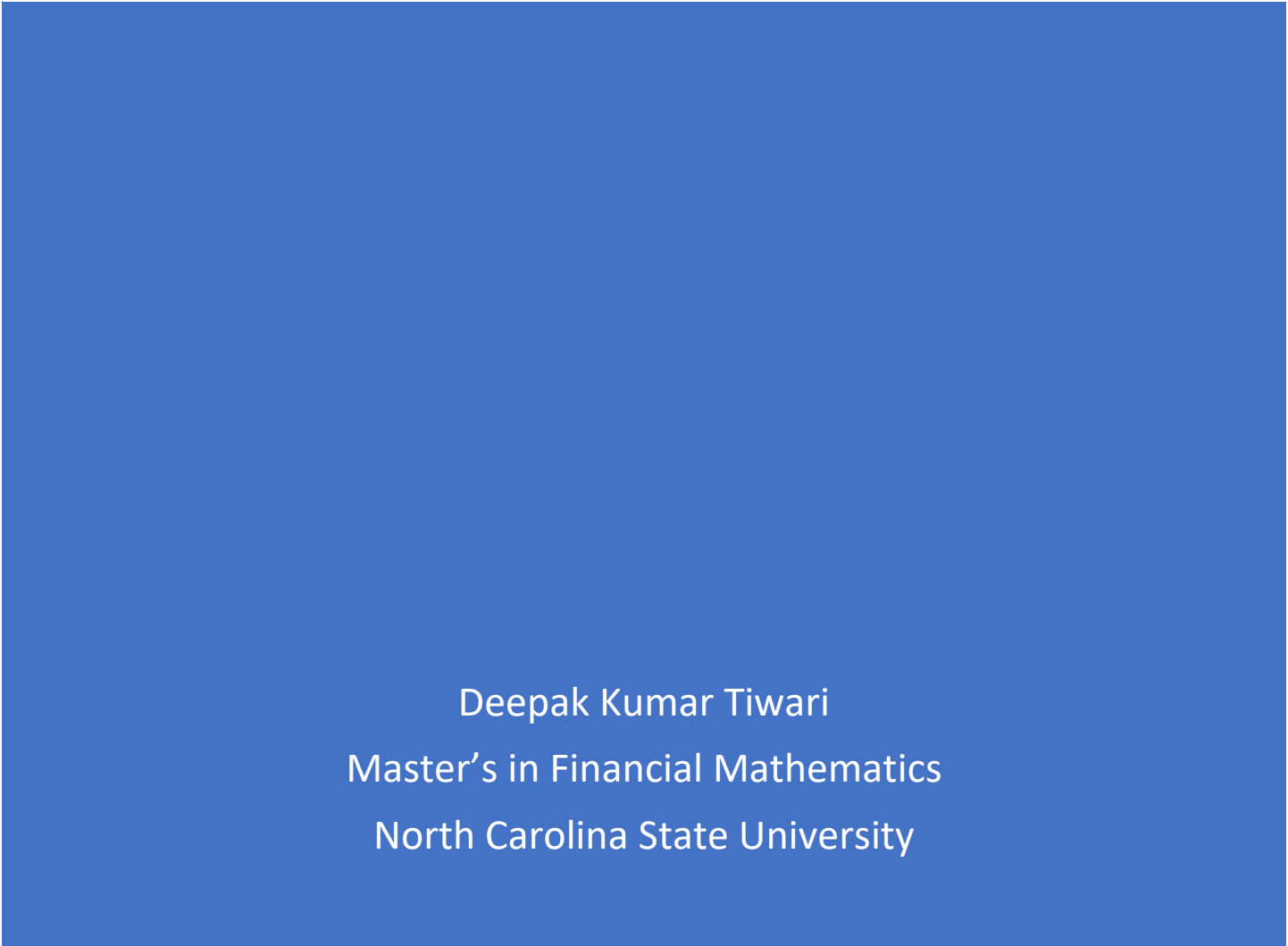




# NATURAL LANGUAGE PROCESSING USING DEEP LEARNING



Deepak Kumar Tiwari  
Master's in Financial Mathematics  
North Carolina State University

**Introduction to NLP:** 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.

**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.

### **Data Preparation and Processing:**

**1. Dealing with NaN 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.

**2. Tokenization:** We then removed all the punctuations from all the reviews, converted all of them to lower case and made a list containing all words.

**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 actually 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.

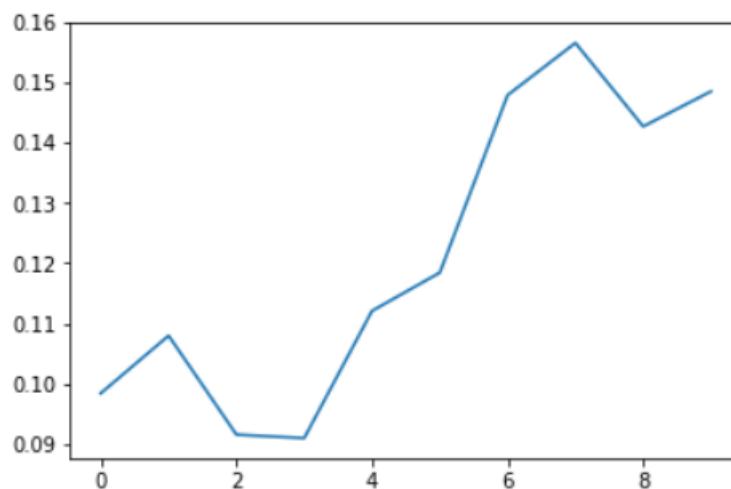
**6. Vectorization:** Then these newly processed reviews are converted into vectors with all the words in the reviews as columns and all the reviews as rows. A column number of times each word appears in each review. If 'love' appears in 1st review 1 time then it would be 1, if doesn't appear then 0 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.

