|  |  |
| --- | --- |
| **Abhishek Tiwari**  **Master’s of Data Science**  **Maulana Abdul Kalam Azad Institute of Technology**  **West Bengal** | Abstract  Semantic Analysis of Natural language Processing- Analysis of Amazon product review text to determine if it’s good or bad review and predict the product rating based on the text. |

Contents

[Introduction to Natural Language Processing 2](#_Toc26025707)

[Semantic Analysis: 2](#_Toc26025708)

[Project Background: 2](#_Toc26025709)

[Data Preparation and Processing: 3](#_Toc26025710)

[1. Dealing with missing values 3](#_Toc26025711)

[2. Tokenization: 3](#_Toc26025712)

[3. Stopwords: 4](#_Toc26025713)

[5. Stemming: 4](#_Toc26025714)

[6. Vectorization: 4](#_Toc26025715)

[7. Scaling: 4](#_Toc26025716)

[Validation Data: Train Test Split: 4](#_Toc26025717)

[Predicting if the review is Good or Bad 5](#_Toc26025718)

[Logistic Regression Model: 5](#_Toc26025719)

[Predicting rating of product based on text reviews 7](#_Toc26025720)

[1. Naive Bayes’ Classifier: 7](#_Toc26025721)

[2. Decision Tree Classifier: 9](#_Toc26025722)

[3. Random Forest Classifier: 10](#_Toc26025723)

[4. Why Support Vector Machine and K-Nearest Neighbors could not be used for this project? 11](#_Toc26025724)

[5. Neural Network (Feed Forward Sequential Neural Network): 12](#_Toc26025725)

[Conclusion 14](#_Toc26025726)

[References 15](#_Toc26025727)

[Project link: 15](#_Toc26025728)

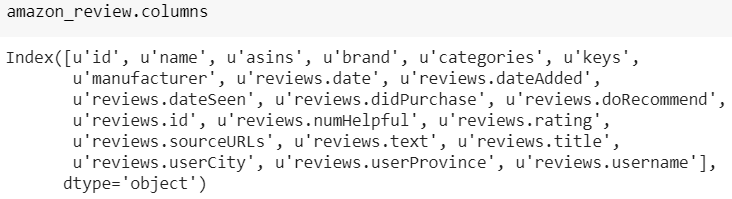
[Project Codes: 15](#_Toc26025729)

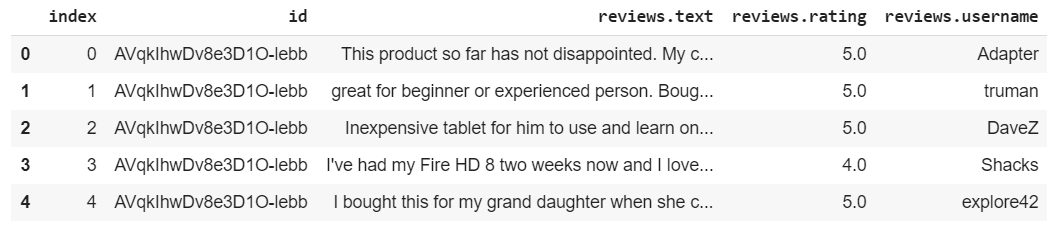
# Introduction to Natural Language Processing

Natural Language Processing is a subfield of many different fields like linguistic, Computer Science and Artificial Intelligence that tries to understand the interactions between computer and Human Natural Language. It deals with processing and analyzing a large amount of Natural language data to understand how humans interact and use that understanding in real-life applications like voice commands to robots, speech recognition, language translations, automatic summarizations, chatbots and many others. Though it sounds easy, Natural Language Processing is one of the hardest fields of Computer Science and Artificial Intelligence. Ambiguity, satirical nature, lack of clarity and preciseness, emotions behind human interactions are some of the reasons which make NLP hard. Machines not only need to understand the words but also how they are ordered, linked together to create a meaning to do any task. There can be either Syntactical NLP analysis that is basically rule-based and applies grammatical rules to the text and Semantical NLP analysis involves understanding the meaning behind the text.

