Capstone Project – 008

Amazon Fine Food Reviews Sentiment Analysis

(using Python)

By

Jayanth Sekhar Viswambhara, M13261066

MS Business Analytics, University of Cincinnati

First Reader: Dr. Dungang Liu

Second Reader: Dr. Peng Wang

Abstract

Emotions are essential to effective communication between humans, so if we want machines to handle texts in the same way, we need to teach them how to detect semantic orientation of a text as positive, neutral, or negative. That is where Sentiment Analysis or Opinion mining is used. Sentiment Analysis is a hot-trend topic of Scientific and market research in the field of Natural Language Processing (NLP), Machine Learning and Deep Learning and is widely applied to voice of the customer materials such as reviews and survey responses, online and social media for applications that range from marketing to customer service that helps the organization to make data-driven decisions.

The focus of this project is application of sentiment analysis on the reviews data collected from amazon.com on fine food products. This study used two different approaches to predict the sentiment of the review which includes vectorized features and word embedding-based features. Moreover, the conventional machine learning classifiers such as Logistic Regression with L1 and L2 Regularization, Multinomial Naïve Bayes, XGBoost, and Support Vector Machine are used for text classification. The dataset used has 568454 reviews of fine foods and contains information like product review, review score, review helpfulness, product and buyer details, and time.

The below table describes the Feature Generation and Machine Learning approaches that are implemented in the project and model performance on test dataset results:

Approach	**BoW (Unigram)**		**BoW (Bigram)**		**TF-IDF**		**Avg Word2Vec**	
	Accuracy	**F-1 score**	**Accuracy**	**F-1 score**	**Accuracy**	**F-1 score**	**Accuracy**	**F-1 score**
Logistic Regression – L1	93.564	0.878	93.740	0.874	94.080	0.883	88.816	0.772
Logistic Regression – L2	92.500	0.864	90.396	0.777	90.392	0.775	88.820	0.772
Multinomial Naïve Bayes	90.716	0.828	92.196	0.831	90.704	0.800	-	-
Random Forest	83.152	0.456	83.112	0.454	83.112	0.454	90.680	0.793
Support Vector Machine - Linear	92.436	0.855	**94.896**	**0.904**	89.012	0.744	88.616	0.766
XGBoost	94.144	0.887	93.532	0.871	93.068	0.860	93.160	0.867

Table of Contents

1. Introduction:

Customer sentiment is one of the vital aspects essential for transforming business approaches to deal with competitors in today's fast paced market. Any large company will deal with a lot of customers every day and thus it is not practically feasible to get individual feedbacks from each of them. But, with the information available through online platforms like company websites, social networking sites such as Twitter, Facebook etc. it's possible to extract customers sentiments, which can be used by the businesses to derive insights and make decisions regarding improving their product quality. In this project, we shall utilize the approach of sentiment analysis to derive individual user perceptions from the reviews data from amazon.com on fine foods. The sentiment for each review for fine foods is classified using a hybrid model which is a combination of the features transformation using Natural Language Processing (NLP) methods such as Bag of Words (Unigram and Bigram), TF-IDF, Word2Vec and the Machine Learning techniques such as Logistic Regression, Naive Bayes, Random Forest, Linear SVM, and XGBoost algorithms. Lastly, we dig further to compare the model's performance in classifying the sentiments, which provides us a bird's eye view of the user perceptions about the fine foods and also acts as a powerful feedback mechanism for amazon.com and retailers, which they could use to make immediate corrections or utilize this for improving the design of their subsequent models.

1.1 Sentiment Analysis:

So, what exactly is sentiment? Sentiment relates to the meaning of a word or sequence of words and is usually associated with an opinion or emotion. And analysis? Well, this is the process of analyzing people's opinions or sentiments towards entities such as products, services, organizations, individuals, issues, events, topics, and their attributes and making inferences; in this case, using machine learning to learn and predict whether a review is positive or negative.

Although linguistics and natural language processing (NLP) have a long history, little research had been done about people's opinions and sentiments before the year 2000. Since then, with the explosive growth of social media and digital applications (e.g., reviews, forum discussions, blogs, Twitter, comments and postings on social networking sites) on the Web, the field has become a very active research area. Individuals and Organizations are increasingly using the content in these media for decision making. Now-a-days, if one wants to buy a consumer product, one is no longer limited to asking one's friends and family for opinions because there are many reviews in the digital application and discussions in public forums on the Web about the product.

For an Organization, it may no longer be necessary to conduct surveys, opinion polls and focus groups to gather public opinions because there is abundance of such information publicly available. However, finding and monitoring opinion sites on the Web and distilling the information contained in them remains a formidable task as each site contains a huge volume of opinion text that is not always easy to decipher and summarize the sentiment. Automated sentiment analysis systems are thus needed.

Sentiment Analysis applications have spread to almost every possible domain, from consumer products, services, healthcare, and financial services to social events and political elections. Many big corporations have also built their own in-house capabilities, e.g., Microsoft, Google, Facebook, Amazon, Hewlett-Packard etc., These practical applications and industrial interests have provided strong motivations for research in Sentiment Analysis.

