness, effectiveness and low cost. However, unsolicited or unwanted text messages generate huge threats for both individuals and organizations in numerous ways. The Text Retrieval Conference (TREC) defines the term ‘spam’ as an unsolicited, unwanted messages that was sent indiscriminately (Cormack, 2008). Further, the spams are usually unsolicited and also being a carrier of malware including advertisements, fraud, phishing messages, promotions etc. Many people suffer from spam messages hence its annoyance to individuals and also considered as less reliable sources. Further, spams generate loss of productivity, misuse of data, storage and bandwidth, spread of viruses and ultimately financial loss for the organizations (Wang, 2013).

Naïve Bayes is the simplest probabilistic classifier in machine learning which used in predictive modelling. The predictor variables in the Naïve Bayes are conditionally independent of other features. Further, Naïve Bayes classifier work with correlating with tokens and calculate the probability using Bayes’ Theorem to predict the spam non-spam occurrences. Moreover, it has used bag of words feature which commonly used in text classification to identify spam messages. Although, this algorithm is simple in nature, this often used in more sophisticated activities. This study will explore the process of spam filtering using Naïve Bayes classifier and further predict the classification of new text as ham or spam.

II. MACHINE LEARNING ALGORITHMS

Machine learning technologies are becoming trends in almost every filed of technological activities. Basically, machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed (Leimbach, 1994). Further, the process of learning begins by using collection of data or observation then searching patterns of gathered data and finally aiming for decision making (Edgar & Manz, 2017). Therefore, the aim of the machine learning is to learn things automatically by computers without human intervention.

All the machine learning algorithms are often categorized into four different types including supervised, unsupervised, semi-supervised and reinforcement (Omar, et. al, 2013).

Supervised Learning: This is used in the situations of dataset which consists with a target or outcome variable (dependent variable – DV) which can be predicted from a given set of predictors (independent variables – IVs) (F.Y.O et. al, 2017). Further, the goal of this is to predict the output for the new data once the algorithm identifies the known data. In addition to that, supervised learning algorithms has two further processes such as classification and regression. The most widely supervised learning algorithms are Linear Regression, Logistic Regression, random forest, gradient boosted trees, support vector machine (SVM), neural networks, decision trees, Naïve Bayes and nearest neighbor (Ghorbani, et. al, 2015). In classification algorithm, incoming or new data is labeled based on the past samples of data and algorithms are trained to recognize the certain types of objects then classify accordingly. This is
mainly differentiating the data based on similar features. Furthermore, regression is used to identify the patterns and calculating the predictions based on continuous outcome.

**Unsupervised Learning:** In unsupervised learning, it consists with only input data and corresponding results are unknown. The goal of this model is to detect the underline structure or distribution the data set through learning more about the data (Dy & Brodley, 2004). Further, this has grouped into two segments due to the complexity of the logic including clustering and association (Khanum, 2015). Discovering the inherent groups in the data set is performed in the clustering process. However, association rule is used to describe the large proportions of data in the association process. The most popular examples of unsupervised learning algorithms are K-Means clustering, t-SNE (t-Distributed Stochastic Neighbor Embedding and PCA (Principal Component Analysis and Association Rule (Dy & Brodley, 2004).

**Semi-supervised Learning:** These algorithms represent a middle ground between supervised and unsupervised features. However, some models combine the features of both aspects (Prakash & Nithya, 2014). This is basically assigns the situations where the dataset consists with both labeled and unlabeled data for training.

**Reinforcement Learning:** This interacts with its environment by producing actions and discovers errors or rewards. The decisions are taken sequentially i.e. outcome depends on the state of the current input and the next input depends on the output of the previous input (Mao, et al, 2016).

### III. NAÏVE BAYES CLASSIFIER

Naïve Bayes classifier is a supervised learning technique which is used for classification tasks. Further, the classifier is mainly used in analytical and predictive problems when the dimensionality of the inputs is high. Despite the simplicity, this is often used in more sophisticated classification methods (Dada, et al, 2019).

**Bayes Theorem:** The classifier in Naïve Bayes is based on the Bayes theorem.

\[
P(A|B) = \frac{P(B|A)P(A)}{P(B)}
\]

According to Bayes Theorem, it can be found the probability of A happening, when B has occurred. In this formula, B is the evidence and A is the hypothesis. Further, the presence of one particular feature does not affect to the other hence that is referred as ‘naïve’. In this formula,

- \(P(A|B)\) is the posterior probability of class \((A, \text{target})\) given predictor \((B, \text{attributes})\)
- \(P(A)\) is class prior probability
- \(P(B|A)\) is the likelihood of the probability of predictor in a given class
- \(P(B)\) is the prior probability of predictor

### IV. SAMPLE DATA SET

The dataset is received from https://www.kaggle.com in order to validate Naïve Bayes classification algorithm using Spam/Ham classification from SMS dataset.
V. DATA ANALYSIS

In order to build the Naïve Bayes model, the following stages were carried out. The major stages were identified as data acquisition, data pre-processing, splitting the data as training and testing, model creation and model evaluation.

