# Yelp Review Sentiment Analysis with NLTK

Biman S. Mondal
August 13, 2025

## Summary

The goal of this project is to perform sentiment analysis on Yelp reviews dataset, specifically to determine if a given review opinion is “positive”, “negative”, or “neutral”. The dataset consists of two .csv files: training.csv with 650k records and test.csv with 50k records. Logistic regression, SVM, Naïve Bayes, and Random Forest machine learning models were created to perform multinomial classification. Additionally, a VADER pre-trained model was used as an “automatic” sentiment analyzer to compare against the classifier models. A comparison of the results was performed to show that logistic regression classification model performed the best outcome with an accuracy of nearly 70%.

## Introduction

Natural language processing (NLP) is the method by which computers make sense of written text. Sentiment analysis, a subfield of NLP, involves predicting the tone of a text. There are many reasons to understand the opinion expressed in text. Sentiment analysis can be useful for businesses trying to understand customer feedback, for a company launching a new product, or for determining the underlying emotion behind a stock. This project aims to apply sentiment analysis to a specific dataset and explore the nuances of its implementation.

The Yelp reviews dataset consists of reviews of businesses scraped from 2015 and developed for the Yelp Dataset Challenge. The full dataset can be found on Yelp; the subset of the original data used for this project was sourced from HuggingFace. The dataset includes the number of stars and a text review. This project assumed that one or two stars means “negative”, three stars means “neutral”, and four and five stars means “positive”.

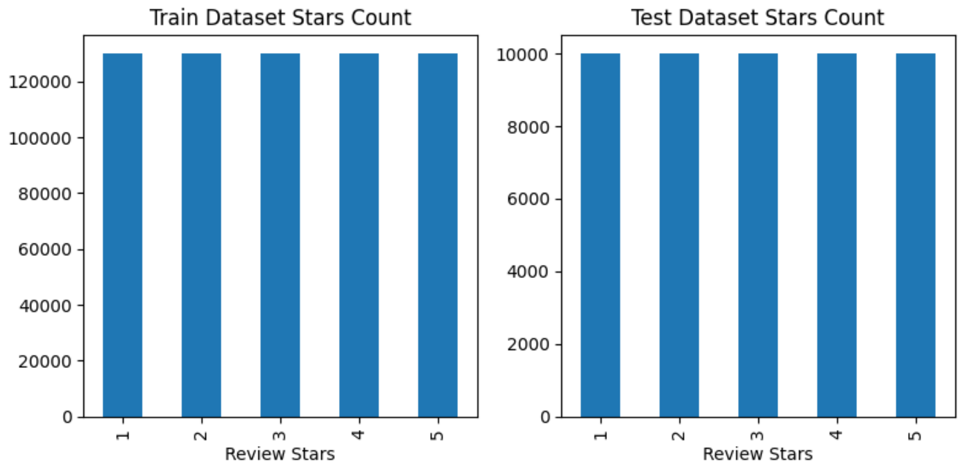
## Methodology

The dataset consists of two .csv files: training.csv with 650k records and test.csv with 50k records. A preprocessing function was used to transform the reviews lemmatized words with no stop words or special characters. Text preprocessing on both datasets takes nearly an hour. To prevent having to regenerate the dataset, the dataframe was written to a local binary file (pickled).

This project leveraged the NLTK library to process the text and transform the text review to numerical features. Classification machine learning models from the Scikit-learn library were developed by targeting the stars and the sentiment. The project utilized *Python* language and *Pandas* dataframes to interrogate the dataset. The plots are created using *Matplotlib* and *Seaborn* libraries. The machine learning models are created using the *sklearn* library. All the code is housed across *Jupyter* notebooks. The files and notebooks were compiled locally.

### Data Wrangling / Exploratory Data Analysis

The dataframe created from the train/test .csv files have two columns [class_index and review_text]. Class_index column represents the stars associated with each review. The following plot shows the distribution of stars among the dataset.



To perform sentiment classification the star rating needs to be converted to [positive, neutral, negative]. To model the dataset (1, 2) stars are categorized as "negative" / -1, while (3) stars is "neutral" / 0 and (4,5) stars are categorized as "positive"/ + 1. Converting the star rating system converted to "positive", "negative" and "neutral" leads to an imbalanced dataset.