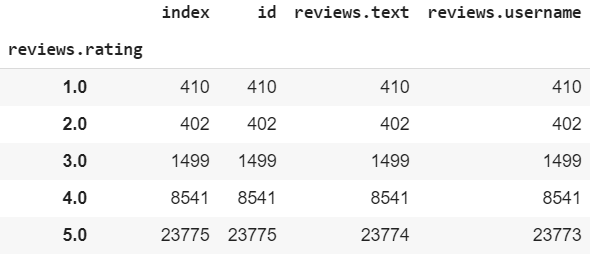
Semantic Analysis:Semantic Analysis is understanding the meaning of the text. It analyzes the surrounding words and understands the context of the text. It processes the logical structure of the text, grammatical rules, meaning of the words and understands the relationships between words.

Project Background: This project tries to understand the relation between reviews and ratings on different products given by the people on the Amazon website. Data used to do this project can be found on the Kaggle website "<https://www.kaggle.com/datafiniti/consumer-reviews-of-amazon-products>". The data set has a total of 34660 reviews and 21 variables like reviewer username, product id, product name, reviews, review rating and other. We are interested only in reviews and rating for this NLP project. Reviews would be our Feature dataset and Rating would be our label.



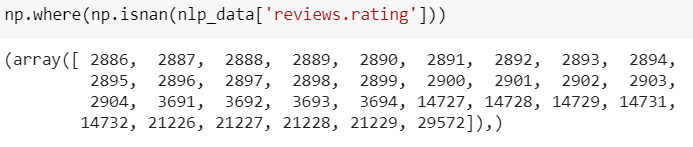


We can see we have majority of products having ratings as 5 and 4 which means customers had satisfactory experience with the product. Companies would be worried with the ratings 1,2 and 3 and want to make experiences of these users better and improve the quality of products so, that they reduce these lower ratings user experiences.



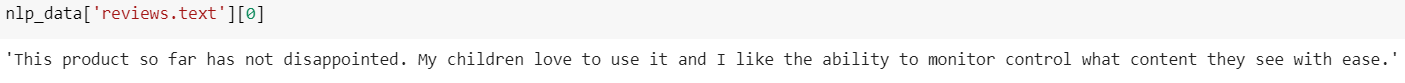
# Data Preparation and Processing:

1. Dealing with missing values**:** There were 33 reviews which have no ratings and 1 rating with no reviews. Since these numbers are too less as compared to the data size of 34626, these records were dropped as we don't want our model to learn from wrong data which would be created because of data imputations. These are the values where we have no rating for the product.



2. Tokenization: Now let’s see step by step, each review would be converted to machine readable vectors of 0 and 1. We then removed all the punctuations from all the reviews, converted all of them to lower case and made a list containing all words.

**This is the original review:**



3. Stopwords: Using Natural Language toolkit, we removed all stop words which would not contribute to predicting ratings. These words are like I, this, there, place, when etc. Words like love, hate would only be left which predicts rating. For example, " I loved the product" and " I hated the product" have a rating of 5 and 0 respectively. Here, "love" and "hate only predicts rating of 5 and 0.

5. Stemming:Then all the words having common prefixes and suffixes that can be found in an inflected word were stemmed to the root word which might not be an actual English word, so, that sentences having the same root word can be trained equally. It also decreases the number of independent variables that words would form later. This makes training efficient. One more process is Lemmatization, it's like Stemming with the difference that root words are actual formal English words.

**After all the above steps the review is converted to this form:**



6. Vectorization: Then these newly processed reviews are converted into vectors with all the words in the reviews as columns and all the reviews as rows. A column number of times each word appears in each review. If 'love' appears in 1st review 1 time then it would be 1, if doesn't appear then 0 and if appears 2 times then 2. Most of the words appear only once in each review thus forming one huge sparse matrix. One of the challenges in Machine Learning is to reduce the sparsity of the matrix and several studies are done to do that and increase efficiency. For this data, we had 34626 rows which are basically the number of reviews and 9103 columns, which are the aggregate number of unique words present in all reviews together.

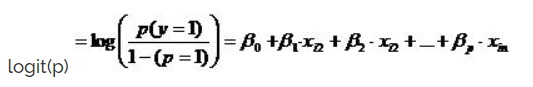
7. Scaling: We scaled the vectors to remove make them centered to 0 and make the variance in the same range.

Validation Data: Train Test Split: We split the data set into 80:20 ratio for initially training the models on 80% data and testing on rest 20%. However, we would also use 10-fold cross-validation to check the stable accuracy of the model. We could have optimized this but since we have 9103 independent variables, so it becomes so hard to do optimization on normal computers.

**Developing models:** Now that we have a scaled vector of 9103 columns and 24626 rows. This is nothing but normal multi-class classification variables with 5 outputs possible.

