MLND Capstone Report

Capstone proposal link:

https://review.udacity.com/?utm_campaign=ret_000_auto_ndxxx_submission-reviewed&utm_source=blueshift&utm_medium=email&utm_content=reviewsapp-submission-reviewed&bsft_clkid=64cbd09c-b1c9-4a98-ad3e-e1524b6c1949&bsft_uid=aa25d999-039c-4dde-b3a6-c14bf61cb092&bsft_mid=c2b5167e-90dc-4a2b-99a3-ae935596f627&bsft_eid=6f154690-7543-4582-9be7-e397af208dbd&bsft_txnid=b3ec98c4-8abe-4039-bbf8-4c4218517150#!/reviews/1324167

1-Definition

1.1-Project Overview:

Natural Language Processing has become the most talked domain in the recent years. As the applications like Text Classification. Topic Modelling, Machine Translation, Sentiment Analysis and Speech Recognition are emerging and using among all industries and become a vital field in the Information Technology. By saying this I am glad to be a part of NLP world and chosen as "Sentiment Analysis' as my MLND capstone project.

https://www.coursera.org/learn/python-text-mining

https://github.com/udacity/deep-learning/tree/master/sentiment-rnn

https://medium.com/udacity/natural-language-processing-and-sentiment-analysis-43111c33c27e

https://www.coursera.org/lecture/text-mining-analytics/5-4-how-to-do-sentiment-analysis-with-corenlp-jPNSG

https://medium.com/ml2vec/using-word2vec-to-analyze-reddit-comments-28945d8cee57

https://cs.stanford.edu/~quocle/paragraph_vector.pdf

1.2-Problem Statement:

The goal of the project is to classify the movie reviews based on the content. Here the content refers to description of the review. Here I have given with input text data and have the attached labels and this comes under supervised learning. Even though the use case seems solvable one but dealing with text is quite complicated compare to numerical data because of the ambiguity exits in here.

Here let me simply brief on the approach. The input text will be fed into the NLP pre-processing and then the cleaned text will go into the next stage of Feature Engineering (TfIdf/Word Embedding) and then to the last stage which is applying Algorithm to the data and finally evaluate each algorithm on the test data to find the best one and the mentioned stages will be explained in detail in the next sections.

1.3-Metrics:

The evaluation metric considered here is the AUC score (precision/recall metrics).

Here the reason for choosing this evaluation metric is based on the characteristic of the problem. Here we are working on classification problem for this kind of problem accuracy is not a good metric like we use in regression problem. Consider a scenario where the skewness exists in the sense (more

labels of 0 and less labels of 1) in this scenario there is a chances that accuracy metric will lean towards more of predicting 0 labels but not 1's , this is where precision/recall comes into picture.

Trying to maximise the precision/recall by minimizing the error term will make the model works better on the datasets. We get a better intuition when we plot an ROC curve. As part of training we will use grid-search techniques to tune the hyper parameters and chose the best parameters that give good AUC score.

Moreover by comparing the final model and benchmark model we can understand much better on both the models. We can compare both model performance metrics such as AUC score, training and prediction times.

These metrics can be defined as follows

Precision = tp/ (tp+fp)

Recall = tp/ (tp+fn)

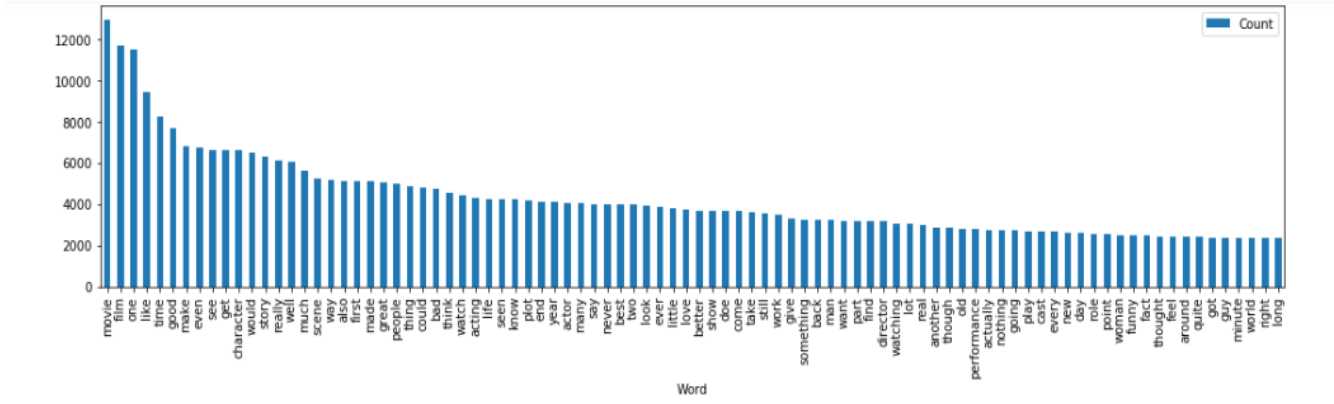
Where tp = true positives, fp = false positives, fn = false negatives

