Table of Contents

1. Abstract

Sentiment Analysis also known as Opinion Mining refers to the use of natural language processing, text analysis to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine.

In this project, we aim to perform Sentiment Analysis of reviews. Data used in this project are online product reviews collected from “amazon.com”. We expect to do review-level categorization of review data with promising outcomes.

2. Introduction

Sentiment is an attitude, thought, or judgment prompted by feeling. Sentiment analysis, which is also known as opinion mining, studies people’s sentiments towards certain entities. From a user’s perspective, people are able to post their own content through various social media, such as forums, micro-blogs, or online social networking sites. From a researcher’s perspective, many social media sites release their application programming interfaces (APIs), prompting data collection and analysis by researchers and developers. However, those types of online data have several flaws that potentially hinder the process of sentiment analysis. The first flaw is that since people can freely post their own content, the quality of their opinions cannot be guaranteed. he second flaw is that ground truth of such online data is not always available. A ground truth is more like a tag of a certain opinion, indicating whether the opinion is positive, negative, or neutral.

“It is a quite boring movie…… but the scenes were good enough”.

The given line is a movie review that states that “it” (the movie) is quite boring but the scenes were good. Understanding such sentiments require multiple tasks.

Hence, SENTIMENT ANALYSIS is a kind of text classification based on *Sentimental Orientation* (SO) of opinion they contain.

Sentiment analysis of product reviews has recently become very popular in text mining and computational linguistics research.

- Firstly, evaluative terms expressing opinions must be extracted from the review.
- Secondly, the SO, or the polarity, of the opinions must be determined.
- Thirdly, the opinion strength, or the intensity, of an opinion should also be determined.
- Finally, the review is classified with respect to sentiment classes, such as Positive and Negative, based on the SO of the opinions it contains.

3. Review of Literature

The most fundamental problem in sentiment analysis is the sentiment polarity categorization, by considering a dataset containing over 50 thousand reviews.

A max-entropy POS tagger is used in order to classify the words of the sentence, an additional python program to speed up the process. The negation words like no, not, and more are included in the adverbs whereas Negation of Adjective and Negation of Verb are specially used to identify the phrases.

For splitting the Data we used Train Test Split from Sklearn:

```
Split arrays or matrices into random train and test subsets

Quick utility that wraps input validation and

``next(ShuffleSplit().split(X, y))`` and application to input data

into a single call for splitting (and optionally subsampling) data in a oneliner.
```

For Accuracy and Test Reports we used:

Accuracy Score:

In multilabel classification, this function computes subset accuracy:

the set of labels predicted for a sample must *exactly* match the

corresponding set of labels in y_true.

Confusion Matrix:

Compute confusion matrix to evaluate the accuracy of a classification.

By definition a confusion matrix :math:`C` is such that :math:`C_{i, j}`

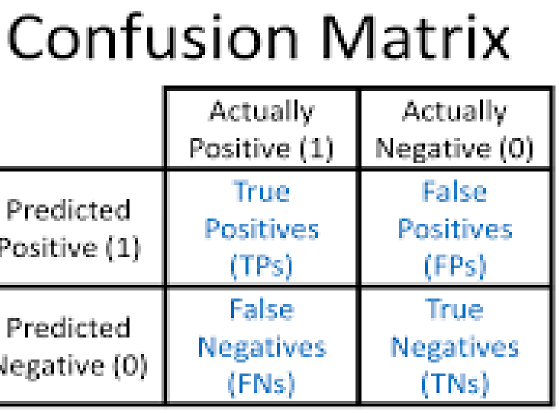
is equal to the number of observations known to be in group :math:`i` and

predicted to be in group :math:`j`.

Thus in binary classification, the count of true negatives is

:math:`C_{0,0}`, false negatives is :math:`C_{1,0}`, true positives is

:math:`C_{1,1}` and false positives is :math:`C_{0,1}`.



Confusion Matrix

	Actually Positive (1)	Actually Negative (0)
Predicted Positive (1)	True Positives (TPs)	False Positives (FPs)
Predicted Negative (0)	False Negatives (FNs)	True Negatives (TNs)

Classification Report Time:

A Classification report is used to measure the quality of predictions from a classification algorithm. How many predictions are True and how many are False. More specifically, True Positives, False Positives, True negatives and False Negatives are used to predict the metrics of a classification report as shown below.

Ex-

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        50
Iris-versicolor       0.77      0.96      0.86        50
 Iris-virginica       0.95      0.72      0.82        50