2. Objective:

To predict the sentiment of the customers towards fine foods using the customer reviews data extracted from amazon.com. In this Project, we are going to perform sentiment analysis by following the below mentioned sequence of steps.

1. Data Cleaning (Preprocessing)
2. Exploratory Data Analysis
3. Feature Generation (transforming the words to vectors)
4. Modelling (Classification)
5. Comparison of Results using evaluation metrics

3. Overview of the Data:

This is the first step in any data analysis project. We need to completely understand the data even before we implement machine learning models on the data.

3.1 Data Source:

In our current analysis, I will be using the amazon fine food reviews dataset by Stanford Network Analysis Project (SNAP). Source Link: https://snap.stanford.edu/data/web-FineFoods.html

The data span a period of more than 10 years, including 568,454 reviews for 74,258 products by 256,059 users up to October 2012. Reviews include product and user information, ratings, and a plaintext review. We also have reviews from all other Amazon categories.

3.2 Data Definition:

The dataset named 'Reviews.csv' consists of 568,454 reviews and 10 columns. Metadata for with Column descriptions are as listed below:

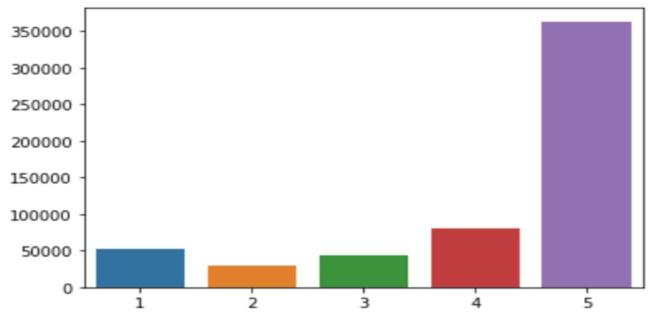
- **Id**.
- **ProductId:** unique identifier for the product.
- **UserId:** unique identifier for the user.
- **ProfileName:** Profile Name of the user
- **HelpfulnessNumerator:** Number of users who found the review helpful.
- **HelpfulnessDenominator:** Number of users who indicated whether they found the review helpful or not.
- **Score:** Rating by the user to a product on a scale of 1 to 5.

- **Time:** Timestamp for the review.
- **Summary:** Brief summary of the review by the user to a product.
- **review:** Text of the review by the user to a product.

3.3 Data Cleaning & Preprocessing:

Basic Cleaning:

The Data obtained is structured and do not have missing values except for the columns ProfileName and Summary, which we do not require in our analysis. Before cleaning, let us understand the data. Let us now look at the distribution of reviews by score.



We can observe that the data is imbalanced as there are more positive reviews (reviews with score > 3) than negative (reviews with score < 3) and neutral reviews (reviews with score = 3).
For our analysis, we will focus only on positive and negative reviews and ignore neutral reviews, which makes it a binary classification problem.
Also, we removed two reviews which have HelpfulNumerator greater than HelpfulDenominator, which does not make any sense.
For our analysis, we will focus only on positive and negative reviews and ignore neutral reviews, which makes it a binary classification problem.

Data Preprocessing:

Why do you need to preprocess this text? – Not all the information is useful for classification of the sentiment. The way the language is written, it contains lot of information which is grammar specific. Thus, when converting to numeric format, word specific characteristic like capitalization, punctuations, symbols, special characters, suffixes/prefixes etc., are redundant and noisy data. Cleaning the data in a way that similar words map to single word and removing the grammar relevant information from text can tremendously reduce the vocabulary and thus reducing the input dimension to the model.

The following steps are followed to preprocessing the data.

- **Removing HTML tags and URLs**

The reviews contain lot of HTML tags and URLs which does not add any value for classification. The HTML tags and URLs add noise to the data and can interrupt the next steps if we do not remove them first. These can be removed using the Regex module in python and replace them with an empty space.

- **Expanding Contractions**

Contractions are the shortened/combined form of commonly used words, which are easier for humans to understand. But, for machines expanding the contractions allows them to handle the text data efficiently. Eg: I'm, after expanding becomes, I am

- **Removing Punctuations, Numbers and Special Characters**

As discussed, punctuations, numbers and special characters do not help much. It is better to remove them from the text just as we removed HTML tags and URLs.

- **Converting to Lower case**

Since the semantics of a words or a phase do not depend upon which case the word is written in, the uppercase words are converted to lowercase to avoid duplication of the words. For example, the words 'DOG', 'dog', and 'Dog' convey the same meaning. This helps in reducing the dimensionality of the feature set.

- **Removing stop words**

Stop words are the words which are commonly used and removed from the sentence as pre-step in different NLP tasks. Examples of stop words are: 'a', 'the', 'an', 'this' etc. For doing 'NLTK' module of python is used in the study to remove stop words.