# Predicting if the review is Good or Bad

Logistic Regression Model: As this is Binary Classifier problem, we used logistic regression because of its prediction power as well as easier to interpret as compared to complex methods like Random Forest, Linear Discriminant Analysis, Neural Network etc. However, we have used those complex models to predict product rating based on review being multi-classifier problem. Logistic regression has below math behind it. Log of Odds are linear function of predictors.

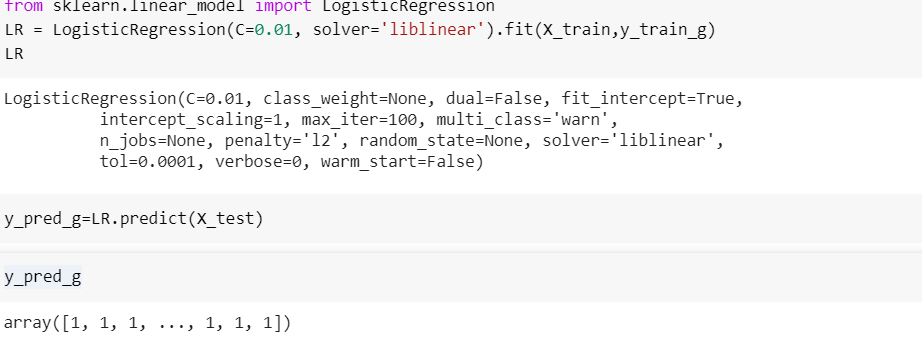


**Here good review is if rating of the product is 4 or greater and bad reviews if rating less than 4. Good has been coded as 1 and bad review as 0.**



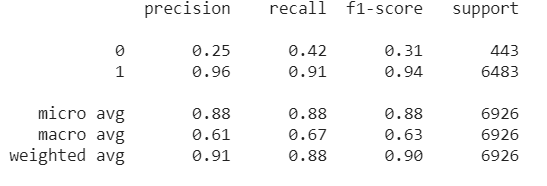
**We are trying to classify good and bad reviews through logistic regression using Sklearn library**

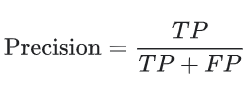
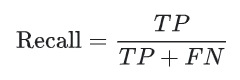
**Model setup:**



Here C is inverse of Regularization strength, must be positive float. Small value means strong regularization. Regularization basically penalize the loss function for its complexity. Here loss function is liblinear. It will try to minimize the loss function in 100 iterations.

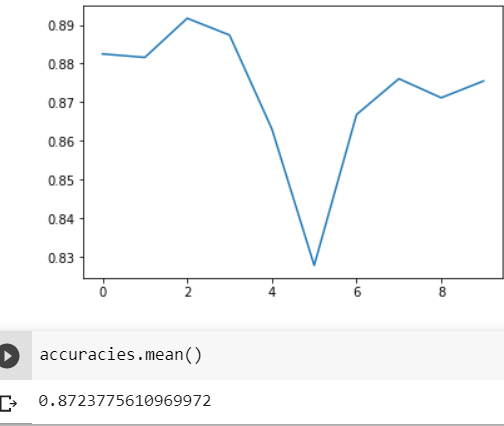
Logistic Regression had a good accuracy when tested on test data:



Here precision is correct true prediction among overall true predictions for bad review is quite low and precision is quite high for good reviews. Overall model has decent precision of 88%. Recall is correct true prediction among actual true values. We can see model is predicting 91% correct for good reviews and only 42% for bad reviews. Support is number of samples in that class. We can model has predicted 443 bad reviews and 6483 good reviews out of total 6926 reviews in test dataset. F-1 score is high for good reviews as 94% and 31% for bad reviews indicating model is predicting good reviews well but lacking accuracy in predicting bad reviews. This can be due to not enough bad reviews to train from in training datasets.

We performed 10-fold cross validation to check the optimum accuracy of our logistic regression model. In this method, data is divided into 10 equal random fold and model is trained 10 times on 9 training datasets and validate the model on 1 remaining fold.



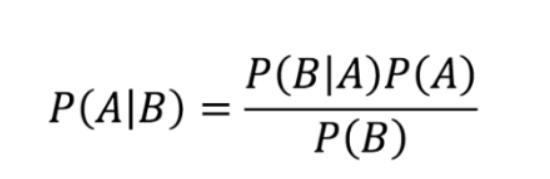
We can see we have a highest accuracy of more than 89% on 3rd validation and mean accuracy of 87.23% which is quite good performance.