    avg / total       0.91      0.89      0.89       150
```

4. Objective of the Project

- Analyze and categorize review data.
- Analyze sentiment on data set from document level (review level).
- Categorization or classification of opinion sentiment into-
  - Positive
  - Negative

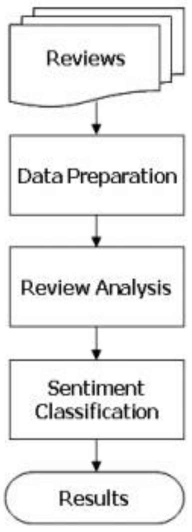


Figure 1: A typical sentiment analysis model.

Need of Sentiment Analysis:

Industry Evolution:

Only the useful amount of data is required in the industry as compared to the set of complete unstructured form of the data. However the sentiment analysis done is useful for extracting the important feature from the data that will be needed solely for the purpose of industry. Sentimental Analysis will provide a great opportunity to the industries for providing value to their gain value and audience for themselves. Any of the industries with the business to consumer will get benefit from this whether it is restaurants, entertainment, hospitality, mobile customer, retail or being travel.

Decision Making:

Every person who stores information on the blogs, various web applications and the web social media, social websites for getting the relevant information you need a particular method that can be used to analyze data and consequently return some of the useful results. It is going to be very difficult for company to conduct the survey that will be on the regular basis so that there comes the need to analyze the data and locate the best of the products that will be based on user's opinions, reviews and advices. The reviews and the opinions also help the people to take important decisions helping them. research and business areas.

Internet Marketing:

Another important reason behind the increase in the demand of sentimental analysis is the marketing done via internet by the business and companies organization. Now they regularly monitor the opinion of the user about their brand, product, or event on blog or the social post. Thus, we see that the sentimental Analysis could also work as a tool for marketing too.

Applications of Sentiment Analysis:

Sentiment analysis has large amount of applications in the NLP domain. Due to the increase in the sentiment analysis, social network data is on high demand. Many companies have already adopted the sentimental analysis for the process of betterment. Some of major applications are mentioned as following:

Online Commerce:

from various areas like service and quality details of the users of company users experience about features, product and any suggestions. These details and reviews have been collected by company and conversion of data into the geographical form with the updates of the recent online commerce websites who use these current techniques.

Voice of the Market (VOM):

Whenever a product is to be launched by a specific company, the customers would to know about the product ratings, reviews and detailed descriptions about it. Sentiment Analysis can help in analyzing marketing advertising and for making new strategies for promoting the product. It provides the customer an opportunity to choose the best. among the all.

5. System Design

Hardware Requirements:

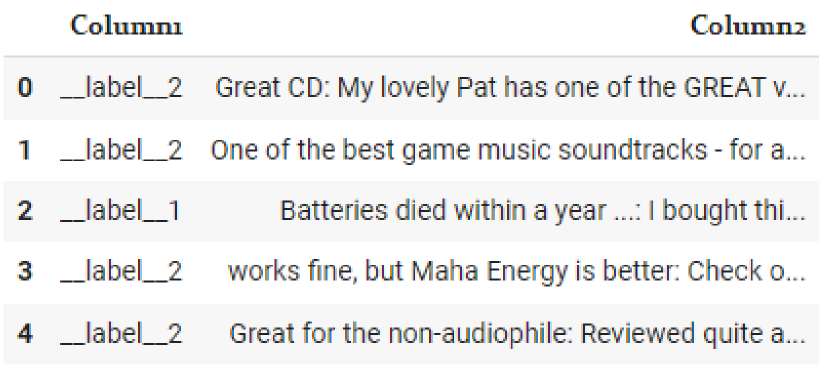
- Core i3/i5 processor
- At least 8 GB RAM
- At least 60 GB of Usable Hard Disk Space

Software Requirements:

- Python
- Anaconda Distribution
- NLTK Toolkit
- Window 10,Operating System.

Data Information

- The Amazon reviews dataset consists of reviews from amazon. Reviews include label and a plaintext review.
- The Amazon reviews full score dataset is constructed by randomly taking 28,000 samples. In total there are 1,000,000 samples.
- The labels are marked as :
  - "__label__1 " for Negative Review
  - "__label__2 " for Positive Review



	Column1	Column2
0	__label__2	Great CD: My lovely Pat has one of the GREAT v...
1	__label__2	One of the best game music soundtracks - for a...
2	__label__1	Batteries died within a year ...: I bought thi...
3	__label__2	works fine, but Maha Energy is better: Check o...
4	__label__2	Great for the non-audiophile: Reviewed quite a...

6. Methodology for Implementation (Formulation/Algorithm)

Data Collection:

Used Dataset contains product reviews with their labels and the Dataset is a .csv downloaded from

https://www.kaggle.com/bittlingmayer/amazonreviews?select=test.ft.txt.bz2

Sentiment Sentence Extraction & Pos Tagging:

Tokenization of reviews after removal of STOP words which mean nothing related to sentiment is the basic requirement for POS tagging. After proper removal of STOP words like “am, is, are, the, but” and so on the remaining sentences are converted in tokens. These tokens take part in POS tagging .

In natural language processing, part-of-speech (POS) taggers have been developed to classify words based on their parts of speech. For sentiment analysis, a POS tagger is very useful because of the following two reasons: 1) Words like nouns and pronouns usually do not contain any sentiment. It is able to filter out such words with the help of a POS tagger; 2) A POS tagger can also be used to distinguish words that can be used in different parts of speech.

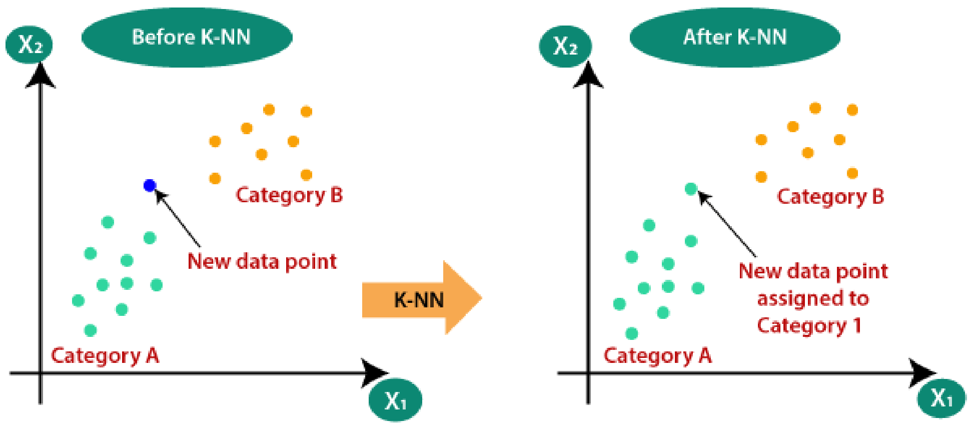
Negetive Phrase Identification:

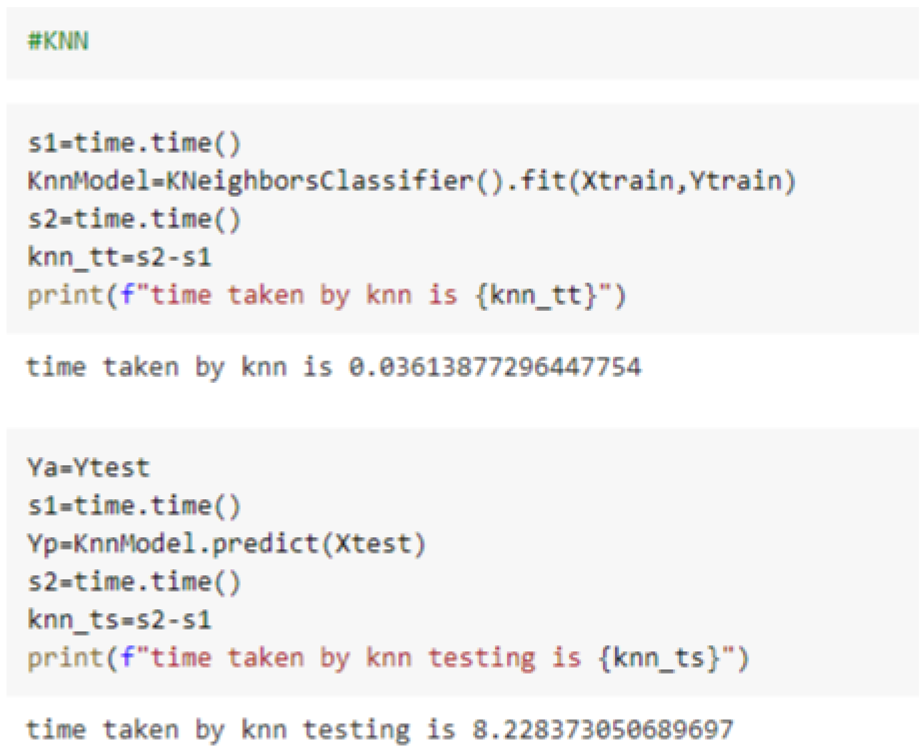
Words such as adjectives and verbs are able to convey opposite sentiment with the help of negative prefixes. For instance, consider the following sentence that was found in an electronic device’s review: “The built in speaker also has its uses but so far nothing revolutionary." The word, “revolutionary" is a positive word according to the list in. However, the phrase “nothing revolutionary" gives more or less negative feelings. Therefore, it is crucial to identify such phrases. In this work, there are two types of phrases have been identified, namely negation-of-adjective (NOA) and negation-of-verb (NOV).

Sentiment Classification Algorithms:

K-Nearest Neighbours Classification:

- K-Nearest Neighbour is one of the simplest Machine Learning algorithms based on Supervised Learning technique.
- K-NN 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 algorithm can be used for Regression as well as for Classification but mostly it is used for the Classification problems.
- 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.





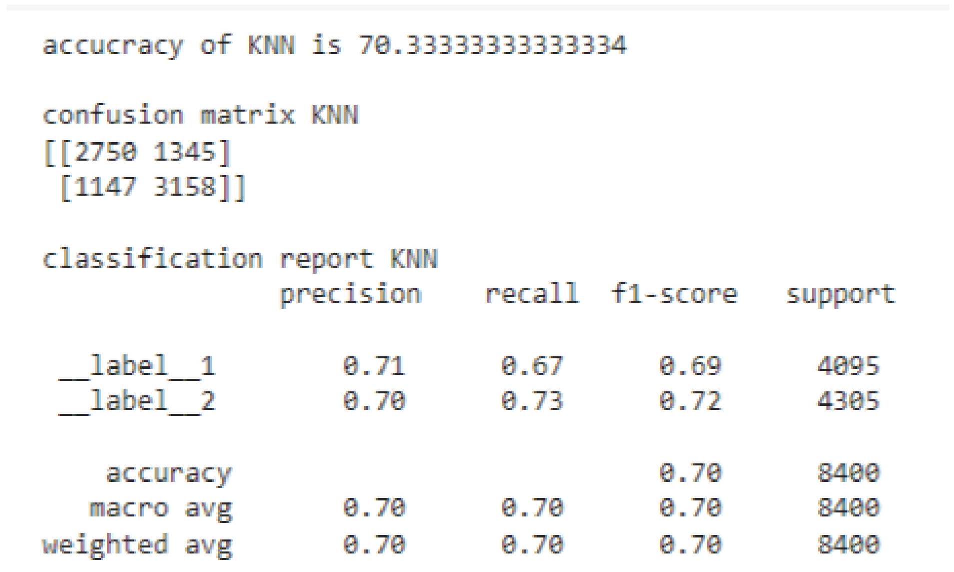
```
#KNN
```