- **Lemmatization**

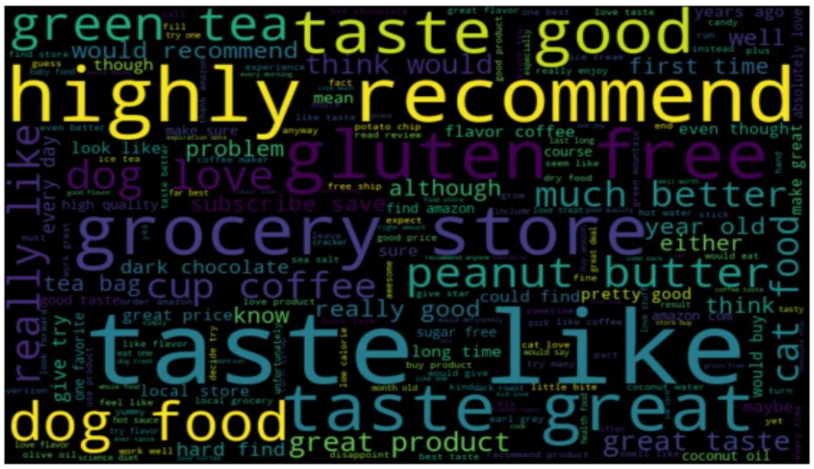
As the first step of Lemmatization, it is required to perform tokenization of words i.e., splitting the sentence into array of words, which helps in transforming each word separately. When it comes to Lemmatization, it is the process that reduces the words to its' root word, by utilizing the grammar rules and dictionary for mapping words to root form. This is also useful to remove redundancy in the vocabulary. To perform this action, WordNetLemmatizer from nltk module in python is utilized.

3.4 Exploratory Data Analysis:

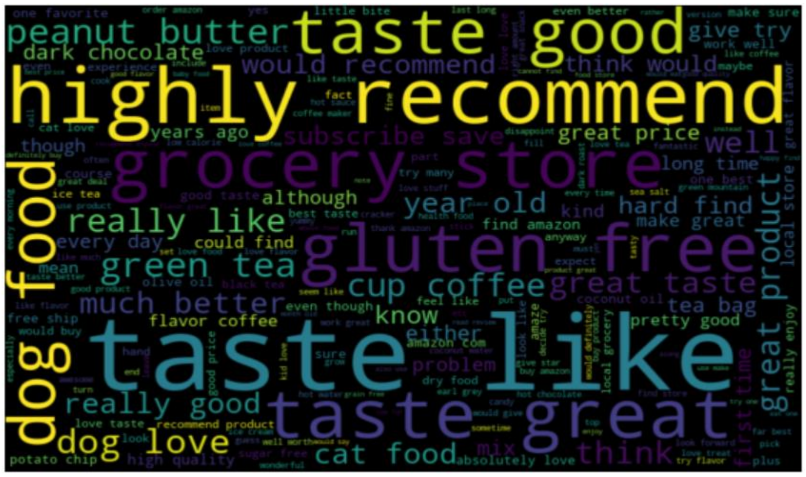
In this section, we will explore the cleaned reviews text as exploring and visualizing data is an essential step to gain insights about the data.

Understanding the common words used in the reviews using WordCloud

A WordCloud is a visualization wherein the most frequent words appear in large size and the less frequent words appear in smaller sizes. Let's us visualize the common words in all the reviews.