![Figure 1: Steps of Machine Learning Process](image)

**a) Data Acquisition:** The sample data set (spam.csv) is loaded to Python-Jupiter lab. The dataset contains messages which composed by two columns (v1: label including ham or spam and v2: SMS which contains raw text). Further, the data set is comprised of total 5572 observations of messages. It includes 4825 ham messages and 747 spam messages.

```python
In [3]:
import numpy as np
import pandas as pd
import nltk

In [4]:
import pandas
df_sms = pd.read_csv('C:\Users\LAB-User\Desktop\spam.csv',encoding='latin-1')
df_sms.head()

In [9]:
Number of observations in each label spam and ham
df_sms.label.value_counts()

Out[9]:
ham  4825
spam  747
Name: label, dtype: int64

In [7]:
print (len(df_sms))

5572
```

**b) Data Pre-Processing:** Data pre-processing is the process of cleaning the data before applying the algorithms. The data set includes an additional unnamed column. Therefore, it has dropped the unwanted columns Unnamed:2, Unnamed: 3 and Unnamed:4 using coding. The following code indicates how it removes the additional column and the label with SMS text content of sample raw data using head and tail function. Further, it has used bag-of-words methods in data cleaning process.

```python
In [5]:
df_sms = df_sms.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1)
df_sms = df_sms.rename(columns={'v2':'label', 'v2':'sms'})

In [6]:
df_sms.head()

Out[6]:
<table>
<thead>
<tr>
<th>label</th>
<th>sms</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Go until juryong point, crazy... Available only...</td>
</tr>
<tr>
<td>1</td>
<td>Ok let... Joking wifi u oni...</td>
</tr>
<tr>
<td>2</td>
<td>Free entry in 2 a wkly comp to win FA Cup final...</td>
</tr>
<tr>
<td>3</td>
<td>U dun say so early hor... U c already then say...</td>
</tr>
<tr>
<td>4</td>
<td>Nah I don't think he goes to usf, he lives ino...</td>
</tr>
</tbody>
</table>
```
Further, following descriptive information indicates how the label, message and length of each message.

Implementation of Bag-of-Words (BoW): BoW is a very common feature extraction procedure in Natural Language Processing (Belinkov & Glass, 2019). It is highly required to check whether the text document is best fit as a feature vector prior to apply machine learning algorithms.

All these rows of data converted into tokens or individual words. Vocabulary is the collection of unique words. Further, those tokens were observed and found the frequency of each token. Any order or structure of the words in the document (bag of words) is discarded. However, this is used when the known words are in the document only.

Tokenization: The text corpus is split into individual elements. Further, it has converted all strings into lower cases in this stage.

The above outcome shows how it has converted the messages of text into lower cases. Then, it has removed all the punctuation in the same tokenization stage.
The above code with the outcome shows how it has removed the punctuations in the text. In addition, that, the below code indicate how it used convert sentences into words or tokenization.

\[
TF(t) = \frac{\text{Number of times the word (t) occurs in the text}}{\text{Total number of words in the text}}
\]

\[
IDF(t) = \frac{\text{Total number of documents}}{\text{Number of documents with term t in it}}
\]

In order to acquire good results with TF-IDF, a huge corpus is needed. Then, it performed vector generation i.e. counting the occurrences of tokens and built a sparse matrix.
Evaluation of Bag-of-Words using scikit-learn in JupyterLab

The mapping from textual data to vector is referred as feature extraction. Then, the CountVectorizer() method is used to data cleaning process. Further, the steps of cleaning include converting all data to lower cases and removing all punctuation marks in the data set. Steps followed in CountVectorizer as follows.