```
s1=time.time()
KnnModel=KNeighborsClassifier().fit(Xtrain,Ytrain)
s2=time.time()
knn_tt=s2-s1
print(f"time taken by knn is {knn_tt}")
```

```
time taken by knn is 0.03613877296447754
```

```
Ya=Ytest
s1=time.time()
Yp=KnnModel.predict(Xtest)
s2=time.time()
knn_ts=s2-s1
print(f"time taken by knn testing is {knn_ts}")
```

```
time taken by knn testing is 8.228373050689697
```



```
accucracy of KNN is 70.33333333333334

confusion matrix KNN
[[2750 1345]
 [1147 3158]]

classification report KNN
              precision    recall  f1-score   support

   __label__1       0.71      0.67      0.69      4095
   __label__2       0.70      0.73      0.72      4305

     accuracy                           0.70      8400
    macro avg       0.70      0.70      0.70      8400
 weighted avg       0.70      0.70      0.70      8400
```

Logistic Regression :

Logistic regression predicts the probability of an outcome that can only have two values (i.e. a dichotomy). The prediction is based on the use of one or several predictors (numerical and categorical). A linear regression is not appropriate for predicting the value of a binary variable for two reasons:

- A linear regression will predict values outside the acceptable range (e.g. predicting probabilities outside the range 0 to 1)
- Since the dichotomous experiments can only have one of two possible values for each experiment, the residuals will not be normally distributed about the predicted line.

On the other hand, a logistic regression produces a logistic curve, which is limited to values between 0 and 1. Logistic regression is similar to a linear regression, but the curve is constructed using the natural logarithm of the "odds" of the target variable, rather than the probability. Moreover, the predictors do not have to be normally distributed or have equal variance in each group.

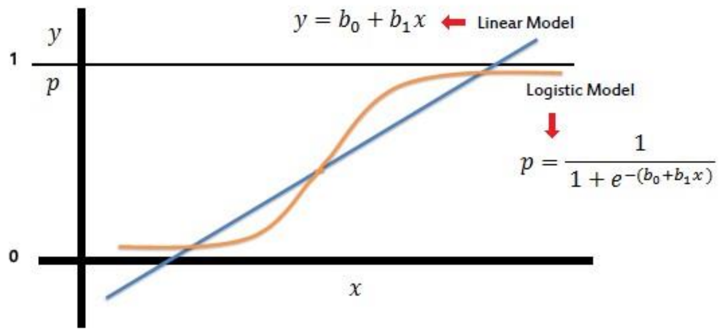
Logistic regression uses maximum likelihood estimation (MLE) to obtain the model coefficients that relate predictors to the target. After this initial function is estimated, the process is repeated until LL (Log Likelihood) does not change significantly.

$$\boldsymbol{\beta}^1 = \boldsymbol{\beta}^0 + [\boldsymbol{X}^T \boldsymbol{W} \boldsymbol{X}]^{-1}.\boldsymbol{X}^T(\boldsymbol{y} - \boldsymbol{\mu})$$

$\boldsymbol{\beta}$ is a vector of the logistic regression coefficients.

$\boldsymbol{W}$ is a square matrix of order N with elements $n_i\pi_i(1-\pi_i)$ on the diagonal and zeros everywhere else.

$\boldsymbol{\mu}$ is a vector of length N with elements $\mu_i = n_i\pi_i$.



```
#LogisticRegression
```

```
s1=time.time()
LRModel=LogisticRegression().fit(Xtrain,Ytrain)
s2=time.time()
lr_tt=s2-s1
print(f"time taken by lr  is {lr_tt}")
```

```
time taken by lr  is 0.7077047824859619
```

```
Ya=Ytest
s1=time.time()
Yp=LRModel.predict(Xtest)
s2=time.time()
lr_ts=s2-s1
print(f"time taken by lr testing is {lr_ts}")
```

```
time taken by lr testing is 0.007616996765136719
```

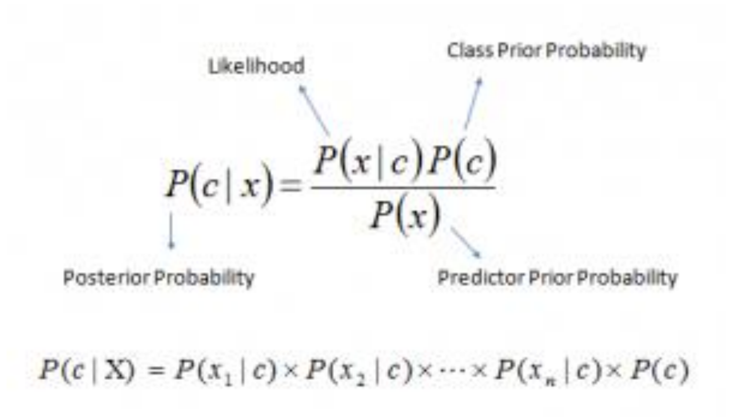
```
accucracy of LR is 87.10714285714286

confusion matrix LR
[[3537  558]
 [ 525 3780]]

classification report LR
               precision    recall  f1-score

 __label__1       0.87      0.86      0.87
 __label__2       0.87      0.88      0.87

    accuracy                          0.87
   macro avg       0.87      0.87      0.87
weighted avg       0.87      0.87      0.87
```

Naive Bayes:

It is a classification technique based on Bayes' Theorem with an assumption of independence among predictors. In simple terms, a Naive Bayes classifier assumes that the presence of a particular feature in a class is unrelated to the presence of any other feature.

Naive Bayes model is easy to build and particularly useful for very large data sets. Along with simplicity, Naive Bayes is known to outperform even highly sophisticated classification methods.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:



$$P(c \mid x) = \frac{P(x \mid c)P(c)}{P(x)}$$

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

- $P(c|x)$ is the posterior probability of *class* (c, *target*) given *predictor* (x, *attributes*).
- $P(c)$ is the prior probability of *class*.
- $P(x|c)$ is the likelihood which is the probability of *predictor* given *class*.
- $P(x)$ is the prior probability of *predictor*.