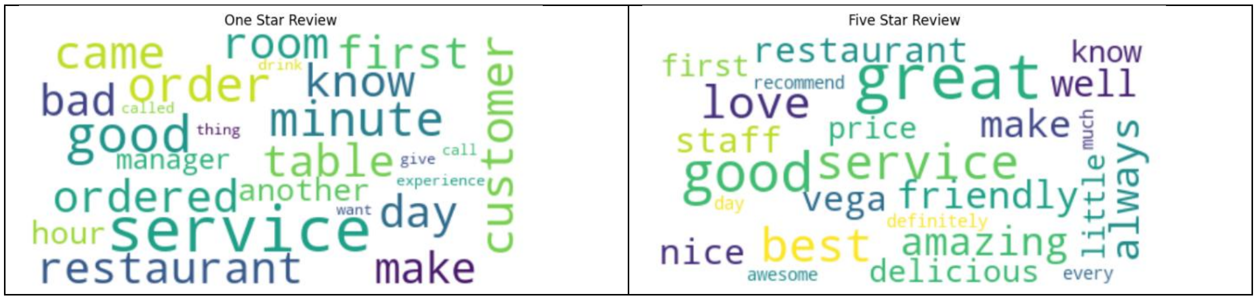
### Text Normalization

Normalization is the process that brings words into standard format. Tokenization is the process of breaking down a continuous text into smaller units called tokens. These tokens can be words but can also be larger or smaller depending on the strategy. These token words can be further standardized using a method called lemmatization i.e. reducing the word to its root base form, as found in a dictionary. For datasets with text features like this there is not a need to perform in depth data wrangling or exploratory data analysis as with traditional datasets with many numerical and categorical features.

The primary step to perform NLP requires that the original text be standardized to prepare for model creation. Standardization involves lower casing, removing stop-words, and performing lemmatization. Lemmatization involves taking a word and reducing it to its base form as found in a dictionary. Refer to ***process_text*** function in the script to see the exact steps taken in normalization. Note that the imported English stop words is a list. Additional filler words were added to remove common words like “really”, “got”, etc.

## BOW and TF-IDF

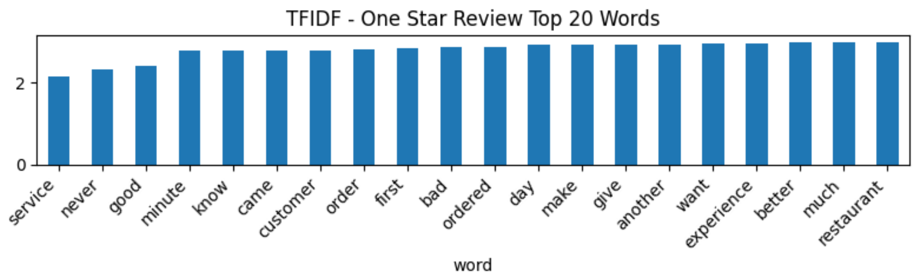
In the normalized format the reviews can be counted in terms of frequency. This method is referred to as the bag of words approach. The following figure is the frequency plot of the counted words from single-star and five-star reviews. The “Counter” function from the “collections” library was used to count the words from 1k processed reviews.  The figure shows that words like “great”, “amazing”, and “nice” are found in the five-star plot whereas “bad” shows up in the one-star plot. Note that “good” is found in both segments because the BOW approach cannot distinguish between “good” and “not” and “good” which is tokenized into two words.

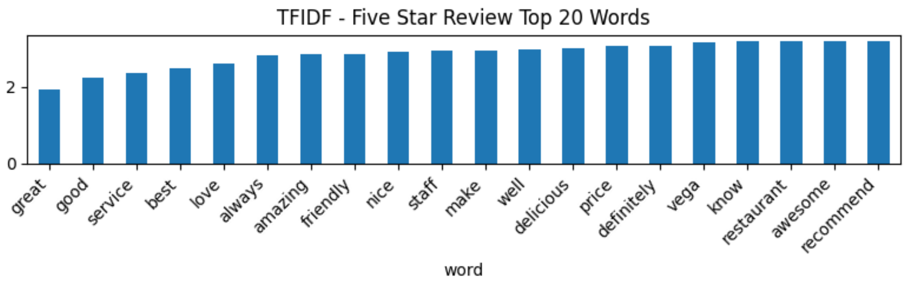


TF-IDF is a more robust method than BOW for vectorizing text as it attempts to look at how rare a word is as well as how frequent the word appears. The TF-IDF score for each word in a document is determined by multiplying its Term Frequency (TF) and its Inverse Document Frequency (IDF). The formula is:$\text{TF}-\text{IDF}(t,d) = \text{TF}(t,d) \times \text{IDF}(t)$

**Term Frequency (TF)** measures how often a word appears in a specific document while the inverse document frequency (IDF) penalizes common words. The final **TF-IDF score** is the product of these two values.

