

19aie205-python for machine learning

**Topic: 3C Context Citation Classification**



**December 8, 2020**

ADITHYAN SUKUMAR (19004)

C.R.ASSWIN(19016)

SREE SAMHITHA K(19034)

VENKAT VIKASH V(19066)

MITHRA K(19070)

**TABLE OF CONTENTS**

1.INTRODUCTION…………………………………………………………………………………..2

2. TEXT PREPROCESSING……………………………………………………………………….3

3.CLASSIFIERS/ALGORITHM………………………………………………………………….5

4.RESULT ANALYSIS…………………………………………………………………………….19

5.CONCLUSION……………………………………………………………………………………..20

6.BIBLIOGRAPHY……………………………………………………………………………..…..21

# **Introduction:**

**What is NLP?**

Natural Language Processing or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.

**Applications:**

1.Disease Prediction

2.Sentiment Analysis

3.Spam Classifiers

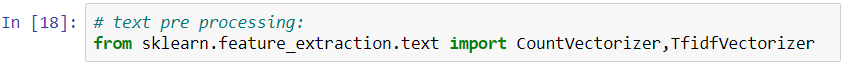
4.Voice driven interfaces like google assistant, Alexa, Siri etc

# **TEXT PRE-PROCESSING:**

A preliminary processing of data in order to prepare it for the primary processing or for further analysis is called Pre-processing. The term can be applied to any first or preparatory processing stage when there are several steps required to prepare data for the user. Removal of useless data is also a form of Pre-Processing. The Pre-Processing required for Context Classification include Removal of Punctuation marks and Removal of Stop words.

Convert a collection of raw documents to a matrix of TF-IDF features.

Equivalent to **CountVectorizer** followed by **TfidfTransformer**.

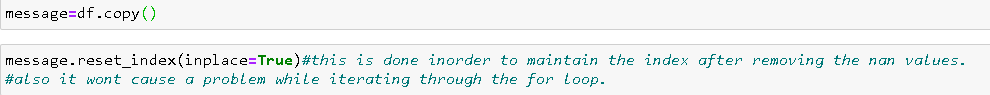


CountVectorizer can lowercase letters, disregard punctuation and stop\_words, but it can't LEMMATIZE or STEM

* **Stop\_words**: Since CountVectorizer just counts the occurrences of each word in its vocabulary, extremely common words like ‘the’, ‘and’, etc. will become very important features while they add little meaning to the text. Your model can often be improved if you don’t take those words into account. Stop words are just a list of words you don’t want to use as features. Alternatively you can set stop\_words equal to some custom list. This parameter defaults to None.
* **Stem words**:  
  There are a lot of words which are different but mat contain similar or same meaning. So they have to be removed from our data. Stemming is the process of producing morphological variants of a root/base word. Stemming programs are commonly referred to as stemming algorithms or stemmers.



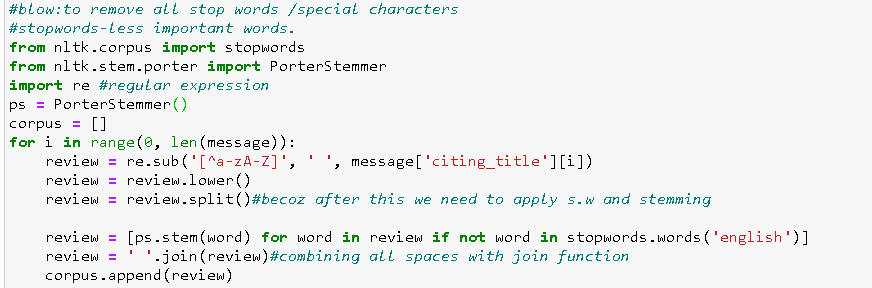
* Initially after the data is read and converted into a dataframe it has to undergo pre-processing. The feature value which is empty has to be removed. The above command drops the whole column of any one element is missing.



* After the NaN values are removed the whole data is copied to another dataframe and the index is reset.

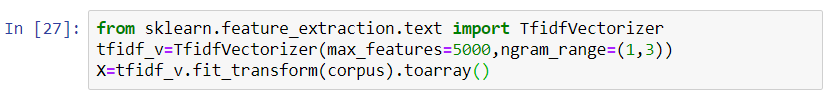


* Downloading all the stopwords necessary for data which is in English.



