# Sentiment Analysis of Movie Reviews

Rotten Tomatoes Dataset

Team Coders007

Atibhi Agrawal - IMT2016096
Neha Krishna Dasari - IMT2016088
Rahil Vijay - IMT2016062

# Contents


**Page**

**1 Introduction** **2**
1.1 What is sentiment analysis ? . . . 2

**2 Data** **2**
2.1 About the Dataset . . . 2
2.2 Exploratory Data Analysis . . . 2

**3 Pre-processing the Data** **7**
3.1 Data Cleaning . . . 7
3.2 Stemming . . . 7
3.3 Lemmatization . . . 7
3.4 Stopwords . . . 7

**4 Feature Engineering using tf-idf** **7**
4.1 What is tf-idf ? . . . 7
4.2 Term-frequency(tf) . . . 8
4.3 Inverse Data Frequency (idf) . . . 8
4.4 tf-idf score . . . 8
4.5 Vector Space Model . . . 8
4.6 Parameters used in Tf-idf . . . 8
4.7 Stacking tf-idf features of words and characters . . . 9
4.8 Why did we use bi-grams? . . . 9
4.9 Conclusion after using Tf-idf . . . 10

**5 Models** **10**
5.1 Logistic Regression . . . 10
5.1.1 Cross Validation . . . 11
5.1.2 Model Accuracy . . . 11
5.2 Support Vector Machines . . . 11
5.2.1 Cross Validation . . . 11
5.2.2 Model Accuracy . . . 11
5.3 Random Forest . . . 12
5.3.1 Cross Validation . . . 12
5.3.2 Model Accuracy . . . 12
5.4 AdaBoost . . . 12
5.4.1 Cross Validation . . . 12
5.4.2 Model Accuracy . . . 12
5.5 Other Approaches . . . 13

**6 Hyperparameter tuning** **13**
6.1 GridSearch . . . 13

**7 Conclusion** **14**

## Problem Statement

The Rotten Tomatoes movie review dataset is a corpus of movie reviews used for sentiment analysis, originally collected by Pang and Lee . In their work on sentiment treebanks, Socher et al they used Amazon's Mechanical Turk to create fine-grained labels for all parsed phrases in the corpus. This competition presented a chance to benchmark our sentiment-analysis ideas on the Rotten Tomatoes dataset. We were asked to label phrases on a scale of five values: negative, somewhat negative, neutral, somewhat positive, positive.

# 1 Introduction

### 1.1 What is sentiment analysis ?

Sentiment analysis is the computational task of automatically determining what feelings a writer is expressing in text. Sentiment is often framed as a binary distinction (positive vs. negative), but it can also be a more fine-grained, like identifying the specific emotion an author is expressing (like fear, joy or anger).

Sentiment analysis is used for many applications. Some examples of applications for sentiment analysis include:
Shifts in sentiment on social media have been shown to correlate with shifts in the stock market.The Obama administration used sentiment analysis to gauge public opinion to policy announcements and campaign messages ahead of 2012 presidential election. It is an integral part of market research and customer service approach.It can be useful to quickly summarize some qualities of text, especially if we have so much text that a human reader cannot analyze all of it.

# 2 Data

### 2.1 About the Dataset

The corpus consists of about 155,000 phrases.Unlike traditional test-train-split using random initialization, here we are splitting the dataset into train and validation datasets in order. Every sentence is broken down into multiple phrases,and so a random split would ensure that starkly similar phrases from the training set would land in the validation set, thereby the validation set performance misleading us into believing that the model generalized well, while all it did was encounter a validation dataset that was mostly a subset of the training dataset.