The following figure was created by vectorizing 1k reviews and plotting the IDF scores in descending order. Note that in the five-star review has “great” and “love” and “amazing” and “recommend” while the one-star review has “never” and “bad”. As in the BOW approach, some words appear in both lists as the algorithm cannot distinguish between “bad service” and “good service”.

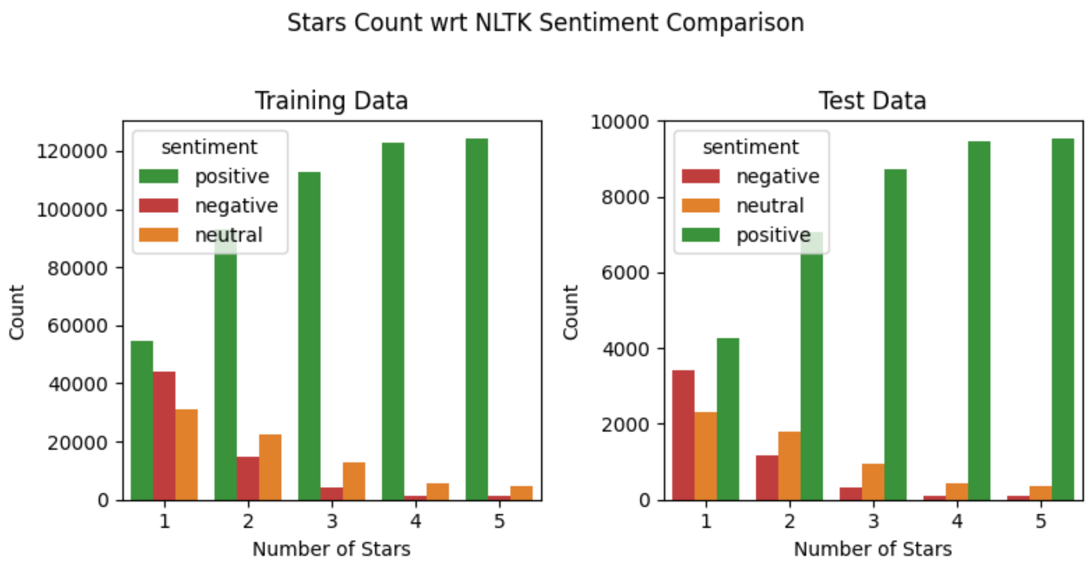




To implement TFIDF, the ***TfidfVectorizer*** function from the ***sklearn*** library was used. The function transforms the features into a sparse matrix. ***TfidfVectorizer*** has several parameters including ***max_df**, **min_df*** and ***max_features*** which affect the size of the TFIDF sparse matrix. The ***max_features*** parameter controls the number of features based on frequency in document. The ***max_df*** parameter that sets the upper bound of common words present in all documents (max_df =0.99 means to ignore words occurring in 99% of documents or higher). Similarly the ***min_df*** parameter sets the bound of a rare word present in documents to be ignored (min_df = 0.01 means ignore all words that occur in 1% or less of all documents). These three parameters affect the size of the TFIDF matrix and thereby affect model run times.

## VADER

VADER stands for **V**alence **A**ware **D**ictionary and s**E**ntiment **R**easoner. VADER is a rule-based sentiment analysis tool. The model uses pre-built lexicon of words to calculate the sentiment of a model. Using the NLTK sentiment library the ***SentimentIntensityAnalyzer*** function went through the processed reviews for the training dataset and test dataset. The following categorical plot shows the sentiments “positive”, “negative” or “neutral”. Note that the “positive” sentiment is the prevalent outcome for each star category in both datasets.



## Class Imbalance

Note from the star histogram that both the test and training datasets are balanced. To model the dataset (1, 2) stars are categorized as "negative" / -1, while (3) stars is "neutral" / 0 and (4,5) stars are categorized as "positive"/ + 1. Converting the star rating system converted to "positive", "negative" and "neutral" leads to an imbalanced dataset.

Class weights for each class are calculated in the following manner.