* Stop\_words and PorterStemmer (Used for stemming) and regular expressions are imported. The re.sub() method replaces or substitutes all occurrences of the RE pattern(like punctuations ,special characters) in string with replace, substituting all occurrences unless max provided. This method returns modified string. In this case all the regular expressions are replaced with ‘ ‘except a-z and A-Z.
* The whole data is converted into lower or upper case and then its split to perform stemming.
* After the words are stemmed, the empty space which is left out is appended.

The goal of using tf-idf is to scale down the impact of tokens that occur very frequently in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training corpus.



**ngram\_range**: An n-gram is just a string of n words in a row. Set the parameter ngram\_range=(a,b) where a is the minimum and b is the maximum size of ngrams you want to include in your features. The default ngram\_range is (1,1).

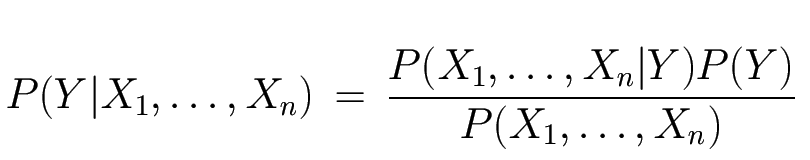
**max\_features:**This parameter is pretty self-explanatory. The TfidfTransformerwill choose the words or features that occur most frequently to be in its’ vocabulary and drop everything else.

# **CLASSIFIERS/algorithms:**

* **Naïve bayes:**

Naive Bayes is based on Bayes theorem, where the adjective Naïve says that features in the dataset are mutually independent. Occurrence of one feature does not affect the probability of occurrence of the other feature. For small sample sizes, Naïve Bayes can outperform the most powerful alternatives. Being relatively robust, easy to implement, fast, and accurate, it is used in many different fields.

According to a dataset, bayes theorem can be



where, Y is class variable and X1,X2,X3,….Xn are a dependent feature vector (of size n) where

probability of event after evidence is seen is considered as **posterior probability**.

We have calculated the probabilities of discrete, individual words and not the probability of something continuous, like weight or height, which is the probability of predictor given class they are also called **livelihoods**.

Initial guess that we observe from a training data is called a **Prior Probability**. This guess can be any probability we want from 0 to 1, but a common guess is estimated from a training data.

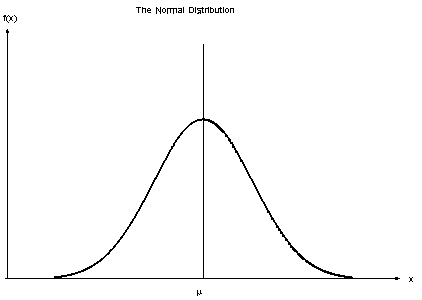
Firstly, the features are extracted and they are set to training data and then they are classified based on naïve bayes, from this classifier trained model is extracted and then samples are predicted from this, these samples are also extracted from identifications chosen from extracted features previously and Using Naive Bayesian equation is calculated the posterior probability for each class. The class with the highest posterior probability is the outcome of prediction.

Example: Players will play if weather is sunny. Is this statement is correct?

* **Gaussian Naïve Bayes**

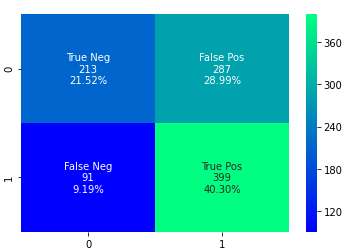
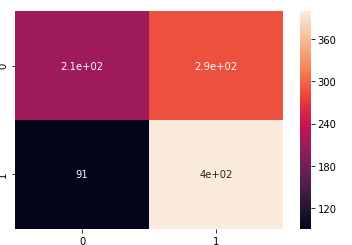
The common version of naïve bayes is gaussian naïve bayes classification. Gaussian naïve bayes is named after the gaussian distributions that represent the data in the training set.

In Gaussian Naïve Bayes, continuous values associated with each feature are assumed to be distributed according to a Gaussian distribution. A Gaussian distribution is also called Normal distribution. When plotted, it gives a bell-shaped curve which is symmetric about the mean of the feature values as shown below:



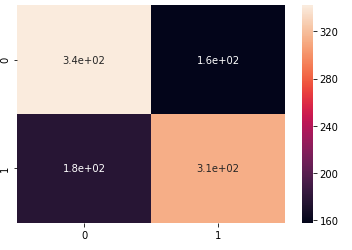
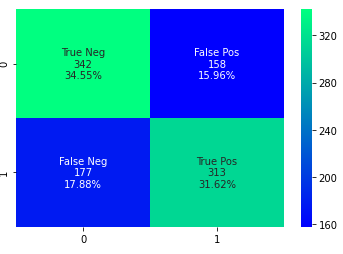
The likelihood of the features is assumed to be Gaussian; hence, conditional probability is given by:

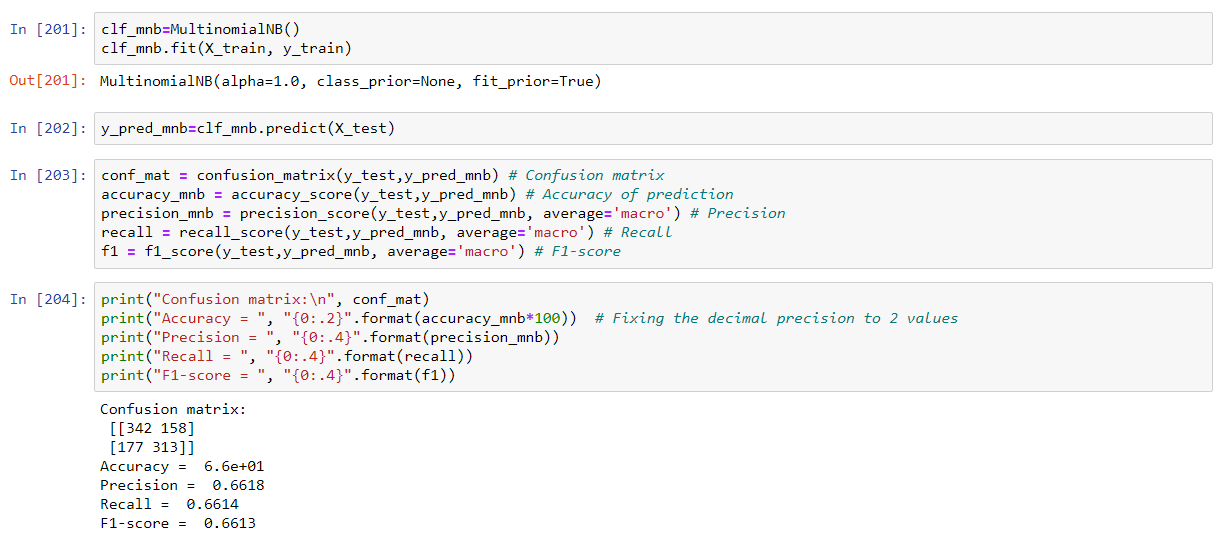


Where sigma is standard deviation and u is the mean of given set of data

**Naive Bayes classifier for Multinomial models**

The multinomial Naive Bayes classifier is suitable for classification with discrete features (e.g., word counts for text classification). The multinomial distribution normally requires integer feature counts. However, in practice, fractional counts such as tf-idf may also work.





which can be expressed as:



Now, as the denominator remains constant for a given input, we can remove that term



Now, we need to create a classifier model. For this, we find the probability of given set of inputs for all possible values of the class variable y and pick up the output with maximum probability. This can be expressed mathematically as:



# **Decision tree**

Initially, the whole training set is considered as the root**.** Feature values are preferred to be categorical. If the values are continuous then they are discretized prior to building the model. Records are distributedrecursively on the basis of attribute values. Order to placing attributes as root or internal node of the tree is done by using some statistical approach

ID3 prefers simpler decision trees

Initially we select and remove *A* from *Attributes* then make *A* the root of the current tree for each value *V* of *A* we should create a branch of the current tree labeled by *V* and *Partition\_V* ¬ Elements of *Training-set* with value *V* for *A* Then induce Tree with *Partition\_V* and *Attributes* and finally attach result to branch *V*

* **Entropy**

Entropy is the measure of randomness. when the entropy is high it is harder to get the conclusion from that information. for example when we are tossing a coin the entropy will be zero when the probability is either 0 or 1. The Entropy is maximum when the probability is 0.5 because there is a equal chance for the both occurrences and there is no chance in perfectly determining the outcome.



where *p( j | t)* is the relative frequency of class j at node t

##### **Information gain:**

Information gain is a statistical property. It measures how well a given attribute separates the training examples according to their target classification. Information gain is a decrease in entropy. Making a decision tree is all about finding an attribute which returns the highest information gain and the smallest entropy. It computes the difference between entropy before split and average entropy after split of the dataset based on given attribute values



##### **where parent Node, p is split into k partitions, and ni is number of records in partition i**

##### Gini index

Gini Index calculates the amount of probability of a specific feature that is classified incorrectly when selected randomly. If all the elements are linked with a single class then it can be called pure. The value of Gini index varies between values 0 and 1. Where 0 expresses the purity of classification, i.e. All the elements belong to a specified class or only one class exists there. And 1 indicates the random distribution of elements across various classes. The value of 0.5 of the Gini Index shows an equal distribution of elements over some classes.