### 2.2 Exploratory Data Analysis

Exploratory Data Analysis is an approach for analyzing datasets to summarize their main characteristics usually through a series of Visualizations. It is used to gain further insight into the data. The different Visualizations we have tried are:

**Figure 1.** Number of reviews against Sentiment

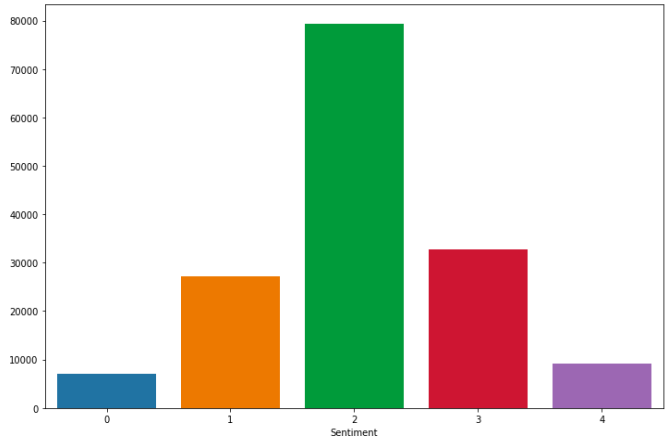
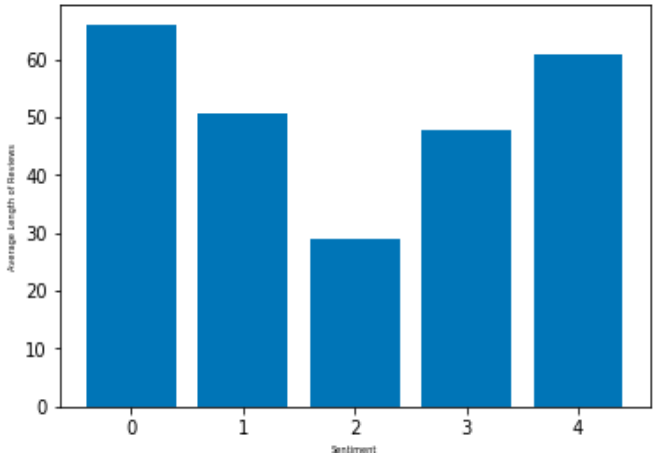



**Figure 2.** Average Length of reviews against Sentiment




We wanted to see if the length of a review affected any Sentiment in a drastic way.