```
#NaiveBayes
```

```
s1=time.time()
NBModel=MultinomialNB().fit(Xtrain,Ytrain)
s2=time.time()
NB_tt=s2-s1
print(f"time taken by NB  is {NB_tt}")
```

```
time taken by NB  is 0.08815646171569824
```

```
Ya=Ytest
s1=time.time()
Yp=NBModel.predict(Xtest)
s2=time.time()
NB_ts=s2-s1
print(f"time taken by NB testing is {NB_ts}")
```

```
time taken by NB testing is 0.0065784454345703125
```

```
accucracy of NB is 85.47619047619047

confusion matrix NB
[[3460  635]
 [ 585 3720]]

classification report NB
               precision    recall  f1-score   support

  __label__1       0.86      0.84      0.85      4095
  __label__2       0.85      0.86      0.86      4305

    accuracy                           0.85      8400
   macro avg       0.85      0.85      0.85      8400
weighted avg       0.85      0.85      0.85      8400
```

Decision Tree:

A decision tree is a tree-like model that acts as a decision support tool, visually displaying decisions and their potential outcomes, consequences, and costs. From there, the "branches" can easily be evaluated and compared in order to select the best courses of action.

Decision tree analysis is helpful for solving problems, revealing potential opportunities, and making complex decisions regarding cost management, operations management, organization strategies, project selection, and production methods.

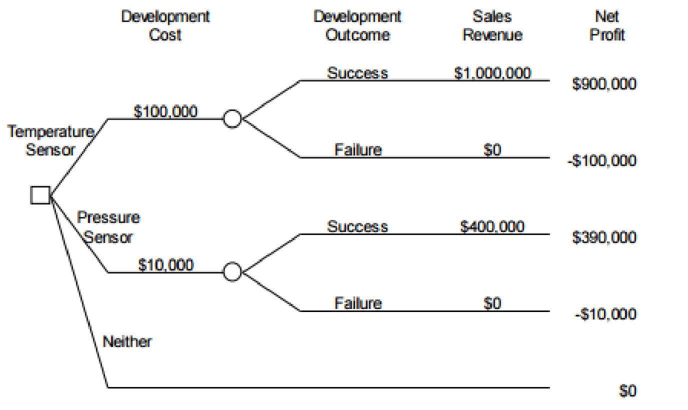


Figure 1.1 *Special Instrument Products decision*

```
#DecisionTree
```

```
s1=time.time()
DTModel=DecisionTreeClassifier().fit(Xtrain,Ytrain)
s2=time.time()
DT_tt=s2-s1
print(f"time taken by DT  is {DT_tt}")
```

```
time taken by DT  is 16.61818838119507
```

```
Ya=Ytest
s1=time.time()
Yp=DTModel.predict(Xtest)
s2=time.time()
DT_ts=s2-s1
print(f"time taken by DT testing is {DT_ts}")
```

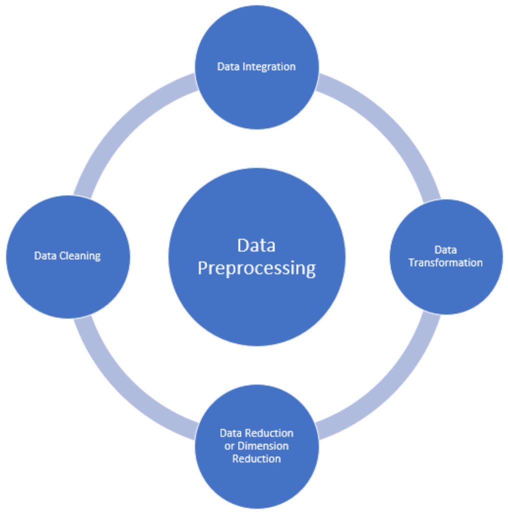
```
time taken by DT testing is 0.026074409484863281
```

```
accucracy of DT is 72.72619047619048

confusion matrix DT
[[2963 1132]
 [1159 3146]]