##### Gain ratio

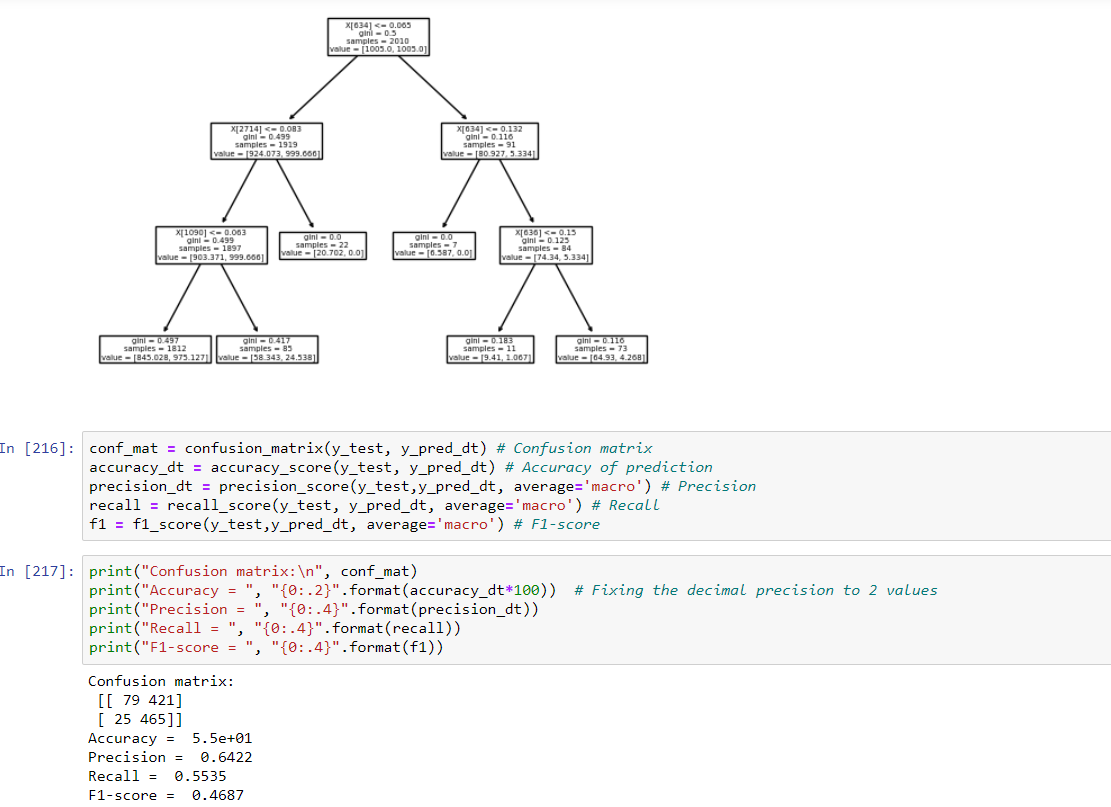
Information gain is biased towards choosing attributes with a large number of values as root nodes. It means it prefers the attribute with a large number of distinct values.

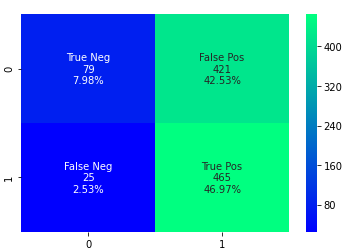
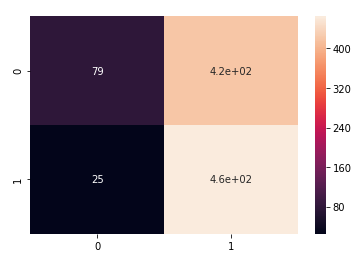
 Gain ratio overcomes the problem with information gain by taking into account the number of branches that would result before making the split. It corrects information gain by taking the intrinsic information of a split into account.