$$w_j = \frac{nsamples}{nclasses * nj}$$

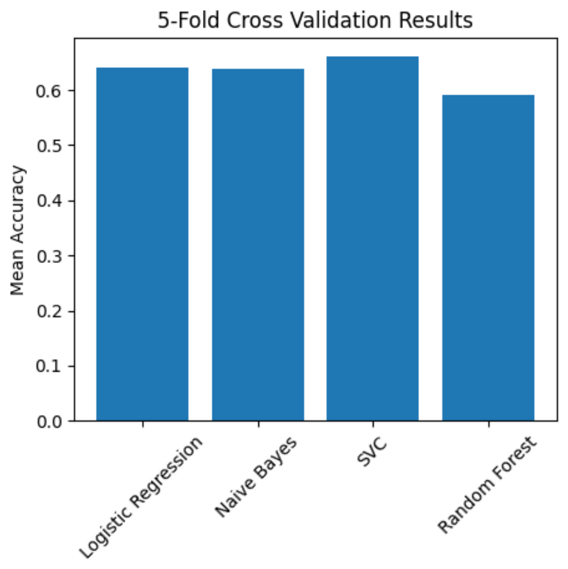
Where nsamples is the total number of samples, nclasses are the number of classes, and nj are the number of samples in the j-th class. For the training dataset: nclasses=3, nsamples=650k, npositive = 260k and nneutral = 130k. The following class weight adjustment has been supplied to each model to correct for the imbalance: class_weights = {-1:0.833, 0:1.667, 1:0.833}.

## Cross Validation

Cross validation prevents overfitting the model and ensures the model is generalizable to unseen data. A five-fold cross validation was used to evaluate all 4 models using StratifiedKFold validator. StratifiedKFold ensures class distribution preservation.

## Results

Four classification models: *Logistic Regression*, *Naïve Bayes*, *SVC*, and *Random Forest* were used to evaluate the entire training dataset. The 5-fold cross validation using StratifiedKFold and 'accuracy' as the metric yields the following plot.



The figure above shows all the models used performed relatively with the same accuracy around 67% with SVC edging out the other models and RandomForest having the worst performance. Although RandomForest yielded the worst accuracy, it also took the longest to run.

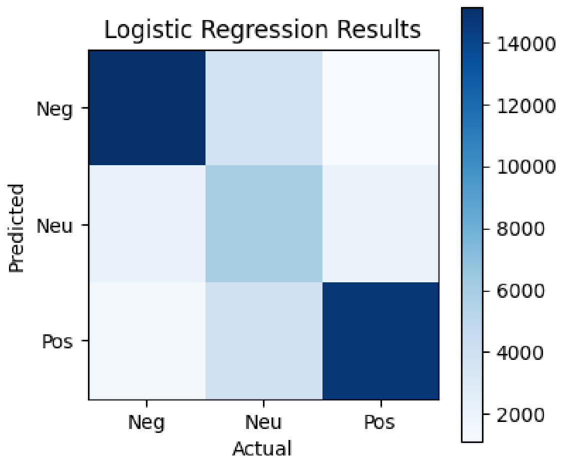
Next each model trained on the training dataset were used to make predictions on the test dataset. The following table outlines the results.

<table>
<tr><td>Logistic Regression</td><td>

```
LogisticRegression(C=0.01, class_weight={-1: 0.833, 0: 1.667,
1: 0.833})
              precision    recall  f1-score   support

          -1       0.81      0.76      0.78     20000
           0       0.44      0.59      0.50     10000
           1       0.83      0.74      0.78     20000

    accuracy                           0.72     50000
   macro avg       0.69      0.70      0.69     50000
weighted avg       0.74      0.72      0.73     50000
```



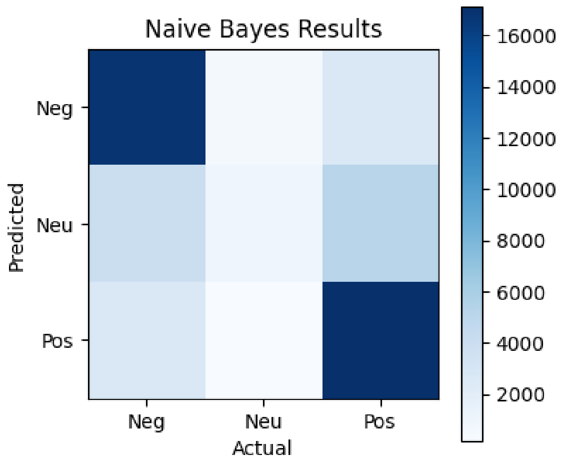
</td></tr>
<tr><td>Naïve Bayes</td><td>

```
MultinomialNB(class_prior=[0.4, 0.2, 0.4])

              precision    recall  f1-score   support

          -1       0.72      0.84      0.77     20000
           0       0.56      0.09      0.15     10000
           1       0.69      0.86      0.76     20000

    accuracy                           0.70     50000
   macro avg       0.65      0.60      0.56     50000
weighted avg       0.67      0.70      0.64     50000
```