classification report DT
               precision    recall  f1-score   support

  __label__1       0.72      0.72      0.72      4095
  __label__2       0.74      0.73      0.73      4305

    accuracy                           0.73      8400
   macro avg       0.73      0.73      0.73      8400
weighted avg       0.73      0.73      0.73      8400
```

7. Data Preprocessing

Data pre processing can refer to manipulation or dropping of data before it is used in order to ensure or enhance performance, and is an important step in the data mining process. The phrase “garbage in, garbage out” is particularly applicable to data mining and machine learning projects. Data-gathering methods are often loosely controlled, resulting in out of range values (e.g., Income: −100), impossible data combinations (e.g., Sex: Male, Pregnant: Yes), and missing values, etc. Analyzing data that has not been carefully screened for such problems can produce misleading results. Thus, the representation and quality of data is first and foremost before running any analysis. Often, data pre processing is the most important phase of a machine learning project, especially in computational biology.

If there is much irrelevant and redundant information present or noisy and unreliable data, then knowledge discovery during the training phase is more difficult. Data preparation and filtering steps can take considerable amount of processing time. Examples of data pre processing include cleaning, instance selection, normalization, one hot encoding, transformation, feature extraction and selection, etc. The product of data pre processing is the final training set.

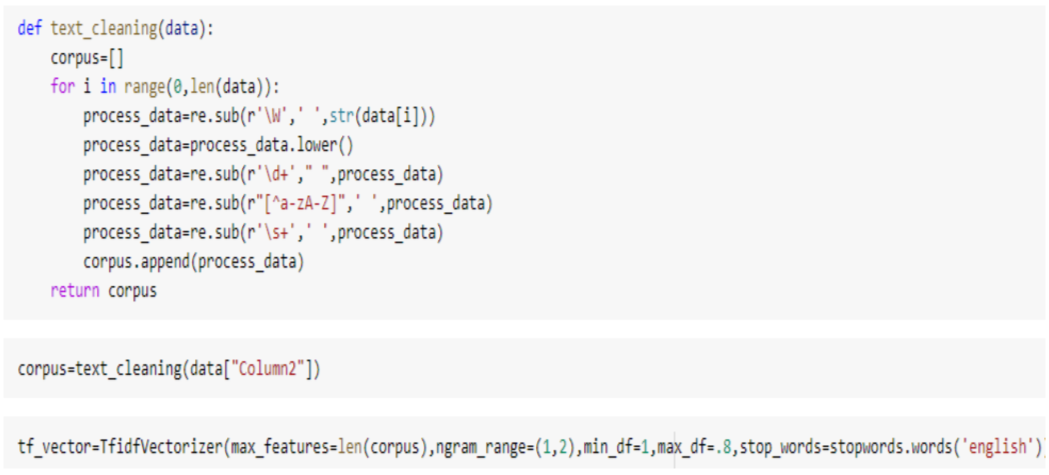


NLTK:

The Natural Language Toolkit (NLTK) is a platform used for building Python programs that work with human language data for applying in statistical natural language processing (NLP).
It contains text processing libraries for tokenization, parsing, classification, stemming, tagging and semantic reasoning. It also includes graphical demonstrations and sample data sets as well as accompanied by a cook book and a book which explains the principles behind the underlying language processing tasks that NLTK supports.

TF-IDF Vectorization:
Term frequency — Inverse document frequency (TFIDF) is based on the Bag of Words (BoW) model, which contains insights about the less relevant and more relevant words in a document. The importance of a word in the text is of great significance in information retrieval.
It is a measure of the frequency of a word (w) in a document (d). TF is defined as the ratio of a word's occurrence in a document to the total number of words in a document. The denominator term in the formula is to normalize since all the corpus documents are of different lengths.

$$TF(w,d) = \frac{occurences\ of\ w\ in\ document\ d}{total\ number\ of\ words\ in\ document\ d}$$



```
def text_cleaning(data):
    corpus=[]
    for i in range(0,len(data)):
        process_data=re.sub(r'\W',' ',str(data[i]))
        process_data=process_data.lower()
        process_data=re.sub(r'\d+'," ",process_data)
        process_data=re.sub(r"[^a-zA-Z]",' ',process_data)
        process_data=re.sub(r'\s+',' ',process_data)
        corpus.append(process_data)
    return corpus
```

```
corpus=text_cleaning(data["Column2"])
```

```
tf_vector=TfidfVectorizer(max_features=len(corpus),ngram_range=(1,2),min_df=1,max_df=.8,stop_words=stopwords.words('english'))
```

Implementation Details:

The model trained is converted to pickle file using joblib library.Two pickle files are created one is for model and another one is for tf_vector.

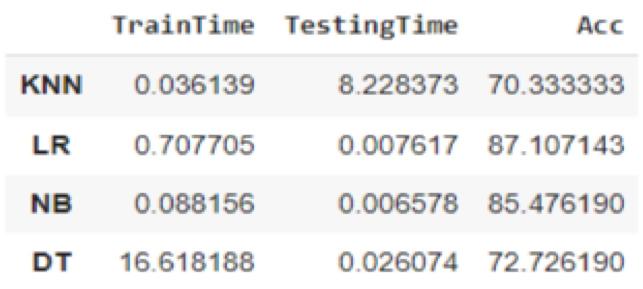
```
# joblib.dump(NBModel,'Model.pkl')
# joblib.dump(tf_vector,'tfvector.pkl')
```