**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.

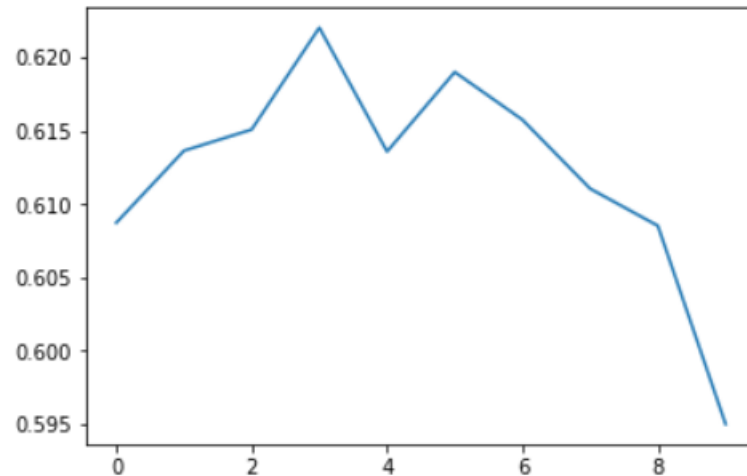
**Developing models:** Now that we have a scaled vector of 9103 columns and 24626 rows. This is nothing but normal multi-class classification variables with 5 outputs possible.

**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%.



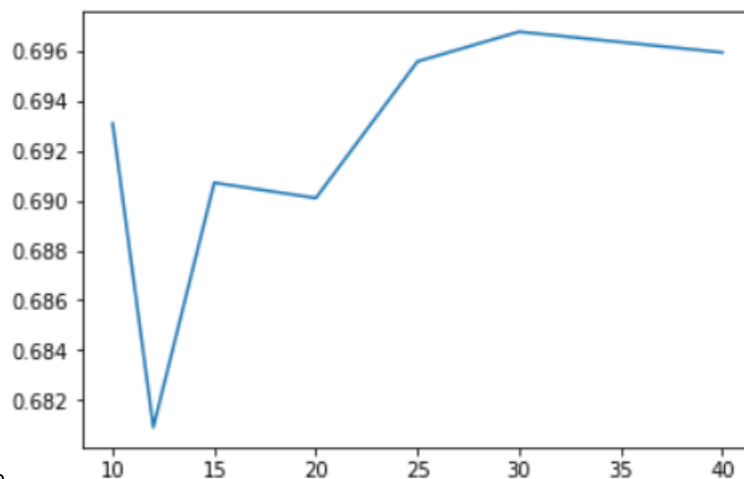
Add description

**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.



Add description

**3. Random Forest Classifier:** 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.



Add description

**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 its computational requirement.

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

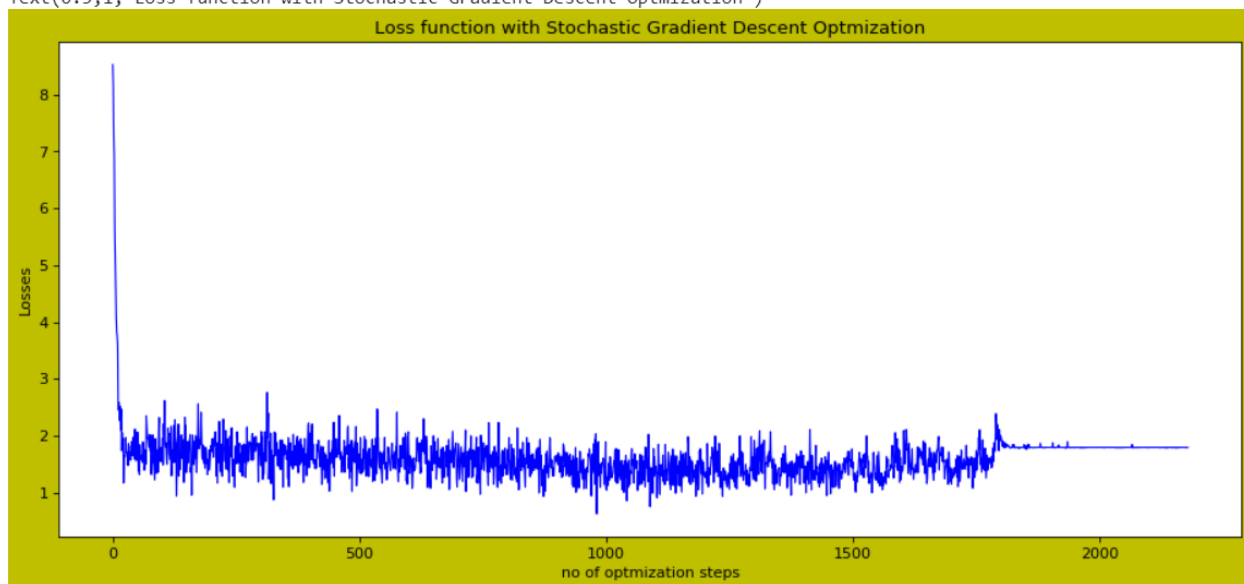
I tried to start with Sequential model for the sake of the computational efficiency. My model had one input layer of 9103 neurons (number of independent variables), one hidden layer of 600 neurons and one output layer of five neurons (number of categorical ratings). Activation function used is Rectified 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  $(27770(\text{training data}) / 256(\text{batch}) * 20(\text{epochs}) = 2169)$  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.

Add

description

Text(0.5,1,'Loss function with Stochastic Gradient Descent Optimization')



**Conclusion:**

1. Neural Network increases the performance of the high dimensional NLP 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.

**You can find the link to the project here:**

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