# Predicting rating of product based on text reviews

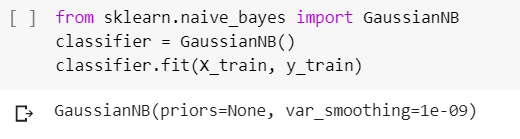
In this part of project, we would be predicting rating of the product based on text reviews. Rating ranges from 1 to 5 with 5 as excellent and 1 being least. We would be using multiclassification methods like Naïve Bayes’, Decision Tree, Random Forest, Support Vector Machine, K-Nearest Neighbor and Artificial Neural Network model.

1. Naive Bayes’ Classifier: I started with Naive Bayes Classifier because it performs well on multi-class categorical text problems because of its less training data and time requirement. Also, if independent conditional probability of observation on class holds true then it gives quite good predictions. However, in this project, it had the worst performance because of the way too many independent variables and complexity of the relation between reviews and ratings. Highest accuracy it gave was 15.65%.

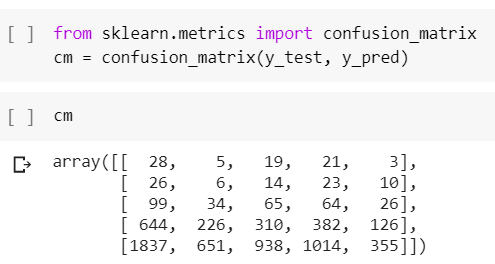
Naïve Bayes’ Classifier has following. The goal of any probabilistic classifier is, with features x\_0 through x\_n and classes c\_0 through c\_k, to determine the probability of the features occurring in each class, and to return the most likely class. Therefore, for each class, we want to be able to calculate P(c\_i | x\_0, …, x\_n). In the context of classification, you can replace A with a class, c\_i, and B with our set of features, x\_0 through x\_n. Since P(B) serves as normalization, and we are usually unable to calculate P(x\_0, …, x\_n), we can simply ignore that term, and instead just state that P(c\_i | x\_0, …, x\_n) ∝ P(x\_0, …, x\_n | c\_i) \* P(c\_i), where ∝ means “is proportional to”. P(c\_i) is simple to calculate; it is just the proportion of the dataset that falls in class i. P(x\_0, …, x\_n | c\_i) is more difficult to compute. In order to simplify its computation, we make the assumption that x\_0 through x\_n are **conditionally independent** given c\_i, which allows us to say that P(x\_0, …, x\_n | c\_i) = P(x\_0 | c\_i) \* P(x\_1 | c\_i) \* … \* P(x\_n | c\_i). This assumption is most likely not true — hence the name naive Bayes classifier, but the classifier nonetheless performs well in most situations



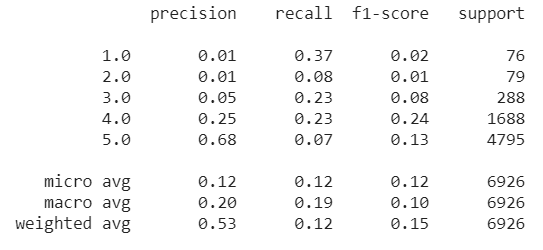
Our model named Gaussian Naïve Bayes’ is imported from Sklearn Naïve Bayes’ library

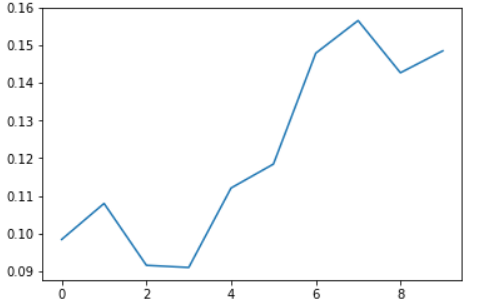


Let’s see the confusion matrix and assess how our Naïve Bayes’ model performed:



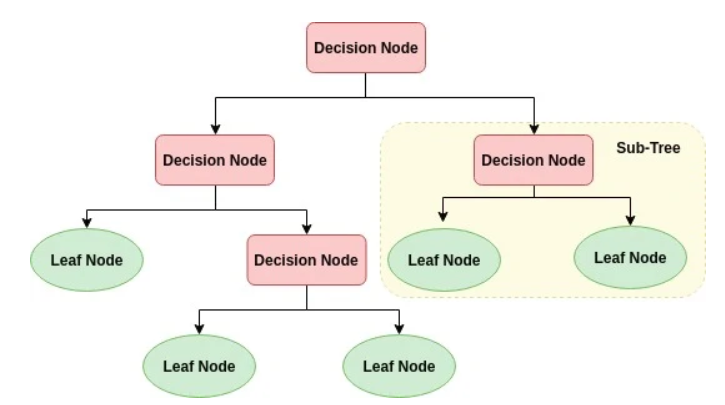
We can see how bad Naïve Bayes’ model worked on this text analysis data with accuracy of only 12% and highest precision of 68% for review rating 5.

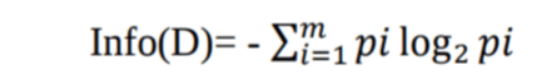
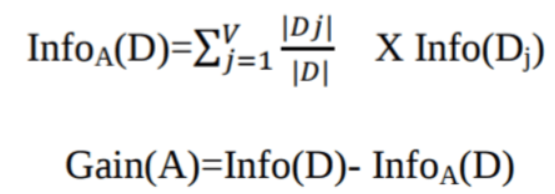




Through 10-fold cross validation, it has highest efficiency of 15.65%. Clearly we need to find the model with better accuracy.

2. Decision Tree Classifier: Decision Tree Classifier increased the performance dramatically to 62%. This is quite intuitive that there is a tree-like relationship between words or combination of words and ratings. Decision Tree works in following way:



Where,

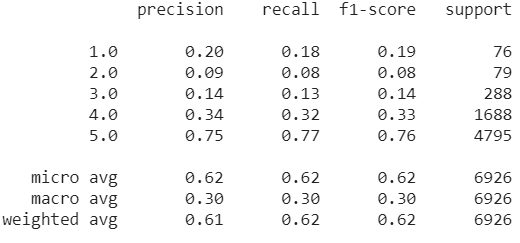
Info(D) is the average amount of information needed to identify the class label of a tuple in D.

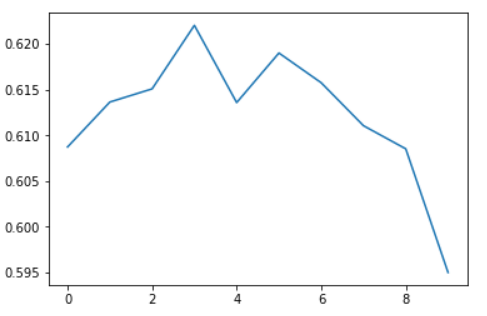
|Dj|/|D| acts as the weight of the jth partition.

InfoA(D) is the expected information required to classify a tuple from D based on the partitioning by A. The attribute A with the highest information gain, Gain(A), is chosen as the splitting attribute at node N().

Decision tree classifies based on entropy and information gain. Entropy is basically instability in the data, and it tries to reduce it and subsequently increase the information gain after each branching. We can see Decision Tree did very good job over Naïve Bayes’ for this text analysis with accuracy of 62% and good precision of 77% for rating 5 however, precision for lower ratings are still bad and we aim to increase the precision of this lower ratings because that’s what company would be worried about.

For 10-Fold Cross-validation, we can see we accuracy ranges from 59.5 % to 62%. We can assume, Decision tree has optimum accuracy of 61% approximately.

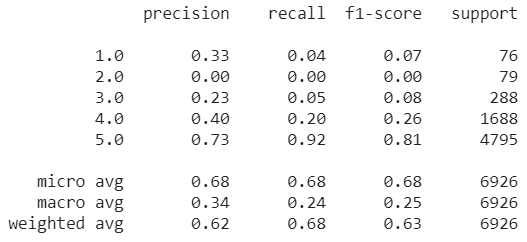


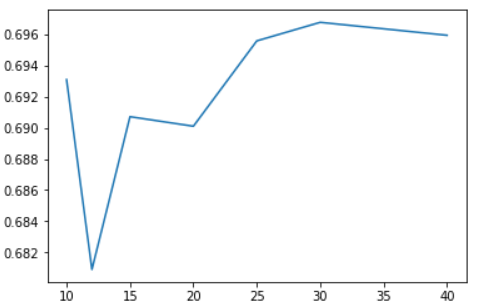


3. Random Forest Classifier: To classify a new object from an input vector, input vector is classified by each of the trees in the forest. Each tree gives a classification, and we say the tree "votes" for that class. The forest chooses the classification having the most votes (over all the trees in the forest). Each tree is grown as follows:

1. If the number of cases in the training set is N, sample N cases at random - but with replacement, from the original data. This sample will be the training set for growing the tree.
2. If there are M input variables, a number m<<M is specified such that at each node, m variables are selected at random out of the M and the best split on these m is used to split the node. The value of m is held constant during the forest growing.
3. Each tree is grown to the largest extent possible. There is no pruning

This is just when there are multiple trees in the classifier model and in my model, performance increased to an average of 69.8% on 10-fold cross-validation for the decision tree of 30. I had to optimize the number of trees and the performance ranged from 68% when number of estimators were equal 10 to 69 % when 40 with the maximum at 30.



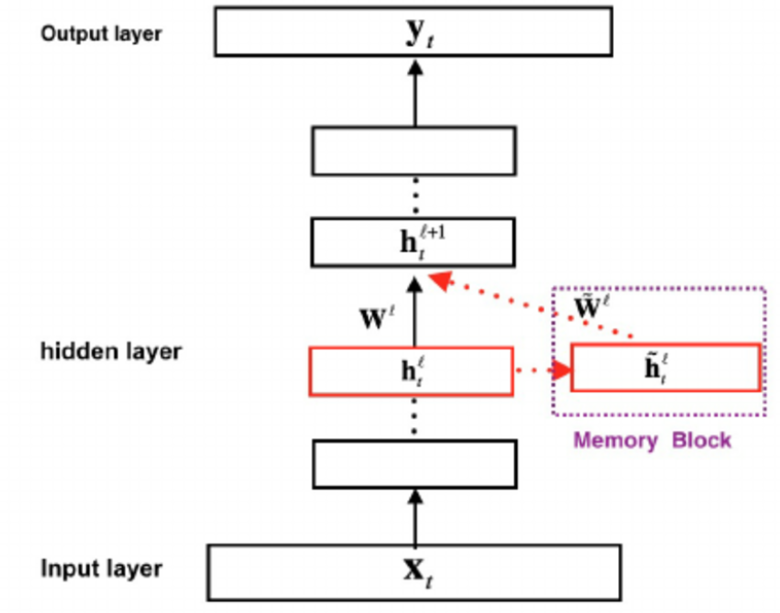


### 4. Why Support Vector Machine and K-Nearest Neighbors could not be used for this project?

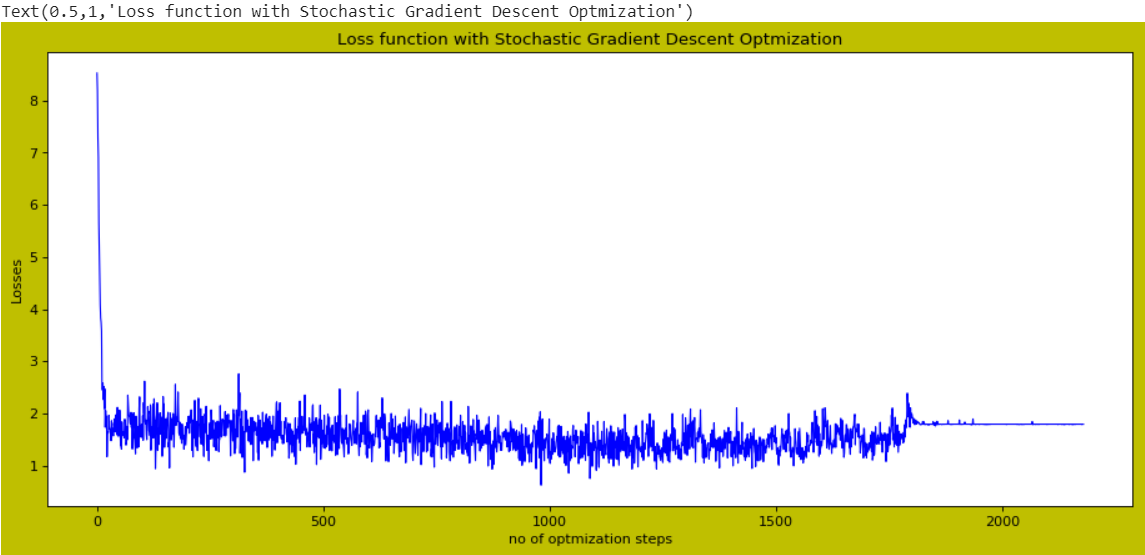
Support Vector Machine and K-Nearest Neighbor Classifier could not be used for this vector because of the very high dimensionality of 9103 independent variables. SVM tries to bring all data together to RAM all at once and tries to understand the relation between independent and dependent variables so it's quite efficient for less training data set with a smaller number of variables. KNN wouldn't work for similar reasons, it might work for little less high dimensional data with Manhattan distance but with Euclidean distance, it's just impossible because of it's a computational requirement.