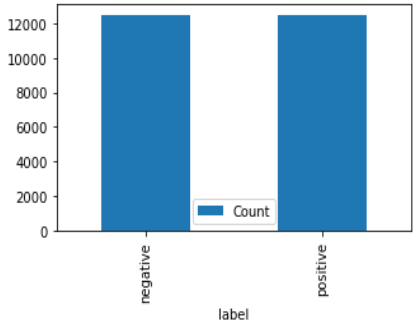
Moreover the best parameter for different models with different hyper parameters will be found using GridSearchCV and this will provide the best parameters that gives good accuracy/precision/recall metrics.

2-Analysis

2.1-Data Exploration/Visualization:

Here are the graphs represents some important information about text data. The label data has equally distributed as positive and negative with each of 12500. When coming to input text data the most talked words are about the term movie/film and some positive words like good, like, great and love and also the terms like plot, story and character.





2.1.1-Sentiment Tweets

Here the data represents the movie reviews with the relevant labels as positive and negative. The movie description considered as input and the labels as output for this use case. The total number of reviews in the dataset were 25K records which is quiet good enough to get a good model. The considered ratio of train, test and validation are 0.8, 0.1 and 0.1 respectively. Below is the link for the dataset. The target labels are balanced in the sense both the positive and negative labels has equal quantity with each of 12500 records.

https://github.com/udacity/deep-learning/tree/master/sentiment-network

The cleaned data will be available after removing punctuation, emoticons and symbols and applying stop words removal. This process can be better explained in Methodology NLP pre-processing part.

2.1.2-Pretrained Word Embeddings

In addition to considering just tokens from the tweets and will make more sense by applying word embeddings to the tweets that considers the semantic relation among the tweets. These pretrained embeddings can be downloaded from link.

https://github.com/facebookresearch/fastText/blob/master/pretrained-vectors.md

I am using vec file to load the pretrained embedding's and containing vectors of vocab length with 300 dimensions and finally pre-trained embedding's will help improve the model accuracy and also account for semantic similarity.

2.2-Algorithms and Techniques

I am furnishing here the type of algorithms that will be used to solve this project: Probabilistic (NB), Linear (SVC) and SGD classifier. Let me brief on each of the algorithms below.

Naïve Bayes:

Naïve Bayes algorithm works on the probabilistic approach and is based on Bayes theorem with the assumption of independence between every pair of features and doesn't consider the interactions between features. This algorithm is fast compared to the other ones.

Support Vector Machines:

SVM is a representation of the data as points in space separated into categories by a clear gap that is as wide as possible and uses a kernel trick and is effective in high dimensional spaces and doesn't directly provide probability estimates.

This algorithm is geometric in nature and looks at the interactions between features to a certain degree and is good at capturing the interactions which aren't happening in NB.

Stochastic Gradient Descent Classifier:

SGDC is a simple and very efficient approach to fit the data and is particularly useful when the number of samples is very large. It supports different loss functions and penalties for classification and is very sensitive to feature scaling.

Probabilistic Model Naïve Bayes with TfIdf technique will form a benchmark model and is a simple that doesn't require the tuning of many hyper parameters and also fast to train the model. Here the assumption is that Decision Trees will help in improving the accuracy. DTs can learn non-linear

relationships that can't learn by linear models. Random Forrest/XGboost will help to overcome overfitting and improve the accuracy.

Moreover other than TfIdf we are using pretrained word embedings that help to create a semantic related vectors and also help to improve the model accuracy by applying the same algorithms as mentioned in the above statement.

3-Methodology

3.1-Data Pre-processing

This stage involves some text cleaning and transformation. Let's just mention the approach with the steps here. Data cleaning using NLP (Natural Language Processing) techniques with python NLTK toolkit. The steps under NLP are

Text Cleaning (removing special chars, emoticons and symbols).

Tokenization: involves tokenizing the sentence into words.

Stop words/Lemmatization: removal of stop words and involves transforming the different forms of same word to root word.