- Create an instance of the CountVectorizer class

```
In [19]: from sklearn.feature_extraction.text import CountVectorizer
count_vector = CountVectorizer()
```

- Call the fit () function to find the vocabulary from one or more documents

```
In [20]: count_vector.fit(documents)
count_vector.get_feature_names()
```

```
In [20]: count_vector.fit(documents)
count_vector.get_feature_names()
```

```
Out[20]: ['are',
          'call',
          'from',
          'hello',
          'home',
          'how',
          'me',
          'money',
          'now',
          'tomorrow',
          'win',
          'you']
```

- Call the transform () function on one or more documents as it needed to encoded each as a vector

```
In [21]: doc_array = count_vector.transform(documents).toarray()
doc_array
```

```
Out[21]: array([[1, 0, 1, 0, 1, 0, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 1, 0, 0, 0, 0],
            [0, 1, 0, 0, 0, 0, 1, 0, 0, 0]],
            dtype=int64)
```

```
In [22]: frequency_matrix = pd.DataFrame(doc_array, columns = count_vector.get_feature_names())
frequency_matrix
```

```
Out[22]:    are  call  from  hello  home  how  me  money  now  tomorrow  win  you
   0  1  0  0  0  0  0  0  0  0  0  0  0  0
   1  0  1  0  0  0  0  0  0  0  0  1  0
   2  0  0  0  0  0  1  0  0  0  0  0  0
   3  0  1  0  0  0  0  0  0  1  0  0  1
```

c) Splitting Dataset in Training and Testing Sets

The data set is split into training and testing test (X_train is for training data of ‘sms_message’ column, y_train is for training data of ‘label’ column, x_test is for testing data for ‘sms_message’ colomn and y_test is our testing data for the 'label' column).
Then, the data set needed to convert into matrix using CountVectorizer(). Further, it is required to fit the training data (x_train) into CountVectorizer() and receive the matrix. Furthermore, transforming testing data (x_test) to return the matrix.

In [23]:
from sklearn.cross_validation import train_test_split

X_train, X_test, y_train, y_test = train_test_split(df_sms['sms'],
    df_sms['label'], test_size=0.20,
    random_state=1)

C:\ProgramData\Anaconda3\lib\site-packages\sklearn\cross_validation.py:41: DeprecationWarning: This module was deprecated in version 0.18 in favor of the model_selection module into which all the refactored classes and functions are moved. Also note that the interface of the new CV iterators are different from that of this module. This module will be removed in 0.20.
    "This module will be removed in 0.20.", DeprecationWarning)

Then, the data set needed to convert into matrix using CountVectorizer(). Further, it is required to fit the training data (x_train) into CountVectorizer() and receive the matrix. Furthermore, transforming testing data (x_test) to return the matrix.

d) Model Creation: Multinomial Naive Bayes Implementation

Naive Bayes consists with four forms of its implementations such as Gaussian, Multinomial, Complement and Bernoulli Naive Bayes (Rennie, et. al, 2003). The Multinomial Naive Bayes has been used in this study due to the classification of discrete features such as word counts. It takes integer word counts as inputs. However, Gaussian is used for continuous data as inputs whereas complement classifier estimates parameters of a category. But, Bernoulli assumes the distribution of probability as Bernullian.

Therefore, it has imported MulinomialNB Classifier in training data set using fit ().

In [26]:
    # Instantiate the CountVectorizer method
count_vector = CountVectorizer()

    # Fit the training data and then return the matrix
training_data = count_vector.fit_transform(X_train)

    # Transform testing data and return the matrix.
testing_data = count_vector.transform(X_test)

Out[26]: MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True)

According to the code snippets, it can clearly be seen that the algorithm has been trained using the training data set. Further, predictions on the test data can be processed using predict ()

In [27]: predictions = naive_bayes.predict(testing_data)

e) Model Evaluation

Predictions are done using test data. Further, accuracy of the predictions is checked in this stage.

Accuracy score indicates how often the classifier makes the correct predictions. Further, the ratio of correct predictions to the total predictions. Precision Score indicates the proportion of messages classified as spam were true spam or true positives

irrespectively correct classification. Recall Score is the proportion of messages which actually indicated as spams were classified as spams.

With considering the above output data, accuracy score of data set is 98%. Further, it has indicated as 2 messages as spams and other 98 of messages as non-spams in a sample of 100 messages. It has further used F1 score to evaluate the weighted average of the precision and recall scores. This value mainly ranges between 0 and 1. As this model receives all four matrices close to 1 this fits with the model best.

VI. CONCLUSION

The study covers how it detected the spam and non-spam messages using Naïve Bayes algorithm. This algorithm allows to handle large number of features i.e. words. Thus, Naïve Bayes support for handling large number words easily.

Further, the data set followed set of processes of machine learning in order to build the model. The data set acquired from the kaggle site and performed pre-processing using many methods including bag-of-words. Then, split the data into train and test model. Furthermore, it has built the model and evaluated. The accuracy of the model is tested using accuracy score, precision score, recall score and F1 score.

REFERENCES