```
['Model.pkl']
```

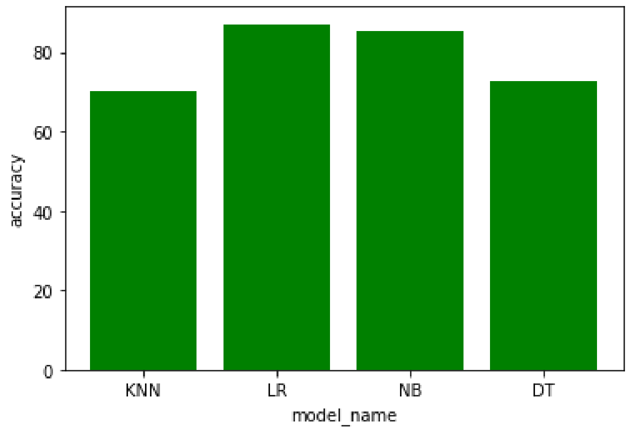
8. Results & Sample Output

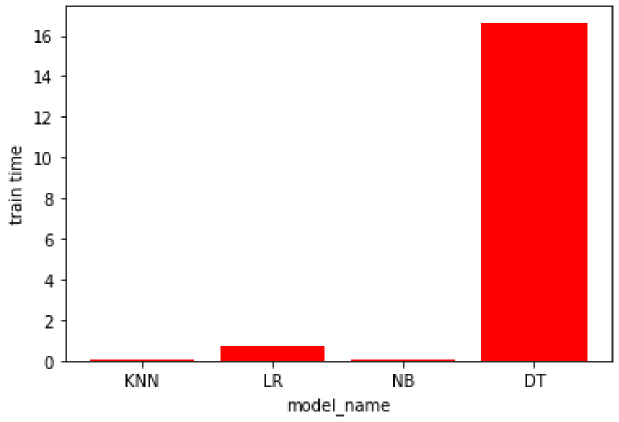
After training four different models for each algorithm K-Nearest Neighbour Classification, Logistic Regression, Naive Bayes & Decision Tree.

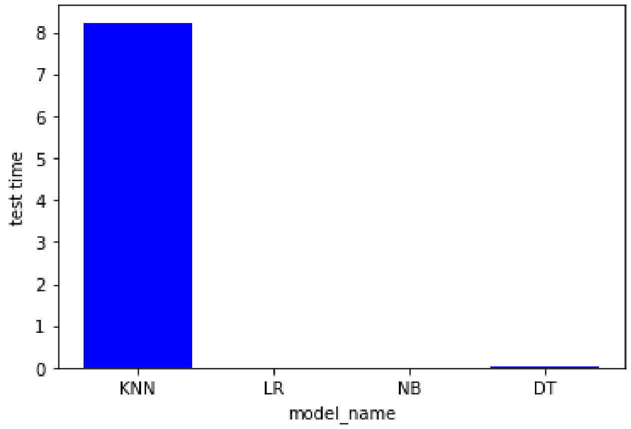
We created a Dataframe with the names of algotithm vs the Training Time, Testing Time & Accuracy. We also plotted Bar Graph for visualization and to select one of the four algorithms which performs better than all three to use as final model.



	TrainTime	TestingTime	Acc
KNN	0.036139	8.228373	70.333333
LR	0.707705	0.007617	87.107143
NB	0.088156	0.006578	85.476190
DT	16.618188	0.026074	72.726190

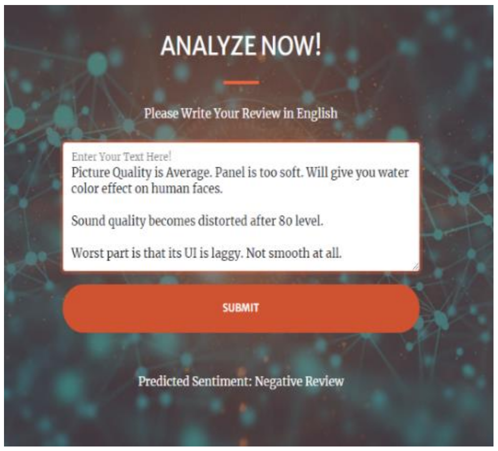






As shown in the above graphs Naive Bayes model performs better than the other algorithms so we used Naive Bayes model as final model.

Given below are some sample outputs for the used model.

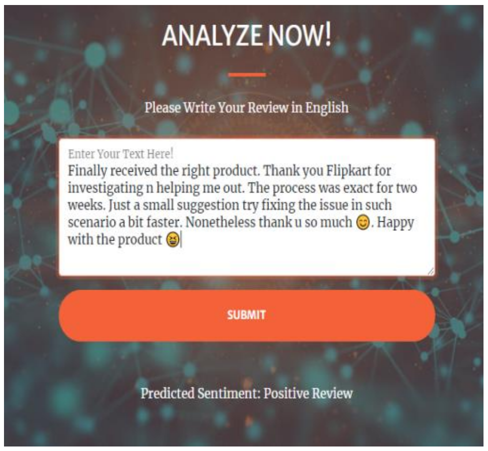


Picture Quality is Average. Panel is too soft. Will give you water color effect on human faces.

Sound quality becomes distorted after 80 level.

Worst part is that its UI is laggy. Not smooth at all.

Predicted Sentiment: Negative Review



Finally received the right product. Thank you Flipkart for investigating n helping me out. The process was exact for two weeks. Just a small suggestion try fixing the issue in such scenario a bit faster. Nonetheless thank u so much 😊. Happy with the product 😛

Predicted Sentiment: Positive Review

9. Conclusion & Future Scope

We have seen that Sentiment Analysis can be used for analyzing opinions in blogs, articles, Product reviews, Social Media websites, Movie-review websites where a third person narrates his views. We also studied NLP and Machine Leaming approaches for Sentiment Analysis. We have seen the implementation of Sentiment Analysis via Classification Algorithms. We have seen that sentiment analysis has many