**Figure 3.** WordCloud for Sentiment 0 or Negative Reviews

**Figure 4.** WordCloud for Sentiment 1 or Somewhat negative Reviews

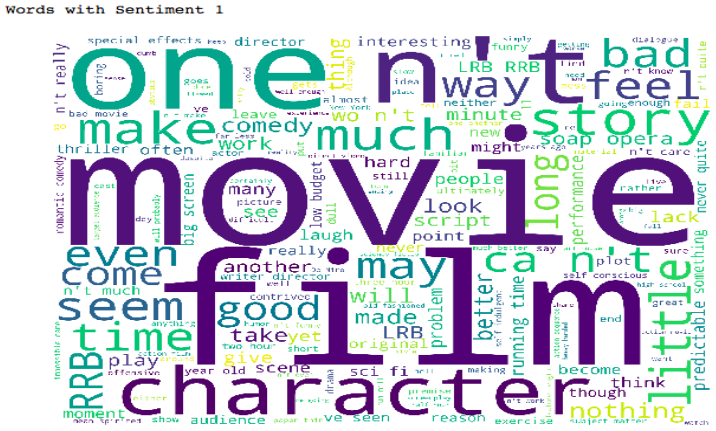



**Figure 5.** WordCloud for Sentiment 2 or Neutral Reviews

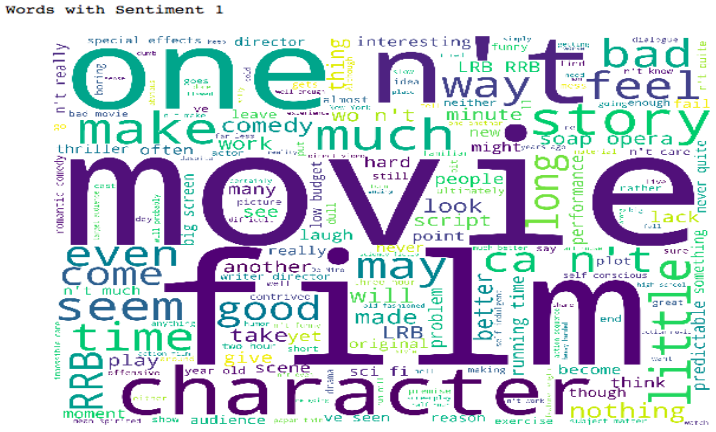



**Figure 6.** WordCloud for Sentiment 3 or Somewhat Positive Reviews

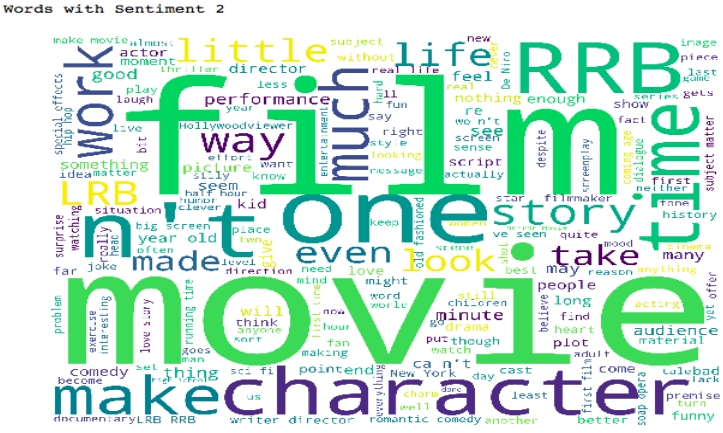
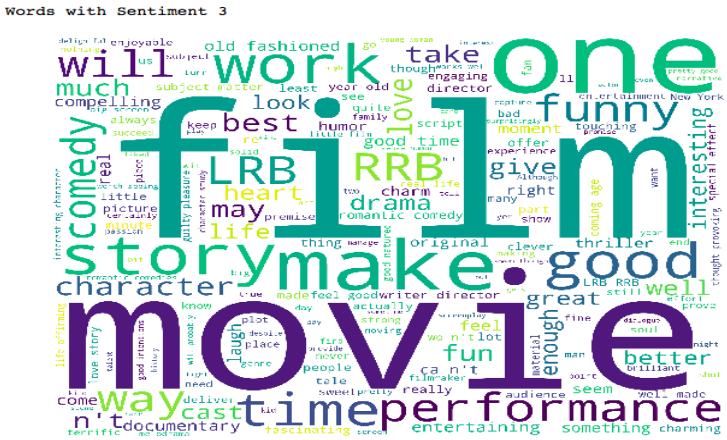
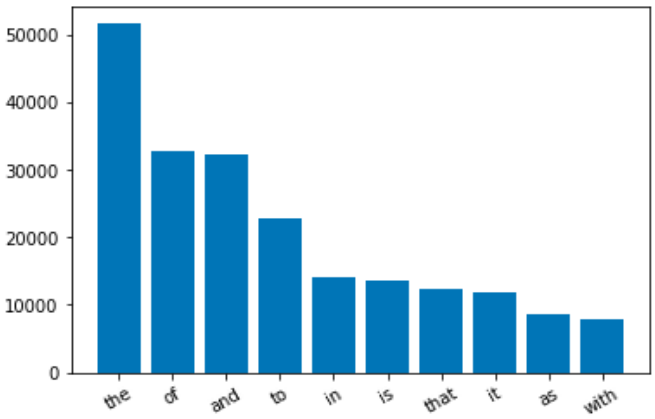


**Figure 7.** WordCloud for Sentiment 4 or Positive Reviews



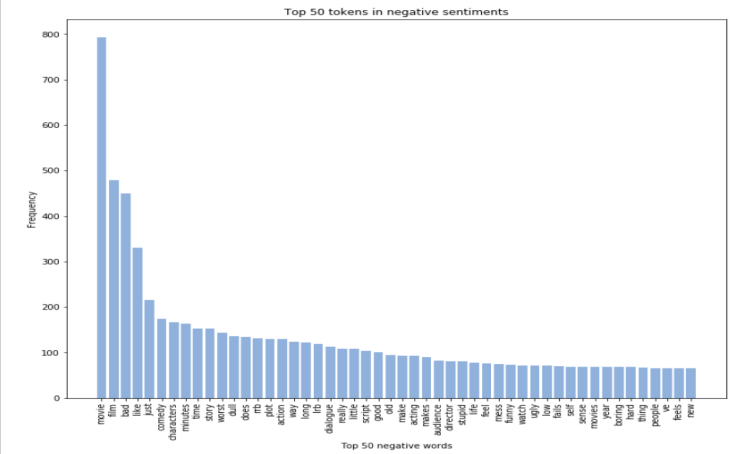

We wanted to see WordCloud representations of all phrases of each Sentiment. Words clouds are also called tag clouds and are a way to represent the content of some text visually. We infer that words like film, movie, one are most common words across all sentiments.OOP CODE (Friday)