****

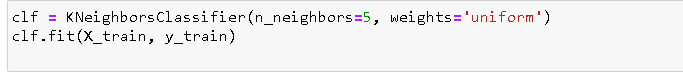
# **K-NEAREST NEIGHBOUR**

KNN is a machine learning algorithm which is mainly used for classification and regression. It belongs to the category of supervised learning. KNN algorithm assumes the similarity between the new case/data and available cases and put the new case into the category that is most similar to the available categories. K-NN algorithm stores all the available data and classifies a new data point based on the similarity. This means when new data appears then it can be easily classified into a well suite category by using K- NN algorithm.

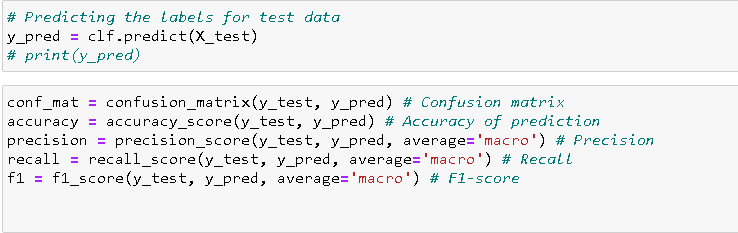
K-NN is a non-parametric algorithm, which means it does not make any assumption on underlying data.

It is also called a lazy learner algorithm because it does not learn from the training set immediately instead it stores the dataset and at the time of classification, it performs an action on the dataset.

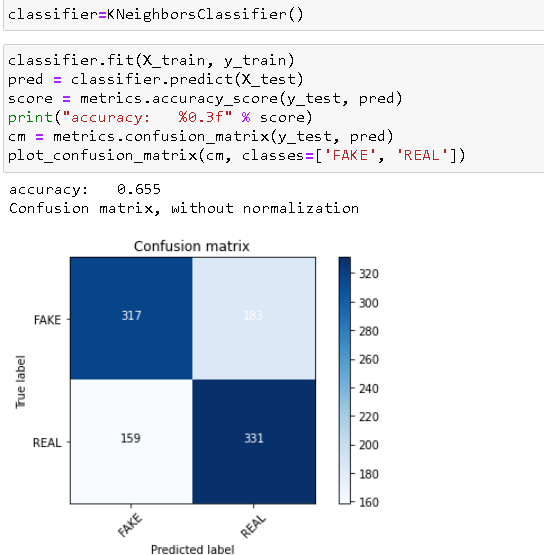
KNN algorithm at the training phase just stores the dataset and when it gets new data, then it classifies that data into a category that is much similar to the new data.

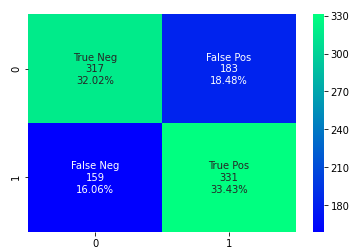
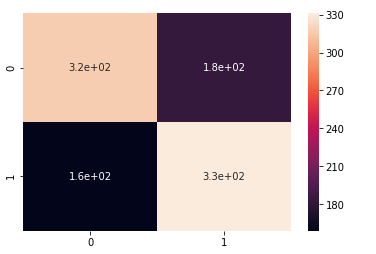


* Using Kneighbours Classifier function to specify the number of neighbours and mentioning that all points in the neighbourhood are weighted equally



* Predicting the labels for test data and finding confusion matrix,accuracy, precision, recall and f1 score.



* Plotting the confusion matrix.   
  

# **RANDOM FOREST:**

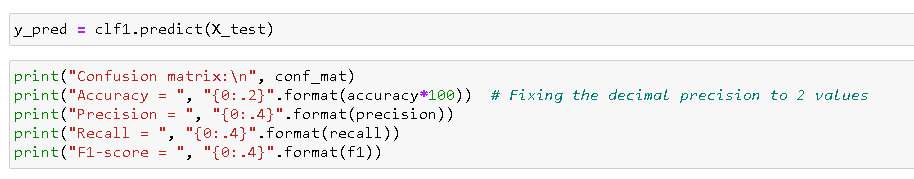
Random Forest algorithm is a supervised classification algorithm. There is a direct relationship between the number of trees in the forest and the results it can get. Larger the number of trees, the more accurate the result. It creates a forest by some way and makes it random. The difference between Random Forest algorithm and the decision tree algorithm is that in Random Forest, the processes of finding the root node and splitting the feature nodes will run randomly.

The basic idea behind this is to combine multiple decision trees in determining the final output rather than relying on individual decision trees.

Random Forest has multiple decision trees as base learning models. We randomly perform row sampling and feature sampling from the dataset forming sample datasets for every model. This process is called as Bootstrap.