</td></tr>
</table>

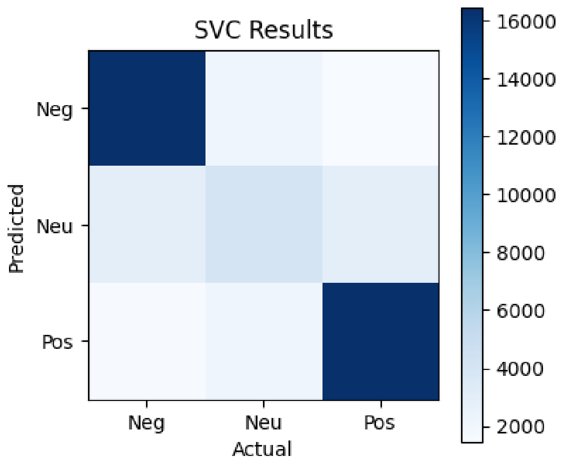
**Single Vector Classification**

```
LinearSVC(class_weight={-1: 0.833, 0: 1.667, 1: 0.833})

              precision    recall  f1-score   support

          -1       0.79      0.82      0.80     20000
           0       0.50      0.41      0.45     10000
           1       0.79      0.82      0.80     20000

    accuracy                           0.74     50000
   macro avg       0.69      0.68      0.69     50000
weighted avg       0.73      0.74      0.73     50000
```

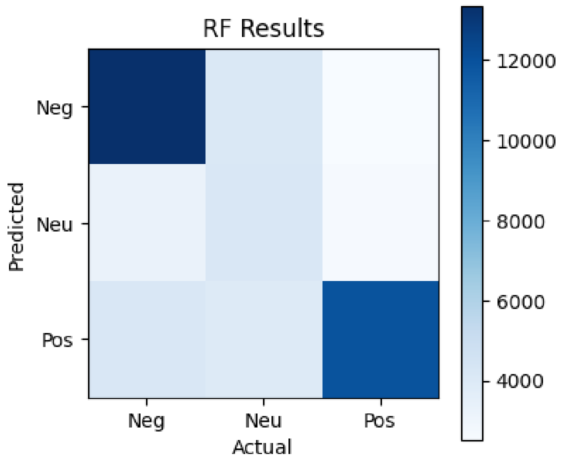


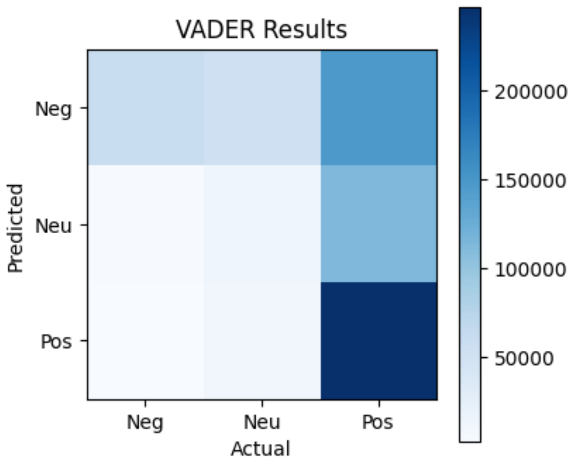
**Random Forest**

```
RandomForestClassifier(class_weight={-1: 0.833, 0: 1.667, 1:
0.833}, max_depth=3, random_state=9)
              precision    recall  f1-score   support

          -1       0.64      0.67      0.66     20000
           0       0.34      0.42      0.37     10000
           1       0.70      0.60      0.64     20000

    accuracy                           0.59     50000
   macro avg       0.56      0.56      0.56     50000
weighted avg       0.60      0.59      0.59     50000
```



| | |
|---|---|
| VADER | <pre>              precision    recall  f1-score   support<br><br>          -1       0.89      0.23      0.36    260000<br>           0       0.17      0.10      0.13    130000<br>           1       0.49      0.95      0.64    260000<br><br>    accuracy                           0.49    650000<br>   macro avg       0.52      0.43      0.38    650000<br>weighted avg       0.59      0.49      0.43    650000</pre>  |

The results show that each model predicts with similar accuracy on test data as with the 5-fold cross validation with training dataset. This proves that the models are not overfit and there is good generalization on unseen data. Even with class weighting applied, every model had difficulty predicting “neutral” class with Logistic Regression model having the highest F1 score of 0.5 on the “neutral” class.

## Recommendations

This project outlined the steps to peform sentiment analysis. The dataset was large and required significant run-times to get results. Processing of the reviews text could be improved through other NLP libraries like spaCy. The modeling accuracy of ~70% in models can be improved with a better feature extraction and also tuning the models. Performing hyper parameter optimization will allow the models to be tuned to give their best performance. Additionally, other pre-trained models including BERT and deep learning methods could be implemented on the dataset.