**Figure 8.** Most frequently used words against frequencies of occurence.



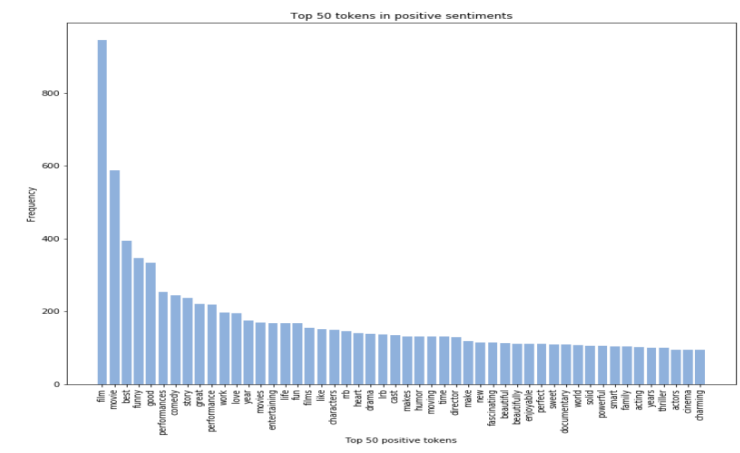

Here we see that the most frequently occurring words are articles and prepositions generally which do not really contribute a lot in deciding the Sentiment of a given Phrase.

**Figure 9.** Frequency vs Top 50 Tokens among the negative sentiment



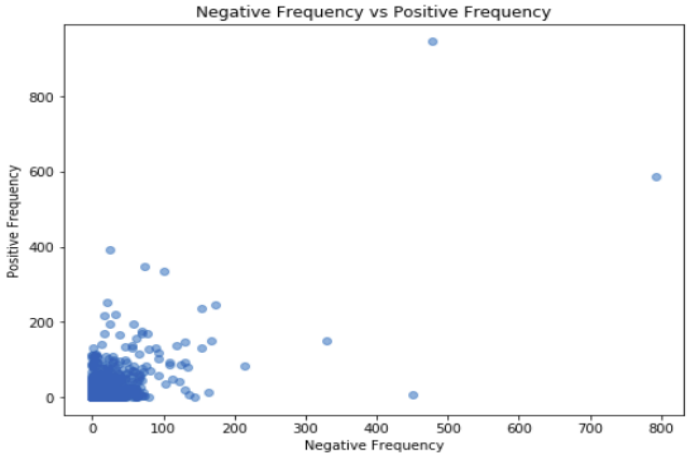

"Bad" ranks 3.

**Figure 10.** Frequency vs Top 50 Tokens among the positive sentiment




"Good" ranks 5 and "Best" ranks 3.

**Figure 11.** Negative Frequency vs Positive Frequency




We notice that in both the negative and positive plots, neutral words such as film, movie, story are quite high up in the rank.

# 3 Pre-processing the Data

## 3.1 Data Cleaning

We converted Phrases to lower case. We also removed punctuation, unnecessary characters, numbers etc. We also dropped two unnecessary columns PhraseId and SentenceId.

## 3.2 Stemming

Stemming usually refers to a crude heuristic process that chops off the ends of words in the hope of achieving this goal correctly most of the time, and often includes the removal of derivational affixes.