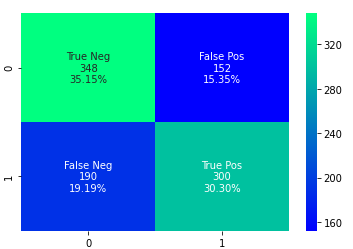
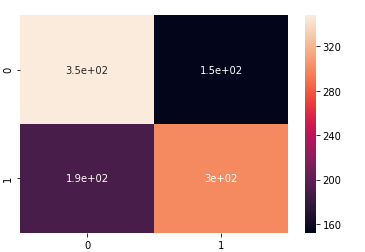
* Clf1 is a variable used to store the data. The RandomForestClassifier function is called the and the fit() method fits the training set as features and label in to the Random Forest classifier model.



* The output of the model is predicted and Confusion matrix, accuracy , Precision, recall and F1-score are found and printed.



* Plotting the confusion matrix.



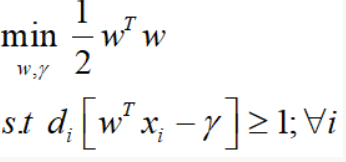
# **SVM:**

Support Vector Machines or (SVM) is a Supervised model used for classification and regression which can be done by finding the hyperplane in a N-dimensional space where N is the number of features which is used to classify data.

* **Hard Margin:**

If the training data is linearly separable then we can select two parallel hyperplanes that separate the two classes of data, so that the distance between them is as large as possible. The region bounded by these two hyperplanes is called the “margin”, and the maximum-margin hyperplane is the hyperplane that lies halfway between them.

The condition is to have the hyperplane at least 1unit away from both the sides. We can write the problem formulations as mentioned below.

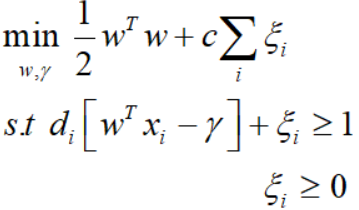


* **Soft Margin:**

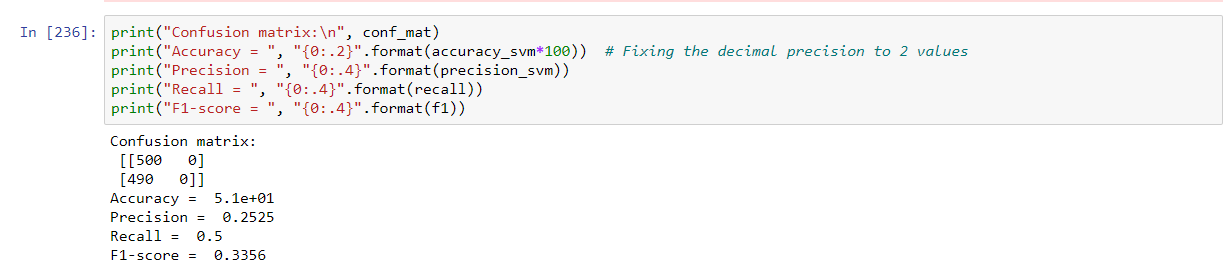
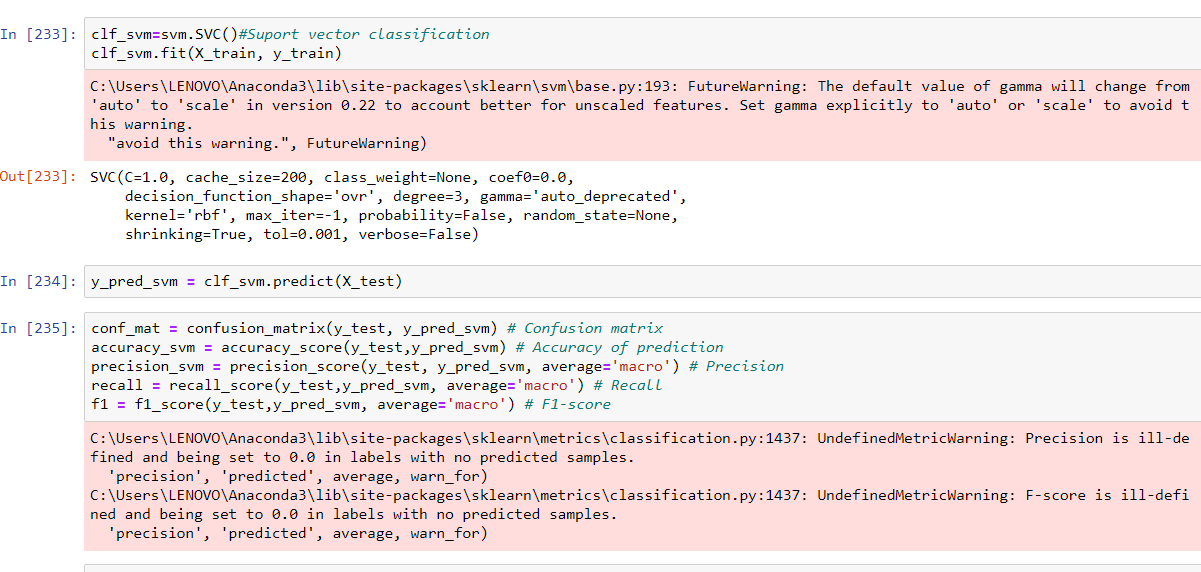
If the training data is not linearly separable then we cannot simply select two parallel hyperplanes because it might lead to wrong predictions in some case. So Soft Margin SVM is used to classify these type of datas.