Text length filter: Consider the words that has minimum length of 2.

3.1.1-Document Term Matrix Model:

This model can represent the cleaned text in the form of dtm matrix and considers words occurrence and doesn't worry about semantic relation or word order. I have used sklearn inbuilt model called "TfIdf model' to produce this matrix. In this matrix each row represents a document (sentiment tweet) and each column represent a feature (unique word). This model stands for "term-frequency inverse document-frequency" and uses a formula to calculate the values to form a matrix. Here is the formula.

More frequent words within a document are assigned a more weight and the words that occur in many documents are assigned a lower weight.

3.1.2-Word Embeddings:

Word embedding's advantage over typical TfIdf/BoW models is providing the sematic relations between words/text that mean by taking word order into account. Each word is represented as a row in embedded matrix. Matric can be represented as each row represents a word and with the columns provides a multi-dimensional representation of a word. These embeddings can be loaded with the pre-trained models that can be downloaded from the online providers. Here the considered embedding dimension will be 300 as the length of the sentences vary from an average of 500 to 2000 in length.

1. Firstly will convert cleaned text into dtm matrix using TFIDF vectorizer with the relevant tuned parameters.
2. The embedding matrix will be created as per the mentioned steps.
    1. The pre-trained Fastext word vectors (either bin/vec file) were loaded from a disk into RAM.
    2. Create a vector (dim size=300) for each word in the vocabulary of the sentitment data and make sure that word is part of the TFIDF features.

3. Create a single vector for each sentiment tweet by averaging the vectors of all words in a sentence.

Note: Fasttest will provide a vectors for unseen words that are not present on the pre-trained vocabulary and the vocabulary has around 25 million unique words and weigh around 8-10 gb.

3.2-Implementation

3.2.1-Benchmark Implementation:

Naïve Bayes classifier algorithm was used to develop the benchmark model, which works with the concept of probabilistic approach. MultinomialNB classifier under sklearn package was used in the programming module. Before providing the cleaned data to this classifier the cleaned data (this data comes after pre-processing the text data through NLP techniques using NLTK library) has to undergo feature engineering using TfIdf technique with the default parameters that provided by this method.

3.2.2-SVM/SGDC technique with TfIdf:

To improve the accuracy/AUC score compare to the benchmark model we are going with the other models like svm/sgdc.

SVM

Support Vector classifier algorithm was used as one of the preferred algorithm to compre to benchmark, which works with the concept of decision boundaries. SVC will use the full data and solve a convex optimization problem with respect to the data points.

LinearSVC classifier under sklearn package was used in the programming module. Before providing the cleaned data to this classifier the cleaned data (this data comes after pre-processing the text data through NLP techniques using NLTK library) has to undergo feature engineering using TfIdf technique with the default parameters that provided by this method.

SGDC

Stochastic Gradient Descent classifier algorithm was used as one of the preferred algorithm to compre to benchmark, which works with the concept of gradient descent. SGDC treat the data in batches and performs a gradient descent aiming to minimize the expected loss with respect to sample distribution where assuming the data are of iid (independent and identically distributed) samples.

SGDClassifier under sklearn package was used in the programming module. Before providing the cleaned data to this classifier the cleaned data (this data comes after pre-processing the text data through NLP techniques using NLTK library) has to undergo feature engineering using TfIdf technique with the default parameters that provided by this method.

Below are the evaluation metrics chart of the mentioned algorithms (with TfIdf technique).

```
training accuracy score= 0.8636
             precision    recall  f1-score   support

   negative       0.87      0.86      0.86      2520
   positive       0.86      0.87      0.86      2480

avg / total       0.86      0.86      0.86      5000

TIME taken for Base Modeling (MultinomialNB(alpha=1.0, class_prior=None, fit_prior=True) with Tfidf)= 0:00:03.188072
training accuracy score= 0.8814
             precision    recall  f1-score   support

   negative       0.87      0.89      0.88      2413
   positive       0.89      0.87      0.88      2587

avg / total       0.88      0.88      0.88      5000

TIME taken for Base Modeling (LinearSVC(C=1.0, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.0001,
     verbose=0) with Tfidf)= 0:00:06.567822
training accuracy score= 0.8866
             precision    recall  f1-score   support

   negative       0.87      0.90      0.88      2381
   positive       0.91      0.87      0.89      2619

avg / total       0.89      0.89      0.89      5000

TIME taken for Base Modeling (SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
```

From the above metrics we can say that SGDC performed little better with 89% score as compared to the SVC with 88% and Benchmark model (NB) with 86% in terms of AUC score in the sense by considering both precision and recall score that will measure AUC.