applications and it is important field to study. Sentiment analysis is an emerging research area in text mining and computational linguistics, and has attracted considerable research attention in the past few years. Future research shall explore sophisticated methods for opinion and product feature extraction, as well as new classification models that can address the ordered labels property in rating inference. Applications that utilize results from sentiment analysis is also expected to emerge in the near future.Sentiment analysis has Strong commercial interest because Companies want to know how their products are being perceived and also Prospective consumers want to know what existing users think.

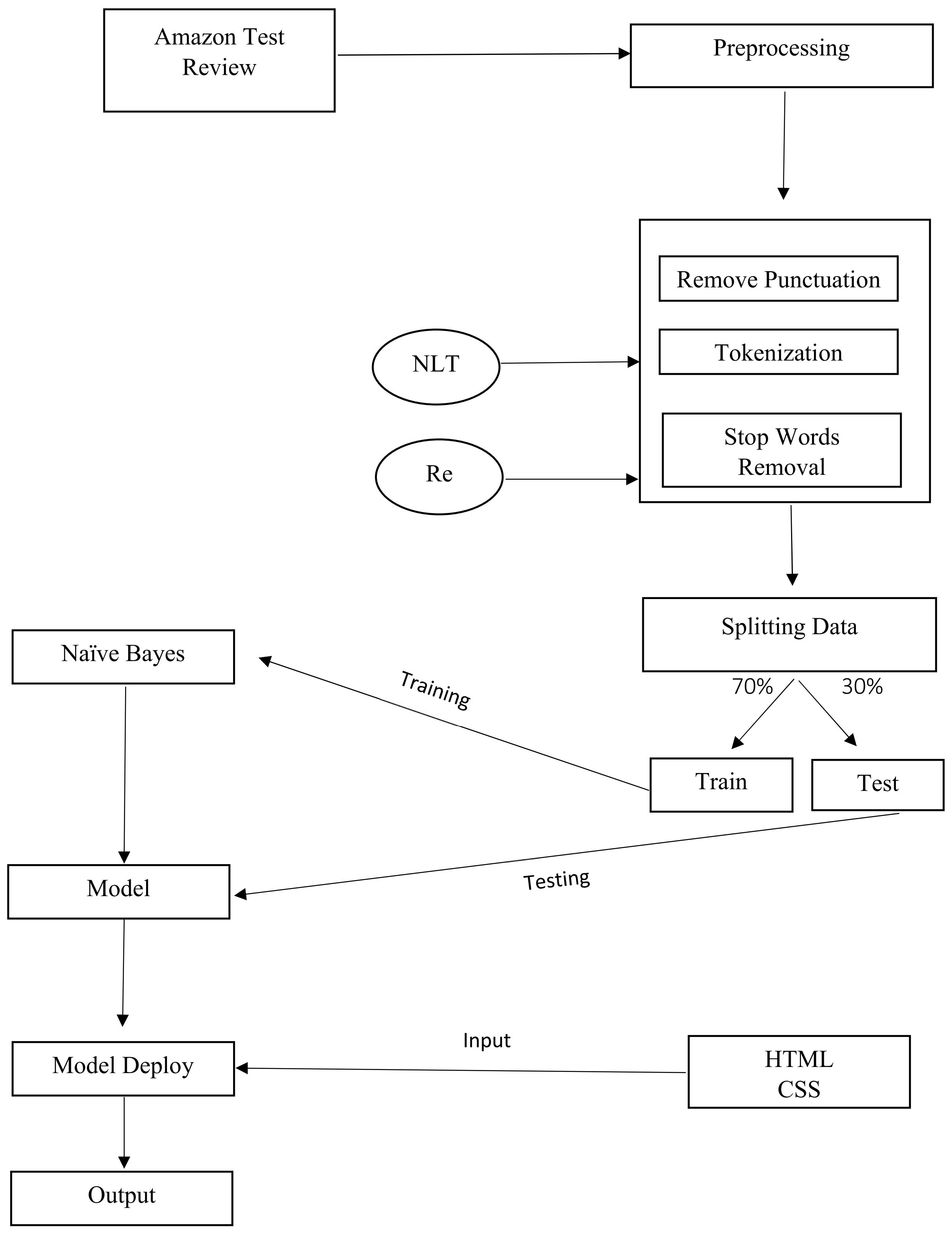
Some future scopes of Sentimental Analysis are:

- An application can be made for our work in future.
- We can improve our system that can deal with sentences of multiple meanings.
- We can also increase the classification categories so that we can get better results.
- We can start work on multi languages like Hindi, Spanish, and Arabic to provide sentiment analysis on other languages.

References

- https://en.wikipedia.org/wiki/Sentiment_analysis
- https://scikit-learn.org/stable/modules/linear_model.html#logistic-regression
- https://scikit-learn.org/stable/modules/neighbors.html#nearest-neighbors-classification
- https://scikit-learn.org/stable/modules/naive_bayes.html#multinomial-naive-bayes
- https://scikit-learn.org/stable/modules/tree.html#classification
- https://scikit-learn.org/stable/modules/model_evaluation.html#classification-metrics
- https://www.nltk.org/
- https://towardsdatascience.com/tf-idf-for-document-ranking-from-scratch-in-python-on-real-world-dataset-796d339a4089

WORKFLOW DIAGRAM



Amazon Test Review
Preprocessing
Remove Punctuation
Tokenization
Stop Words Removal
NLT
Re
Splitting Data
70%
30%
Naïve Bayes
Training
Train
Test
Model
Testing
Model Deploy
Input
HTML CSS
Output