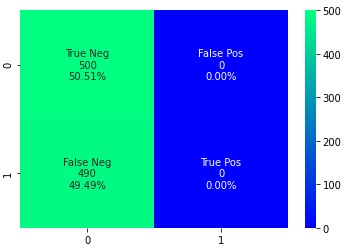
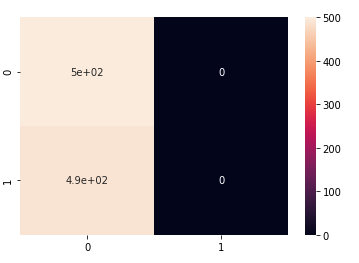
Since there is an overlap between the data points, we introduce a slack variable into the formulation to tally the problem.

The formulation is given below.



Here the psi is the slack variable used to tally the overlapping problem and c is the control parameter used to prevent the misclassification of data to one side. It can be solved using python as shown below.





* **Accuracy**

Accuracy is one metric for evaluating classification models. Informally, **accuracy** is the fraction of predictions our model got right. Formally, accuracy has the following definition:



For binary classification, accuracy can also be calculated in terms of positives and negatives as follows:



Where TP = True Positives, TN = True Negatives, FP = False Positives, and FN = False Negatives.

* **Precision:**

proportion of positive identifications was actually correct



* **Recall:**

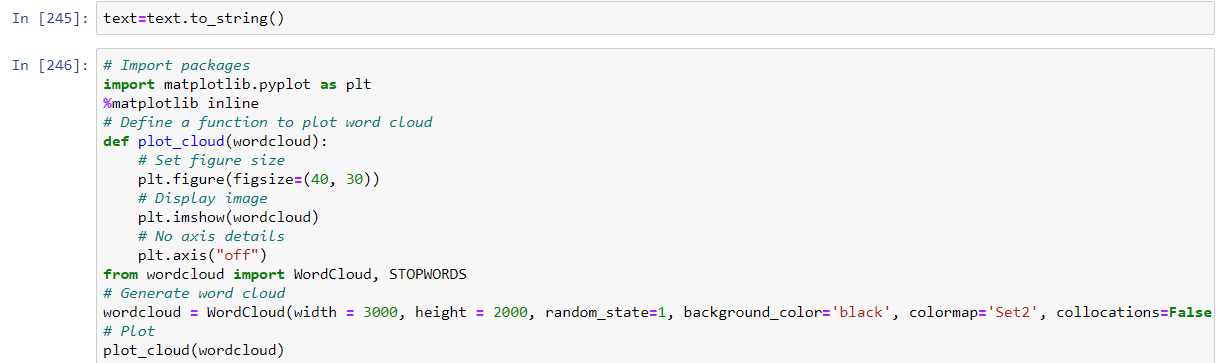
proportion of actual positives was identified correctly.