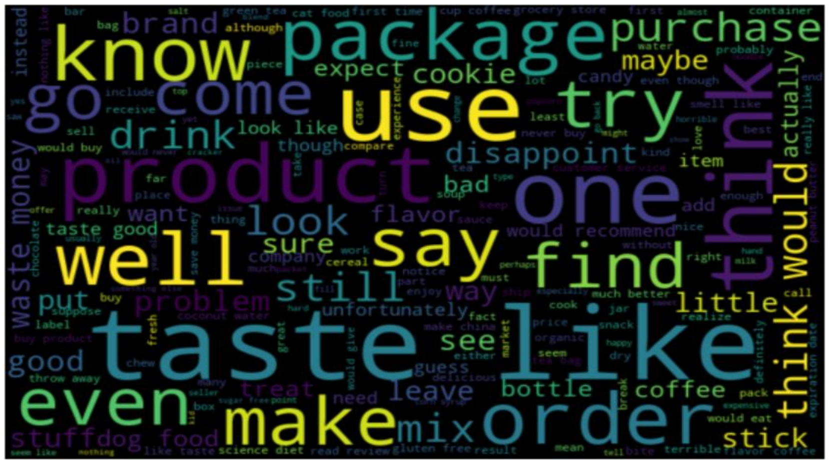


Understanding the common words used in the positive reviews using WordCloud



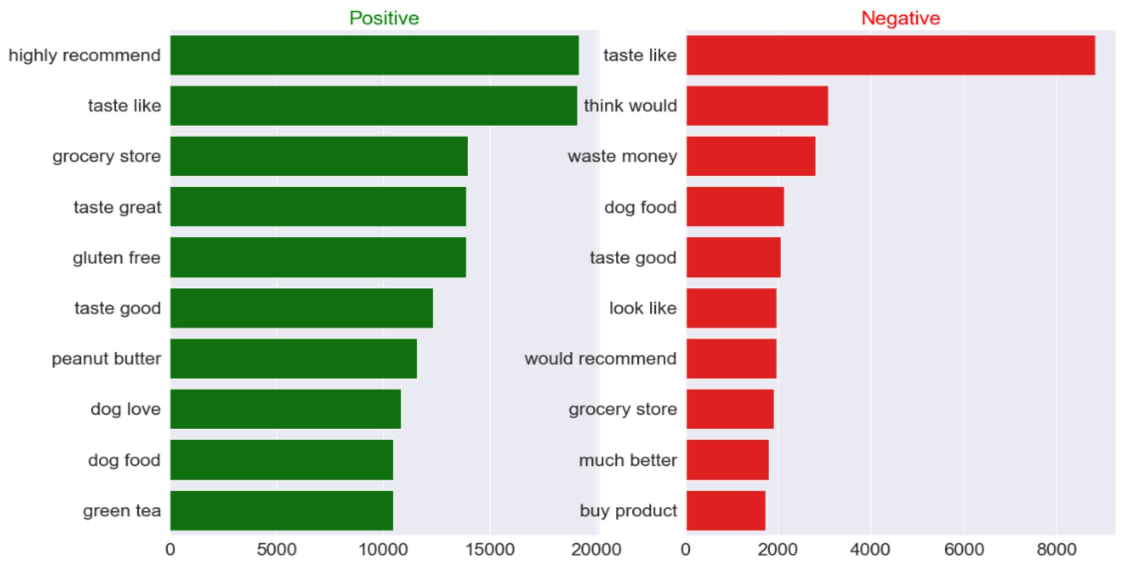
From the above, we can see that the common words in all the reviews and positive reviews are mostly similar. This makes sense as there are more positive reviews in the distribution of reviews. Words like ‘highly’, ‘recommend’, ‘good’, ‘like’, ‘taste’, ‘gluten’, ’food’ has high frequency.

Understanding the common words used in the negative reviews using WordCloud



We can observe that words like ‘use’, ’taste’, ‘say’, ‘well’, ‘know’, ‘make’ has high frequency of occurrence in negative reviews. But, these words doesn’t relate to negative in meaning.

Top 10 common words in the positive and negative reviews – Bigrams



From the above plot, we can see that two bigrams such as ‘highly recommend’, ‘taste great’ makes sense for a positive context. Whereas, only one bigram, ‘waste money’ makes sense for a negative context.

4. Modelling – Sentiment Analysis:

In this section, 3 different approaches are used to perform the review classification task with each of them combined with 6 different classification algorithms. This will provide us with 16 combinations of the result. This section discusses each approach in detail. Moreover, each approach contains multiple set of features which are also compared in terms of accuracy.

4.1 Feature Generation

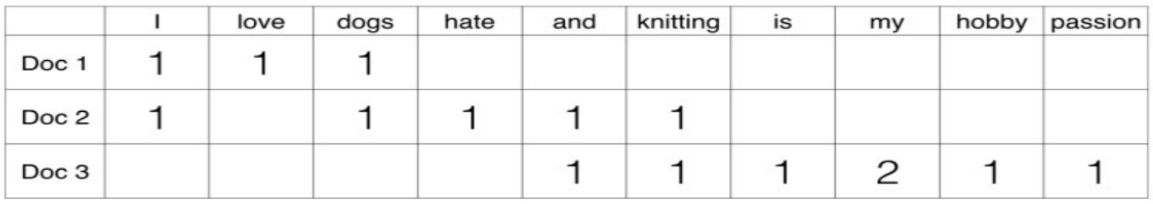
For this text data to make sense to out machine learning algorithm we will need to convert each review to a numerical representation, which is called Vectorization. Let us discuss the methods used to build features from the reviews data, by transforming the list of reviews containing words or tokens to vectors, which is used as an input data in the Machine Learning classifiers.

4.1.1 Bag of Words (BoW)

Bag of Words (BoW) is the simplest way of extracting features from the text data, reviews in this case. BoW converts text into the matrix of occurrence of words within a document. This model concerns about whether given words occurred or not in the document and ignores grammar and order of words. This model utilizes the Count Vectorizer to count the appearance of words in each document.

Example: There are three documents:

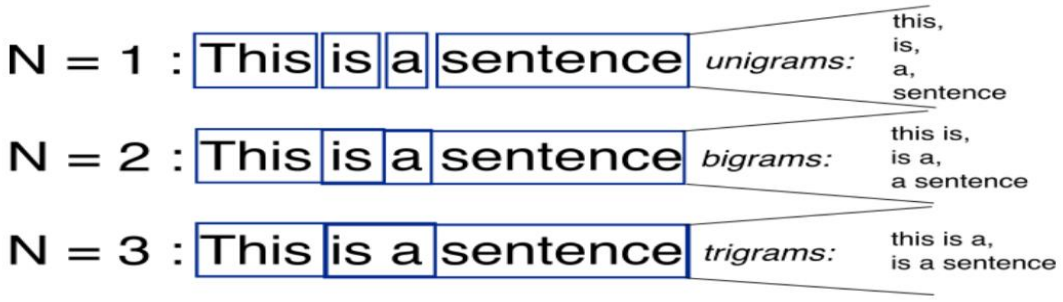
Doc 1: I love dogs. Doc 2: I hate dogs and knitting. Doc 3: Knitting is my hobby and passion.