3.2.3-SVM/SGDC technique with Tfldf and Fasttext:

Here the feature engineering has followed some steps by taking cleaned text as input.

Firstly get the vector representation of the cleaned text by having tfidf score for each vocabulary of the corpus. Secondly get the vector representation of the word that present in tfidf vocab for each sentence. Average the vector representation and use this metric to represent an each sentence in the corpus and this data should be fed into the Algotrithm for further insights.

In the word embedding we have chosen pretrained word vectors that trained on Wikipedia datasets with vocabulary of around 2.5 million words and with the embedding size of 300. Below are the evaluation metrics chart of the mentioned algorithms (with Tfldf technique and Fasttext word embedding).

```
training accuracy score= 0.82
             precision    recall  f1-score   support

   negative       0.82      0.82      0.82      2503
   positive       0.82      0.82      0.82      2497

avg / total       0.82      0.82      0.82      5000

TIME taken for Base Modeling (FastText(vocab=2519370, size=300, alpha=0.025) with Tfidf and Fasttext)= 0:00:18.569201
training accuracy score= 0.818
             precision    recall  f1-score   support

   negative       0.83      0.81      0.82      2583
   positive       0.80      0.83      0.81      2417

avg / total       0.82      0.82      0.82      5000

TIME taken for Base Modeling (FastText(vocab=2519370, size=300, alpha=0.025) with Tfidf and Fasttext)= 0:00:19.272300
```

From the above metrics we can say that SVC and SGDC (TfIdf with Fasttext) performed nothing better with 82% score as compared to the SGDC (TfIdf) with 89% in terms of AUC score in the sense by considering both precision and recall score that will measure AUC.

3.2.4- Coding Challenges/Other Complications Faced:

The first challenge I was faced was NLP pre-processing step and is the main important step in this usecase. I was confident that I did code a good pre-processing as I did went through many of the tutorials online (as mentioned the link in the project overview section) to better understand the internal mechanics.

As per considering coding challenges I haven't faced much as it is straight forward to work on if we know the proper stages well before hand to work on.

3.3-Refinement

3.3.1- SVM/SGDC technique with TfIdf:

For refinement of the existing model parameters I have used sklearn's GridSearchCV package. The b est params found for this model is
{'tfidf__min_df': 1, 'tfidf__sublinear_tf': True, 'tfidf__ngram_range': (1, 2), 'tfidf__use_idf': True} and with parameter 'C' as 10 with precision and recall as 89% , which is just little better compare to the u ntuned SVC with 88%.

One more thing noted to be with SGDC classifier is there is no nothing improvement after doing para meter tuning with GridSearchCV and got precision and recall as 89%, which is similar to the untuned SGDC model.the best params found for this model is
(loss='hinge',penalty='l2',alpha=0.0001,n_iter=10).

3.3.2- SVM/SGDC technique with TfIdf and Fasttext:

The finetuning doesn't help here much to improve the model accuracy/auc score compare to the mentioned untuned models as mentioned in the Implementation part.

3.3.3-SVM/SGDC technique with Universal Sentence Encoder (USE):

USE is much more powerful and is able to encode not only words (like word embedding) but phrases and sentences and outputs a 512-dimensional vector. Here I have chosen deep averaging network (DAN) because of the considering model complexity and resource consumption.
DAN is where input embeddings for words and bi-grams are averaged together and then passed through a feedforward deep neural network to produce sentence embeddings and the compute time is linear in the length of the input sequence.

Below are the evaluation metrics chart of the mentioned algorithms (with Universal State Encoder technique).

```
training accuracy score= 0.834
             precision    recall  f1-score   support

   negative       0.82      0.85      0.83      2458
   positive       0.85      0.82      0.83      2542

avg / total       0.83      0.83      0.83      5000

TIME taken for Base Modeling (LinearSVC(C=10, class_weight=None, dual=True, fit_intercept=True,
     intercept_scaling=1, loss='squared_hinge', max_iter=1000,
     multi_class='ovr', penalty='l2', random_state=None, tol=0.1,
     verbose=0) with Universal Sentence Encoder)= 0:00:04.955837
training accuracy score= 0.826
             precision    recall  f1-score   support

   negative       0.82      0.83      0.83      2490
   positive       0.83      0.82      0.83      2510

avg / total       0.83      0.83      0.83      5000

TIME taken for Base Modeling (SGDClassifier(alpha=0.0001, average=False, class_weight=None, epsilon=0.1,
       eta0=0.0, fit_intercept=True, l1_ratio=0.15,
       learning_rate='optimal', loss='hinge', max_iter=50, n_iter=None,
       n_jobs=1, penalty='l2', power_t=0.5, random_state=None,
       shuffle=True, tol=None, verbose=0, warm_start=False) with Universal Sentence Encoder)= 0:00:06.465512
```

From the metrics we can say that both classifiers have obtained the same precision/recall on an average with USE as feature engineering technique.