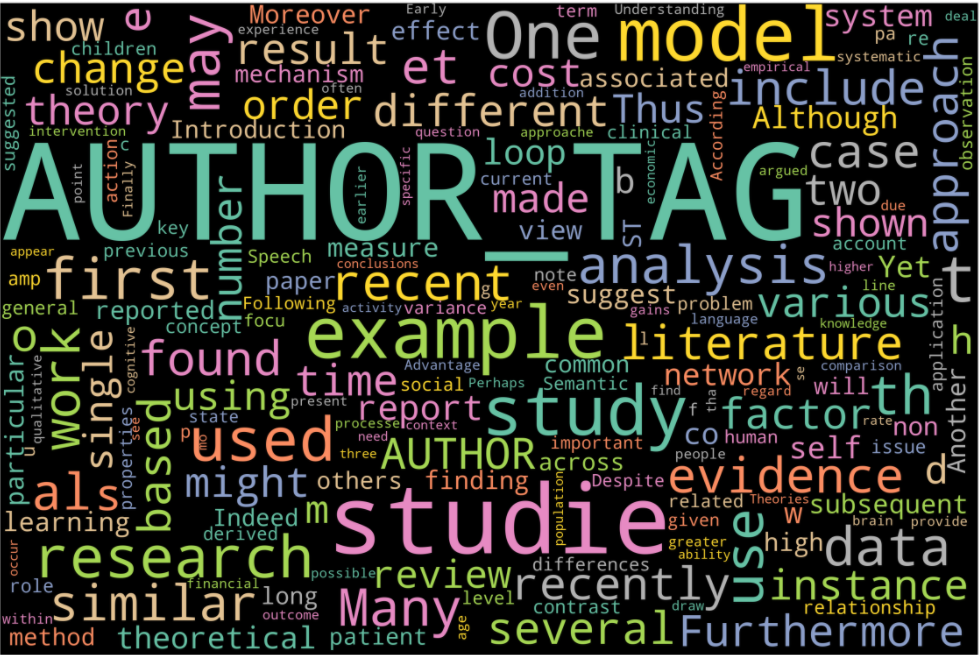


* **F1-score:**

F1 Score. The F1 Score is the 2\*((precision\*recall) / (precision + recall)). It is also called the F Score or the F Measure. Put another way, the F1 score conveys the balance between the precision and the recall.

# **Using Word Cloud to represent the citation context**

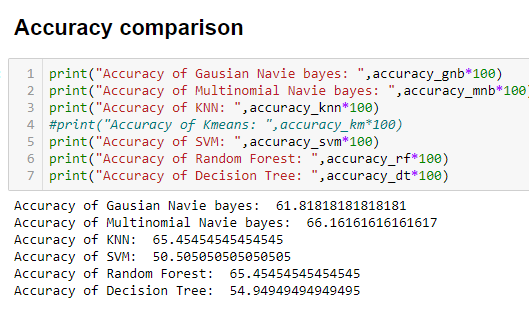
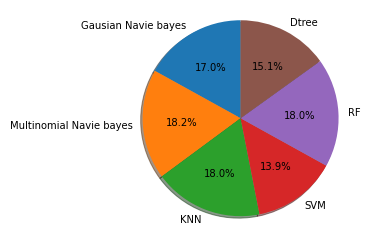
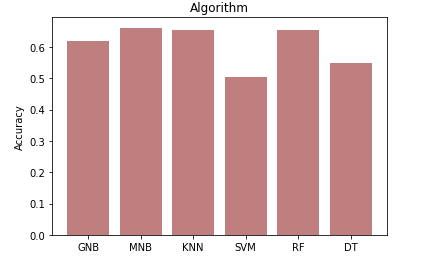




# **Result Analysis**

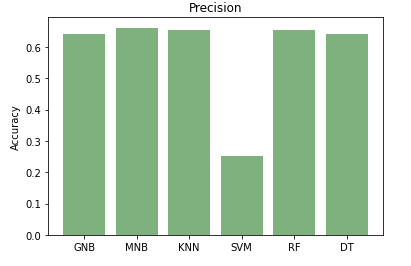
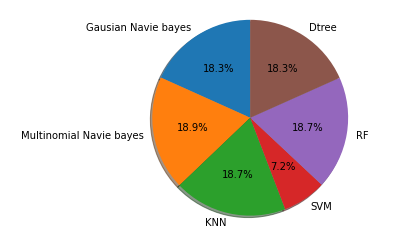
* **Accuracy Comparison:**

**Bar Chart:**  **Pie Chart:**



* **Precision Comparison:**

**Bar Chart:**  **Pie Chart:**



* So from the result analysis we can conclude that Multinomial naive Bayes Algorithm works best for this project(context classification) followed by K-Nearest Neighbour and Random Forest. The least performing algorithms are Decision Tree and Support Vector Machines.

# **Bibliography:**

https://scikit-learn.org/stable/

<https://scikit-learn.org/stable/supervised_learning.html#supervised-learning>

<https://scikit-learn.org/stable/modules/clustering.html#clustering>

<https://scikit-learn.org/stable/supervised_learning.html#supervised-learning>