	I	love	dogs	hate	and	knitting	is	my	hobby	passion
Doc 1	1	1	1							
Doc 2	1		1	1	1	1				
Doc 3					1	1	1	2	1	1

N-gram:

According to Wikipedia, n-gram is a continuous sequence of n items from a given sequence of text. In other words, n-grams are simply all combinations of adjacent words or letters of length n that you can find in your source text. Below picture represents well how n-grams are constructed out of source text.



In this project, I will extend the Bag of Words to **Unigram** and **Bigrams** and see how it affects the performance.

4.1.2 TF-IDF

TF-IDF stands for Term Frequency-Inverse Document Frequency and is a statistic that aims to better define how important a word is for a document, while also taking into account the relation to other documents from the same corpus. This is performed by looking at how many times a word appears into a document while also paying attention to how many times the same word appears in other documents in the corpus.

TF-IDF is a score which is applied to every word in every document in our dataset. And for every word, the TF-IDF value increases with every appearance of the word in a document but is gradually decreased with every appearance in other documents.

Formula to calculate TF-IDF Score:

Term Frequency Formula:

$$Tf = \frac{number\ of\ times\ term\ appears\ in\ document}{total\ number\ of\ words\ in\ document}$$

Inverse Document Frequency Formula:

$$Idf = \ln\left(\frac{total\ number\ of\ documents}{number\ of\ documents\ with\ term\ in\ them}\right)$$

Here $f(w,D)$ is the frequency of word w in the corpus

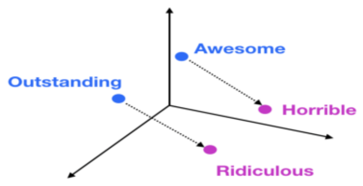
TF-IDF Score Formula:

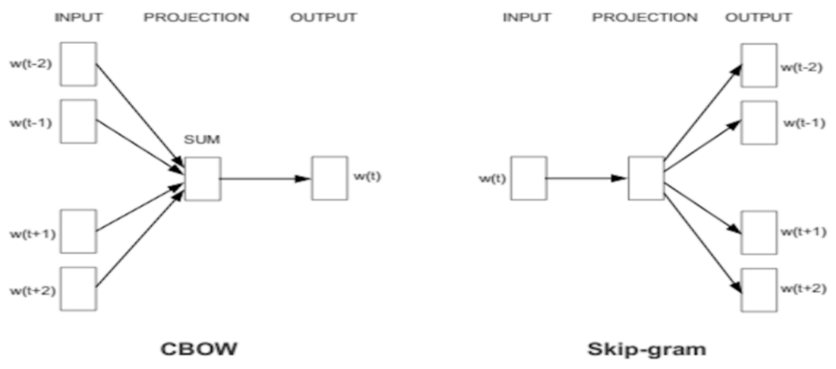
$$TfIdf = Tf \times Idf$$

4.1.3 Word2Vec

Word embeddings are words mapped to real number vectors such that it can capture the semantic meaning of words. BoW and TF-IDF vectorization methods do not capture the meaning between the words, they consider the words separately as features. Word embeddings use some models to map a word into vectors such that similar words will be closer to each other. As shown in the below figure, for example some of the positive words which are adjectives will be closer to each other and vice versa for negative adjectives. It captures semantical and syntactical information of words. To train this model it takes into consideration the words surrounding that word of particular window size. There are different ways of deriving the word embedding vectors. Word2vec is one such method where neural embeddings model is used to learn that. It uses following two architectures to achieve this.

- Continuous Bag of Words (CBOW)
- Skip Gram





In this project, we will be using genism's Word2Vec for creating the model.

4.2 Model Validation

This section discusses model validation using the train and test sets and sheds light on the model performance measurement to perform model comparison.
In this project, we split the original dataset into train and test sets in the ratio of 3:1. But, before splitting the dataset, we considered only the latest 100,000 reviews for our analysis due to computational constraints. The train and test datasets contain 75,000 and 25,000 reviews, respectively. The models are trained using the samples of train set and its performance is measured by fitting the trained model on the unseen samples of the test set.

4.3 Evaluation Metrics

An Evaluation metric quantifies the performance of a predictive model. Standard metrics work well but does not always gives the optimal performance. Therefore, an evaluation metric must be chosen that best captures the performance of a classifying model.
For this project, Accuracy and F-1 score calculated using the confusion matrix are the evaluation metrics used to evaluate the model performance as the classes are imbalanced.

Confusion Matrix table

Actual class	**Predicted as positive**	**Predicted as negative**
Positive	True positive (tp)	False negative (fn)
Negative	False positive (fp)	True negative (tn)

Accuracy: It is the proportion of true results among the total number of cases examined.

F-1 Score: It is a number between 0 and 1 and is the harmonic mean of precision and recall.

$$F_1 = 2 * \frac{precision * recall}{precision + recall}$$

Where, precision is the proportion of predicted positives that is truly positive, and recall is the proportion of actual positives that is classified correctly.