4. Results

Below are the evaluation metrics chart of the mentioned algorithms (with the above mentioned feature engineering techniques).

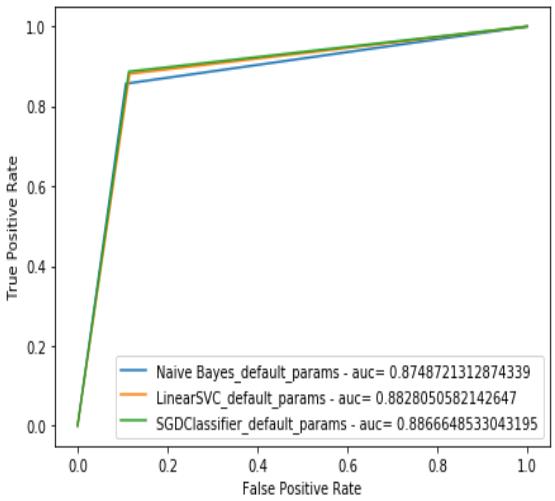
Model Name	Train Score	Test Precision	Test Recall
Naïve Bayes w/t Default TfIdf	86.64	87	87
Linear SVC w/t Default TfIdf	88.58	89	89
SGDC w/t Default TfIdf	88.98	89	89
Linear SVC w/t TfIdf & Fasttext	83.4	83	83
SGDC w/t TfIdf & Fasttext	82.04	82	82
Linear SVC w/t USE	84.12	84	84
SGDC w/t USE	83.2	84	83

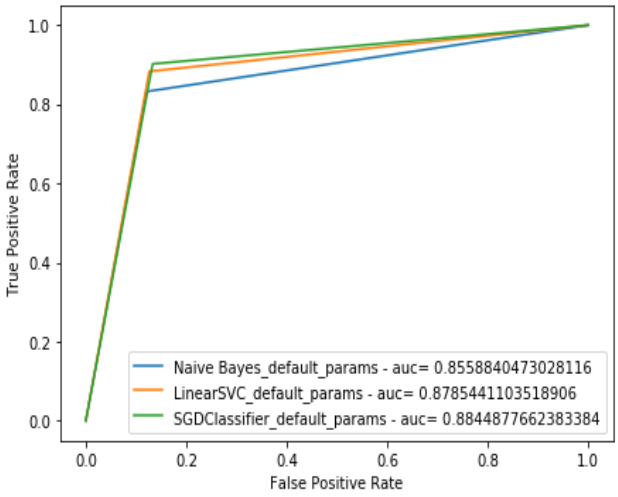
From the above models we can observe that the models haven't played a much key role compare to the feature engineering techniques. On overall SGD classifier with TfIdf technique outperforms compare to the above mentioned algos which is quite contrast to my expectation. I feel that I had coded a good NLP preprocessing techniques as I mentioned in the above steps.

4.1 – Test Robustness:

The below mentioned graphs represents the robustness of algorithms that mean the algorithm SGDC classifier and Linear SVC both are more or less good at classifying the unseen data. The graphs showed here are based on test/train split of 70-30 and 60-40 %. The auc score doesn't change much among these different splits.

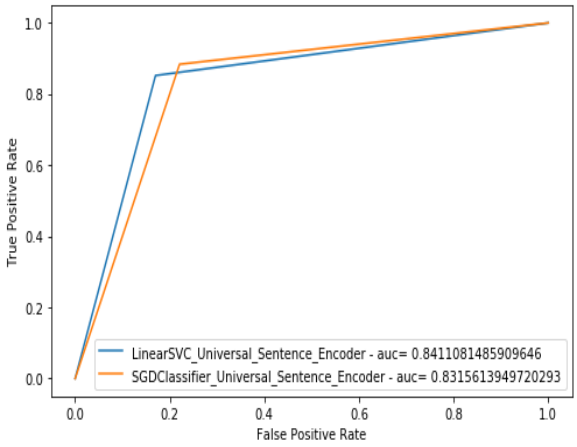
Coming to outliers I guess Feature engineering (TfIdf technique) and NLP pre-processing will take care of removing stopwords and normalizing the tokens that are very common and frequent by assigning the weights to the words.