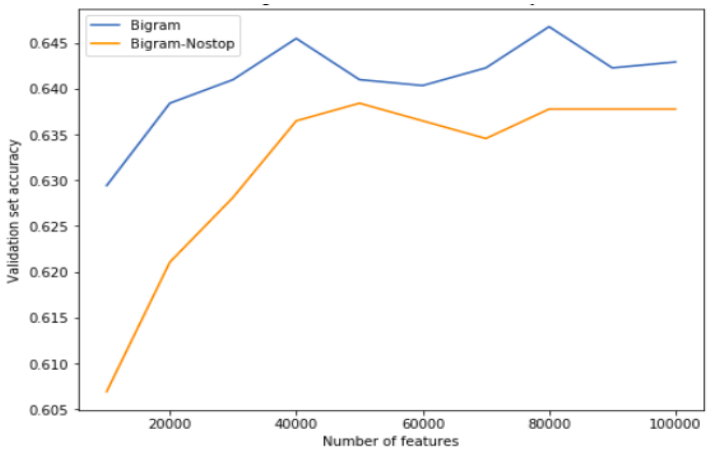
## 3.3 Lemmatization

Lemmatization usually refers to doing things properly with the use of a vocabulary and morphological analysis of words, normally aiming to remove inflectional endings only and to return the base or dictionary form of a word, which is known as the lemma . Lemmatization is used in comprehensive retrieval systems like search engines, compact indexing etc.

## 3.4 Stopwords

Text may contain stop words like 'the', 'is', 'are'. Stop words can be filtered from the text to be processed.Performance of tf-idf is better without custom stopwords. The graph below helps visualize it.

**Figure 12.** Bigram with and without stopwords. Validation Set Accuracy vs Number of features.




# 4 Feature Engineering using tf-idf

## 4.1 What is tf-idf ?

In information retrieval or text mining, the "term frequency – inverse document frequency" (also called tf-idf), is a well known method to evaluate how important is a word in a document. tf-idf is

is a very interesting way to convert the textual representation of information into a Vector Space Model (VSM), or into sparse features.

### 4.2 Term-frequency(tf)

It gives us the frequency of the word in each document in the corpus. It is the ratio of number of times the word appears in a document compared to the total number of words in that document. It increases as the number of occurrences of that word within the document increases. Each document has its own tf.

$$tf_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}}$$

### 4.3 Inverse Data Frequency (idf)

It is used to calculate the weight of rare words across all documents in the corpus. The words that occur rarely in the corpus have a high IDF score. It is given by the equation below.

$$idf(w) = \log N/df_t$$

### 4.4 tf-idf score

Combining these two we come up with the TF-IDF score (w) for a word in a document in the corpus. It is the product of tf and idf:

$$w_{i,j} = tf_{i,j} \times \log N/df_t$$

### 4.5 Vector Space Model

VSM is an algebraic model representing textual information as a vector, the components of this vector could represent the importance of a term (tf–idf) or even the absence or presence (Bag of Words) of it in a document. VSM, interpreted in a *lato sensu*, is a space where text is represented as a vector of numbers instead of its original string textual representation; the VSM represents the features extracted from the document.
Since we have a collection of documents, now represented by vectors, we can represent them as a matrix. These matrices representing the term frequencies tend to be very sparse (with majority of terms zeroed).

### 4.6 Parameters used in Tf-idf

- **analyser = 'word' or 'char'**
- **strip_accents = 'unicode'**
  Remove accents and perform other character normalization during the preprocessing step. 'ascii' is a fast method that only works on characters that have an direct ASCII mapping. 'unicode' is a slightly slower method that works on any characters.
- **encoding='utf-8'**
  If bytes or files are given to analyze, this encoding is used to decode.

- **tokeniser**
  Overrides the string tokenization step while preserving the preprocessing and n-grams generation steps. Only applies if analyzer == 'word'

- **n_gram_range (For word = (1,1), For character =(1,4)**
  We used a bigram.The lower and upper boundary of the range of n-values for different n-grams to be extracted.

- **min_df = 3**
  When building the vocabulary ignore terms that have a document frequency strictly lower than the given threshold. This value is also called cut-off in the literature. If float, the parameter represents a proportion of documents, integer absolute counts.

- **sublinear_df = True**
  Apply sublinear tf scaling, i.e. replace tf with 1 + log(tf).

### 4.7 Stacking tf-idf features of words and characters

We applied tf-idf for words as well as characters. Then we stacked the features obtained from both.