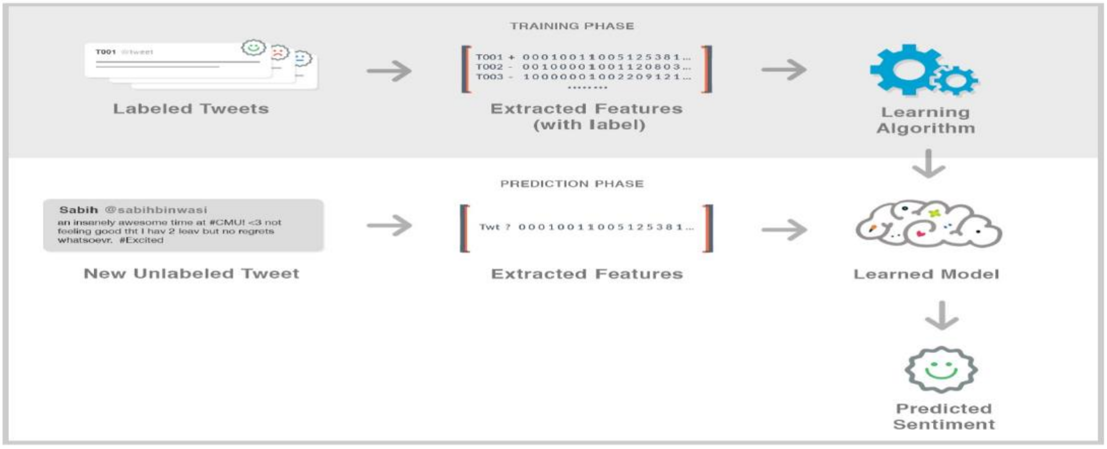
4.4 Hyper Parameter tuning

Hyper parameter tuning can be performed using several approaches like Grid Search and Randomized Search cross validation approaches being the most frequently used. For this project we implemented Randomized Search cross validation as it is computation friendly. Hyper parameters to be tuned differs based on the classifier algorithm in use. The hyper parameter that are tuned for each algorithm is detailed below:

Algorithm	Hyperparameters
Logistic Regression using L1 & L2 penalty	C - Regularization parameter
Multinomial Naïve Bayes	Alpha - Additive smoothing parameter
Random Forest	n_estimators – No. of sequential decision trees max_depth – maximum depth of a tree
Linear kernel Support Vector Machine	Alpha – constant that multiplies the regularization term.
XGBoost	n_estimators – No. of sequential decision trees max_depth – maximum depth of a tree eta – learning rate

4.5 Machine Learning - Classification

Once we built the feature vector using BoW, TF-IDF, Word2Vec as described above for each instance in the dataset, these vectors are fed to the classifier to perform sentiment classification. The Machine Learning techniques implemented in the project for classification are detailed below:


TRAINING PHASE
T001 @tweet
Labeled Tweets
T001 + 00010011005125381...
T002 - 00100001001120803...
T003 - 10000001002209121...
Extracted Features (with label)
Learning Algorithm
PREDICTION PHASE
Sabih @sabihbinwasi
an insanely awesome time at #CMU! <3 not feeling good tht I hav 2 leav but no regrets whatsoevr. #Excited
New Unlabeled Tweet
Twt ? 00010011005125381...
Extracted Features
Learned Model
Predicted Sentiment

4.5.1 Logistic Regression

Logistic Regression is a regression-based linear classifier which is used to model discrete response variables. The coefficients of the regression model are obtained by optimizing a conditional likelihood function. In this model, the posterior probability of a class given a document is mathematically formulated as below given the set of coefficients A and b:

$$P(C_i|X_j) = \frac{e^{(A \cdot X_i + b)}}{1 + e^{(A \cdot X_i + b)}}$$

The above posterior probability can be transformed using a logistic function and represented by the linear combination of features. This helps in conceiving the problem in terms of simple linear regression.

For this project, we will be implementing Logistic Regression models with L1 and L2 Regularization techniques.

Ridge Regression (L2 Regularization):

L2-norm loss function is also known as least squares error (LSE).

w* = minimization of ∑i=1n[log(1+exp(-zi))] + **λ*∑ (wj) ^{2}**

∑ (wj) ^{2} is a regularization term and ∑ [log(1+exp(-zi))] is the Loss term. λ is a hyper parameter.

We added the regularization term (i.e. squared magnitude) to the loss term to make sure that the model does not undergo overfit problem. Here we will minimize both the Loss term and the regularization term. If hyper parameter(**λ**) is 0 then there is no regularization term, then it will overfit and if hyper parameter(**λ**) is very large then it will add too much weight which leads to underfit.

Results:

The below table shows the results obtained using Logistic Regression with L2 penalty norm model's performance of predicting the sentiment on the unseen test set using optimal regularization parameter, C.

Approach	Hyperparameter, C	Accuracy (%)	F-1 Score
BoW (Unigram)	0.016683597	**92.500**	**0.864**
BoW (Bigram)	0.840365851	90.396	0.777
TF-IDF	0.939348647	90.392	0.775
Avg Word2Vec	0.466359076	88.82	0.772

From the results, we can see that the logistic regression with L2 penalty norm model performed well with BoW (Unigram) feature generation.