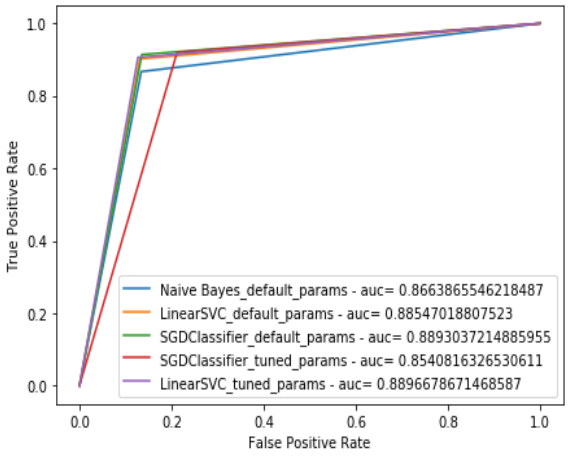


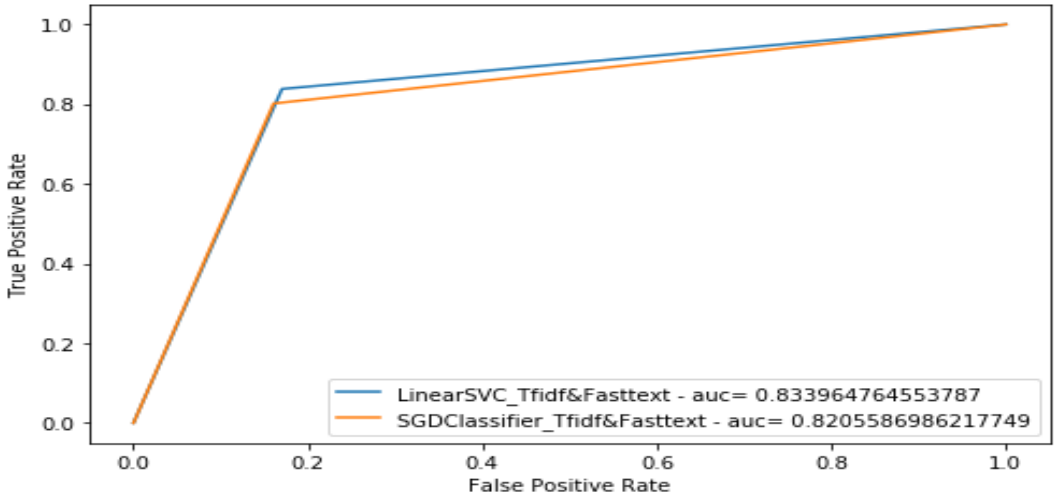


5. Conclusion

5.1- Metrics Visualization:







As provided the precision/recall metric in the previous sections. Here are the auc metric visualizatons for the mentioned algorithm and feature technique that applied. As from the graph we can observe that LinearSVC and SGD Classifer both have got same amount of auc score. AUC means area under curve score that defines how good the model is and varies between 0 and 1 as higher the value and high the model is.

Let me brief on the interesting challenge was I faced is on improving the accuracy of the model. I got a better understanding when I tried out couple of algorithms and is a need for gradient descent approach for this usecase to get the optimal loss value. None of the advanced feature engineering tehniques was suited better other than TfIdf technique to get the better AUC metrics.

I was quiet embraced after seeing the results after applying advanced feature engineering techniques (like Fasttext and USE and are based out of neural networks techniques), which were not encouraging when compare to the TfIdf technique.

Moreover I had tried other models like bagging/boosting techniques though I haven't mentioned here, which results are not encouraging as compare to the mentioned models.

What I was learnt that is stick with basics like based on the usecase firstly work on the base algorithms that suited and then go for advanced techniques but don't be on the intuition that the more advanced will provide good metrics it will but it depend on the usecase and data that we are working on.

I was still unclear on how to improve the accuracy/auc of the model more than 89%. My perception might be to build a RNN with LSTM based approach can improve but no idea on what feature engineering technique can be applied before feeding the data into RNNs.

Let me brief on my intuition why RNN might work here. Based on the previous algos/feature techniques one thing I might have missed is the order of the sequence but in RNNs this will be considered because as the ambiguity in text exists in the sense the way the person's writing review differs. Hope that's all I wanted to say.