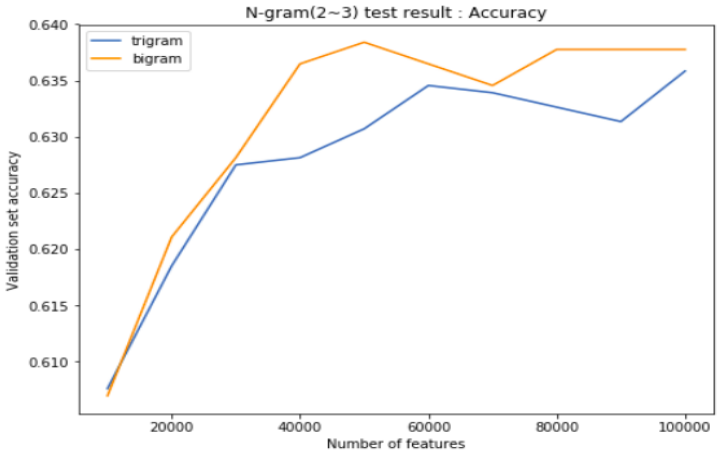
- Word Level TF-IDF
  Matrix representing tf-idf scores of every term in different documents

- Character Level TF-IDF
  Matrix representing tf-idf scores of character level n-grams in the corpus

### 4.8 Why did we use bi-grams?

N-gram Level TF-IDF : N-grams are the combination of N terms together. This Matrix representing tf-idf scores of N-grams. Bigrams are the most informative N-Gram combinations. Adding bigrams to feature set improved the accuracy of our text classification model. We tried bi-grams and tri-grams. Bi-grams were found to be the best.

**Figure 13.** Bigram and trigram plots. Validation Set Accuracy vs Number of features.



### 4.9 Conclusion after using Tf-idf

Tf-idf gives us some basic metric to extract the most descriptive terms in a document. We can compute the similarity between 2 documents using it.

It does not capture position in text, semantics, co-occurrences in different documents, etc. For this reason, TF-IDF is only useful as a lexical level feature. It also cannot capture semantics.

## 5 Models

Our data was split 85% training and 15% testing data. We used 5 fold cross-validation. Unlike traditional test_train_split using random initialization, we split the data-set into train and validation data-sets in order. Every sentence was broken down into multiple phrases, and so a random split would ensure that starkly similar phrases from the training set would land in the validation set. This was so that the validation set performance does not mislead us into believing that the model generalized well, while all it did was encounter a validation data-set that was mostly a subset of the training data-set.

### 5.1 Logistic Regression

Logistic regression is a type of probabilistic statistical classification model. Generally, it is well suited for describing and testing hypotheses about relationships between a categorical outcome variable and one or more categorical or continuous predictor variables. We define the logistic function as

$$\sigma(t) = \frac{1}{1 + e^{-t}}$$

When using logistic regression, we model the conditional probability as:

$$\sigma(\theta^T x) = \frac{1}{1 + e^{-\theta t x}}$$

Where x is the feature vector and y is the response and $\theta$ are the parameters we wish to learn. Thus we want to maximize the following:

$$\text{argmax}_\theta \sum_i \log p\left(y_i | x_i, \theta\right) - \alpha R(\theta)$$

where R($\theta$) is the regularization term,which forces the parameters to be small (for $\alpha$ greater than 0). Generally, there are two popular regularization methods which are called L1 and L2 regularizations.They take the following forms:

$$L_1 : \quad R(\theta) = \|\theta\|_1 = \sum_i |\theta_i|$$

$$L_2 : \quad R(\theta) = \|\theta\|_2^2 = \sum_i \theta_i^2$$

In our case the reason we used regularization is to avoid over-fitting by not generating high coefficients for predictors that are sparse.In general, it is believed that L1 regularization helps perform feature selection in sparse feature spaces. However, even though in our case the feature vectors are highly sparse, L1 does not perform better than L2. Using Grid Search CV for hyper-parameter tuning we got C as 5.112.

### 5.1.1 Cross Validation

[65.685373, 65.441382, 65.268, 65.3, 65.0052]