Lasso Regression (L1 Regularization):

L1-norm loss function is also known as least absolute deviations (LAD), least absolute errors (LAE).

$$w^* = \text{argmin} \sum[\log(1+\exp(-z_i))] + \boldsymbol{\lambda^* \, ||w||1}$$

Here the L1 norm term will also avoid the model to undergo overfit problem. The advantage of using L1 regularization is **Sparsity.**

Results:

The below table shows the results obtained using Logistic Regression with L1 penalty norm model's performance of predicting the sentiment on the unseen test set using optimal regularization parameter, C.

Approach	Hyperparameter, C	Accuracy (%)	F-1 Score
BoW (Unigram)	0.025450689	93.564	0.878
BoW (Bigram)	0.989505461	93.740	0.874
TF-IDF	0.969643705	**94.080**	**0.883**
Avg Word2Vec	0.971373921	88.816	0.772

From the results, we can see that the logistic regression with L1 penalty norm model performed well with TF-IDF feature generation.

4.5.2 Multinomial Naïve Bayes

This model comes from the family of probabilistic classifiers which are based on applying the Bayes' theorem with the naïve assumption of independence between the features. In this model, the document is represented as a bag of words or TF-IDF numerical representation and then the document in each class is modeled as samples drawn from a multinomial word distribution. The conditional probability of a document given a class is then calculated by multiplying the probability of each observed word in the corresponding class. Finally, the Bayes' rule is applied to calculate the posterior probability of a class given a document and the class with the highest posterior probability is assigned to the document. The posterior probability can be mathematically formulated as shown below, assuming there are m unique classes and a document is represented by n-dimensional term vector:

$$P(C_i|D) \propto P(C_i) \cdot \prod_{t=1}^{n} P(x_t|C_i)$$

Where, i = 1,2,3, …., m and t=1,2,3, …., n.

Results:

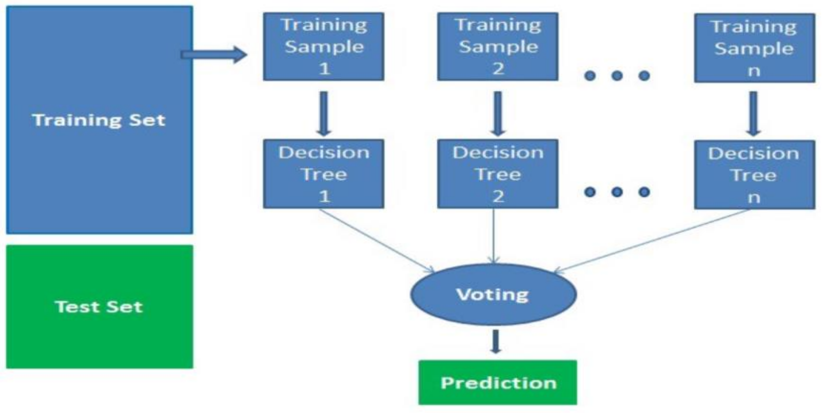
The below table shows the results obtained using Multinomial Naïve Bayes model's performance of predicting the sentiment on the unseen test set using optimal hyperparameter, Alpha.

Approach	Hyperparameter, Alpha	Accuracy (%)	F-1 Score
BoW (Unigram)	0.001	90.716	0.828
BoW (Bigram)	0.870415365	**92.196**	**0.831**
TF-IDF	0.018213607	90.704	0.800

From the results, we can see that the Multinomial Naïve Bayes model performed well with BoW (Bigram) feature generation.

4.5.3 Random Forest

A random forest is a series of decision trees in which the leaf nodes indicate the predicted class. Each decision tree incorporates a selection of features and outputs a decision at the end. These results are then combined from all the decision trees to give the final sentiment prediction. The diagram below visualizes the process quite nicely.



Results:

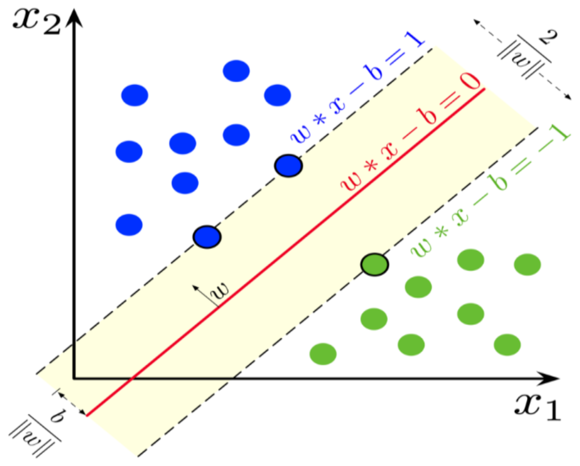
The below table shows the results obtained using Random Forest model's performance of predicting the sentiment on the unseen test set using optimal hyperparameters, n_estimators and max_depth.

Approach	Hyperparameters		Accuracy (%)	F-1 Score
	n_estimators	max_depth		
BoW (Unigram)	400	12	83.152	0.456
BoW (Bigram)	50	11	83.112	0.454
TF-IDF	200	13	83.112	0.454
Avg Word2Vec	100	13	**90.680**	**0.793**

From the results, we can see that the Random Forest model performed well with Avg Word2Vec word embedding feature generation method.