### 5. Neural Network (Feed Forward Sequential Neural Network):

I tried to start with Sequential model for the sake of the computational efficiency. My model had one input layer of 9103 neurons (number of independent variables), one hidden layer of 600 neurons and one output layer of five neurons (number of categorical ratings). Activation function used is Rectified Linear unit, the Loss function is Sparse Categorical cross-entropy and loss optimizer is Stochastic Gradient Descent. I started with a batch size of 72 and epoch size of 50 but due to computational issue, I had to increase the batch size to 256 and decrease the epochs to 20. So, my neural network learnt from (27770(training data) /256(batch) \*20(epochs) =2169) over 2000 iterations. We can take a quick look at what happens behind the process. The batch size should usually be divisible by 8 and training data set should be divisible by batch size but not necessarily, except that last batch would be remainder data sets. In my project, Neural network would train based on the input of 256 samples from training datasets understands the relation estimates the weight(w) then take on next 256, understands more, updates the weights for the 1st layer or can say learns a bit more. It keeps on doing it until the all 27770 data points in my training set are propagated. Epoch size determines how many times the same data sets would train the model. The process is like once, the whole training data is propagated, the same data enters the model again in same 256 batches for 20 times. Each time the iteration is done, the model learns more and updates the weights and does optimization to decrease the loss.



Using this Neural Network model, we achieved an accuracy of 99.38%. We can see how loss decreased with the 2169 Stochastic Gradient Descent optimization. The loss function got minimum at around 1000 optimizations with a loss of 0.62%. We can see that after 1800 iterations, loss function can’t be minimized further.



We checked the model by inputting our own reviews:

|  |  |
| --- | --- |
| "I love the product, its really long lasting" | **5** |
| "I hated the product, it was such a waste of money" | **1** |
| "I fucking loved it" | **5** |
| "Using this product was a nightmare" | **5** |

**Model has predicted all the reviews correctly but the last one. This may be because model is using a word nightmare for the first time or it has seen a 5 rating where these words were used but maybe in a different context.**

# Conclusion

1. Neural Network increases the performance of the high dimensional Natural Language Processing Bag of Word model significantly and can give almost 100% accuracy if trained on the smaller batch size with quite high epochs. However, doing so is computationally very hard and time taking.
2. Support vector machine and K-Nearest Neighbor classifier are very hard to develop on very high dimensional data. However, SVM can perform very well on NLP and research is going on this area.
3. Random Forest Classifier works better than Decision Tree however can cause overfitting, so the number of estimators should be optimized efficiently.
4. K-Fold Cross Validation is very important method to find optimum accuracy of any model and to tune in the parameters. For example, Random Forest in this project had best accuracy when number of trees were 40.
5. All the models besides Neural Network had low accuracy for lower ratings and that’s because model is unable to distinguish between words used in lower rating reviews. This is because of the way human speaks and multiple aspects of same word and the way they are spoken.
6. Our sample data has most of the reviews from ratings 5 and 4 and very less reviews from 1,2,3 reviews that’s why model doesn’t have enough low ratings reviews to train on, so that model can distinguish between low ratings reviews.

# References

* <https://www.stat.berkeley.edu/~breiman/RandomForests/cc_home.htm>
* <https://towardsdatascience.com/>
* <https://wiki.eecs.yorku.ca/user/hj/_media/publication:cfsmn-final.pdf>
* <https://scikit-learn.org/stable/modules/generated/sklearn.metrics.classification_report.html>
* <https://www.datacamp.com/community/tutorials/decision-tree-classification-python>
* <https://www.coursera.org/professional-certificates/ibm-data-science>

### Project link:

<https://colab.research.google.com/drive/10DlMDNfSszBslPtFggMZUxTIPJcRJ4rO>

### Project Codes:

From next page onwards, we have the full python project codes