### 5.1.2 Model Accuracy

66.666

## 5.2 Support Vector Machines

A linear support vector machine is type of binary classifier which tries to separate a data set into two categories with a maximum-margin hyper-plane. For each $x \in F$ a general feature vector set, we have an associated binary classification $y \in \{-1, 1\}$. We define our training set,

$$T = \{(x_i, y_i) : x_i \in F, y_i \in \{-1, 1\}\}$$

Then a hyperplane which statisfies the points of $x \in T$ is defined as

$$w \cdot x - b = 0$$

We want all data points to fall outside of the margin of the hyperplane, so we impose the condition

$$y_i \cdot (w \cdot x_i - b) \geq 1$$

$\forall\, (x_i, y_i) \in T$ So the problem becomes

$$\arg\min_{(w,b)} \frac{1}{2}\|w\|^2$$

under the above constratint.
This type of classifier is not a natural choice for sentiment analysis, as it is binary, while our values for sentiment are not. However, we can easily extend it to suit our purposes by using a multi-class SVM.
In this approach, we create five $x$ vs. all linear SVM classifiers where $x \in \{0, \ldots, 4\}$ We change the data, to set all sentences with sentiment $x$ to 1 and all other ones to $(-1)$. This essentially gives a "reject-accept" type classifier where if we predict a sentence to have output value 1 then it has sentiment $x$ and otherwise it does not. Then we simply predict the sentiment of each sentence with all five classifiers to produce a final sentiment

### 5.2.1 Cross Validation

[65.041398 65.025399 64.904 64.668 64.689175]

### 5.2.2 Model Accuracy

65.5999739566378

### 5.3 Random Forest

The random forest classifier is a machine learning approach which builds decision trees based on random samples of the data. It is a basic and quite well understood approach for learning on large sets of data. Consider training data with N feature vectors all with dimension $k$. Then the depth of each decision tree is $k+1$ where at each step we branch based on the feature and have the last node of the tree (its final leaf) be the assigned output, in our case, the sentiment.The machine learns based on minimizing a pre-determined threshold function for the given tree. We let $F$ be the set of feature vectors and $Y$ the set of outputs.

#### 5.3.1 Cross Validation

[50.95757942, 50.95757942, 50.95757942, 50.95950438, 50.96150213]

#### 5.3.2 Model Accuracy

51.24582584125353

### 5.4 AdaBoost

AdaBoost is one of the most common ensemble algorithms. It basically combines several weak classifiers (e.g. a single split in Decision Tree) with different weights. One advantage of this algorithm is that it will improve the accuracy of weak learning models without over-fitting the data.The parameter we manipulated with AdaBoost is the number of estimators.
Given $(x_1, y_1), \ldots, (x_m, y_m)$, where $x_i \in X$ is the feature vectors and $y_i \in Y$ is the response., the Adaboost algorithm is as follows.

**Step 1** : Initialize $D_1(i) = 1/m$. For $t = 1, \ldots, T$

- Train base learner using distribution $D_t$
- Get base classifier $h_t : X \to \mathbb{R}$
- Choose $\alpha_t \in \mathbb{R}$
- Update

$$D_{t+1}(i) = \frac{D_t(i) \exp\left(-\alpha_i y_i h_t\left(x_i\right)\right)}{Z_t}$$

  where $Z_t$ is a normalization factor.

**Step 2:** Output the final classifier.
We ran the AdaBoost classifier using 50 estimators on the tf-idf feature vectors.

#### 5.4.1 Cross Validation

[54.05507498, 54.31194047, 53.7868772, 54.26488365, 54.18791794]

#### 5.4.2 Model Accuracy

54.572309273054195

### 5.5 Other Approaches

**Stacking:** Stacking is a way of combining multiple models, that introduces the concept of a meta learner. Stacking can be (and is normally) used to combine models of different kinds.The point of stacking is to explore a space of different models for the same problem.

**Ensemble:** A Machine Learning technique that combines several base models in order to produce one optimal predictive model. Among different kinds of Ensemble Methods, we tried, BAGGing. BAGGing combines Bootstrapping and Aggregation to form one ensemble model. Another method was Voting: Multiple models are created with the training set(typically of different kinds) and simple statistics are used to combine the predictions.

**XGBoost:** It is a scalable and accurate implementation of gradient boosting machines and is usually used for Classification, Regression and Ranking Problems. The accuracy score we got for XGBoost was around 54 percent.

**Light GBM:** It is a gradient boosting framework that uses tree based learning algorithm used for ranking, classification and many other machine learning tasks. Since, Light GBM is quite sensitive to overfitting, it is advised to be used on large datasets where it performs its best. Even though, Light GBM was faster than XGBoost , the accuracy scores were similar.