4.5.4 Support Vector Machines (SVM) – Linear Model

SVM is a supervised machine learning algorithm mostly used for classification but can also be used for regression. The Main idea is that based on the labeled data the algorithm tries to find the optimal hyperplane which can be used to classify new data points. In two dimensions, the hyperplane is a simple line. Intuitively the best line is the line that is far away with the largest margin from all the classes.



Results:

The below table shows the results obtained using Linear kernel SVM using SGD Classifier model’s performance of predicting the sentiment on the unseen test set using optimal hyperparameter, Alpha.

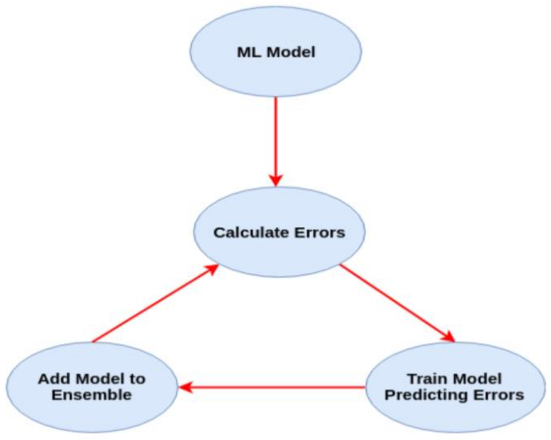
Approach	Hyperparameter, Alpha	Accuracy (%)	F-1 Score
BoW (Unigram)	0.001	92.436	0.855
BoW (Bigram)	0.001	**94.896**	**0.904**

TF-IDF	0.001	89.012	0.744
Avg Word2Vec	0.001	88.616	0.766

From the results, we can see that the Linear kernel SVM model performed well with BoW (Bigram) feature generation method.

4.5.5 XGBoost

XGBoost is a decision-tree-based ensemble Machine Learning algorithm that uses a gradient boosting framework. XGBoost and Gradient Boosting Machines are both ensemble tree methods that apply the principle of boosting weak learners using gradient descent architecture. However, XGBoost improves upon the base GBM framework through systems optimizations and algorithmic enhancements through parallel processing, tree-pruning, handling missing values and regularization to avoid overfitting/bias.



Results:

The below table shows the results obtained using XGBoost model's performance of predicting the sentiment on the unseen test set using optimal hyperparameters,n_estimators, max_depth and eta.

Approach	Hyperparameters			Accuracy (%)	F-1 Score
	n_estimators	max_depth	eta		
BoW (Unigram)	200	12	0.2	**94.144**	**0.887**
BoW (Bigram)	150	9	0.2	93.532	0.871
TF-IDF	150	9	0.2	93.068	0.860
Avg Word2Vec	200	11	0.2	93.160	0.867

From the results, we can see that the XGBoost model performed well with BoW (Unigram) feature generation method.

5. Conclusion

Throughout this project, we have built 16 models for sentiment classification, based on various feature generation and machine learning methods.
Let us compare the overall model performance results on the unseen test dataset obtained to identify the best model using F-1 score and Accuracy in that order as evaluation metrics.

Approach	**BoW (Unigram)**		**BoW (Bigram)**		**TF-IDF**		**Avg Word2Vec**	
	Accuracy	**F-1 score**	**Accuracy**	**F-1 score**	**Accuracy**	**F-1 score**	**Accuracy**	**F-1 score**
Logistic Regression – L1	93.564	0.878	93.740	0.874	94.080	0.883	88.816	0.772
Logistic Regression – L2	92.500	0.864	90.396	0.777	90.392	0.775	88.820	0.772
Multinomial Naïve Bayes	90.716	0.828	92.196	0.831	90.704	0.800	-	-
Random Forest	83.152	0.456	83.112	0.454	83.112	0.454	90.680	0.793
Support Vector Machine - Linear	92.436	0.855	**94.896**	**0.904**	89.012	0.744	88.616	0.766
XGBoost	94.144	0.887	93.532	0.871	93.068	0.860	93.160	0.867

By comparing the model performance using the above evaluation metrics, Support Vector Machine with linear kernel with Bag of Words – Bigram feature generation has the best performance on the test dataset.

6. Future Scope

There are plenty of ways to expand on the work done in this project. Firstly, we can also try TF-IDF Word2Vec feature generation approach. Also, other supervised machine learning algorithms like Kernel SVM, Gradient Boosting, Ada Boosting can also be implemented.

Another approach is using both the review text and review summary text for feature extraction process. In addition to Sentiment Analysis, Topic modelling can also be performed.

We can also leverage Deep Learning methods like LSTMs (Long Short-Term Memory) to extend the distance of relationships among words in the reviews which could result in better accuracy in sentiment classification but the prediction might not have the advantage of the ease of understandability.

7. References

i. https://www.analyticsvidhya.com/blog/2018/07/hands-on-sentiment-analysis-dataset-python/

ii. https://neptune.ai/blog/exploratory-data-analysis-natural-language-processing-tools

iii. https://stackoverflow.com/questions/19790188/expanding-english-language-contractions-in-python

iv. https://towardsdatascience.com/sentiment-analysis-a-how-to-guide-with-movie-reviews-9ae335e6bcb2

v. https://gist.github.com/aparrish/2f562e3737544cf29aaf1af30362f469

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