**CNN:** A Convolutional Neural Network is a multi layered Neural Network with a special architecture to detect complex features in data. We were not allowed to use CNN due to its Neural Network base.

## 6 Hyperparameter tuning

In machine learning, hyperparameter optimization or tuning is the problem of choosing a set of optimal hyperparameters for a learning algorithm. We used GridSearch to do hyperparameter optimization.

### 6.1 GridSearch

The traditional way of performing hyperparameter optimization has been grid search, or a parameter which is simply an exhaustive searching through a manually specified subset of the hyperparameter space of a learning algorithm. A grid search algorithm must be guided by some performance metric, typically measured by cross-validation on the training set or evaluation on a held-out validation set.
Since the parameter space of a machine learner may include real-valued or unbounded value spaces for certain parameters, manually set bounds and discretization may be necessary before applying grid search.

## 7 Conclusion

The Pickle Files are available here: PickleFiles(bit.ly/picklefiles)Out of all the models that we tried, Logistic Regression was giving us the best results with an accuracy of 66.666.

**Table 1.** Comparison of Models

| Model | Cross Validation | Accuracy Score |
|---|---|---|
| Logistic Regression | [65.685,65.441,65.268,65.3,65.005] | 66.666666 |
| Support Vector Machines | [65.041,65.025,64.904,64.668,64.689] | 65.599974 |
| Random Forest | [50.957,50.957,50.957,50.959,50.961] | 51.245825 |
| AdaBoost | [54.055,54.311,53.786,54.264,54.187] | 54.572309 |

## References


[1] Wijesinghe, Asiri. (2015). Sentiment Analysis on Movie Reviews.

[2] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher Manning, Andrew Ng and Christopher Potts,"Recursive Deep Models for Semantic Compositionality Over a Sentiment Treebank",Conference on Empirical Methods in Natural Language Processing (EMNLP 2013)

[3] Kub,Aaron P 2018,'Sentiment Analysis with Python(Part1),Towards Data Science, Medium, <https://towardsdatascience.com/sentiment-analysis-with-python-part-1-5ce197074184>

[4] 'Natural Language Processing and Sentiment Analysis with Python', Python For Engineers,<https://www.pythonforengineers.com/natural-language-processing-and-sentiment-analysis-with-python/>

[5] 'Sentiment Analysis on Trending YouTube Video Statistics and Comments', Kaggle,<https://www.kaggle.com/kanav0183/sentimental-analysis-and-detailed-eda/>

[6] Emmanuel Ameisen, P 2018,'How to Solve 90% of NLP Problems: A Step-by-Step Guide',Insight Data Science, Medium, <https://blog.insightdatascience.com/how-to-solve-90-of-nlp-problems-a-step-by-step-guide-fda605278e4e>

[7] Sebastian Raschka, P 2014,'Naive Bayes and Text Classification- Introduction and Theory',sebastianrashka.com, <http://sebastianraschka.com/Articles/2014_naive_bayes_1.html>

[8] Suyash Lakhotia, P 2018,'Predicting the sentiment of Rotten Tomatoes movie reviews using deep learning',GitHub, <https://github.com/SuyashLakhotia/RottenTomatoesCNN>

[9] Brad Salmon,P 2018,'Look Ma, No for-Loops:Array Programming with NumPy',Real Python, <https://realpython.com/numpy-array-programming/>

[10] Evan Lutins,P 2018,'Grid Searching in Machine Learning: Quick Search and Python Implementation',Medium, <https://medium.com/@elutins/grid-searching-in-machine-learning-quick-explanation-and-python-implementation-550552200596>

[11] 'Reducing Loss: Stochastic Gradient Descent',<https://developers.google.com/machine-learning/crash-course/reducing-loss/stochastic-gradient-descent>

[12] Ankhur Bhagat, P 2014, 'Data Science(Side Projects)'<http://ankoorb.blogspot.com/2014/12/sentiment-analysis-on-rotten-tomatoes.html>

[13] V. K. Singh, R. Piryani, A. Uddin, P. Waila, "Sentiment analysis of movie reviews: A new feature-based heuristic for aspect-level sentiment classification", Proc. Int. Mutli-Conf. Autom. Comput. Commun. Control Compress. Sens., pp. 712-717, Mar. 2013.