# Final Project: Yelp Review Sentiment Analysis

This project conducts sentiment analysis on Yelp reviews to predict whether a user review is positive or negative about a restaurant.

## Data

The data set is derived from <https://www.yelp.com/dataset> and contains the following information:

* File Name: review.json
* File Format: JSON
* Number of Records: 4.1 million
* Attributes:
  + Review ID
  + Business ID
  + Stars
  + Date
  + Text
  + Useful
  + Funny
  + Cool

## Preprocessing the Data

To determine whether a review is positive or negative sentiment, we treated reviews which are rated 3.5 or greater as positive and 2.5 or below as negative. We also used stop word removal and TFIDF in the preprocessing of data. We also add the features from a sentiment library that has around 6000 most frequent words used in reviews along with their sentiment values.

## Feature Selection

To create features, we took the text of the review. In case of VADER, statements that make up a review act as features. At the core of text classification, words act as the features.

## Validating the Results

To validate the data, we will be using 10-fold cross validation using 70% of the data set as the training set and 30% of the data set as the test set.

## Classification Models

We will be using various different classification models to determine the best model for our data. To determine the best parameters for each classifier, we will be evaluating the following:

|  |  |
| --- | --- |
| **Classification Model** | **Parameters Tested** |
| **Logistic Regression** | * maxIter = 1, 10, 50 * regParam = 0., 0.01, 0.02 * elasticNetParam = 0., 0.5, 1 |
| **Decision Tree** | * maxDepth = 5, 10, 20, 30 * minInstancesPerNode = 5, 10 * minInfoGain = 0.0, 0.05, 0.1 * impurity = gini, entropy |
| **Random Forest** | * maxDepth = 5, 10, 20, 30 * minInstancesPerNode = 5, 10 * minInfoGain = 0.0, 0.05, 0.1 * impurity = gini, entropy |
| **Gradient Boosted Tree** | * maxDepth = 5, 10, 20, 30 * minInstancesPerNode = 5, 10 * minInfoGain = 0.0, 0.05, 0.1 * maxIter = 1, 10, 50 * stepSize = 0.1, 0.5 |
| **Naive Bayes** | * smoothing = 1, 5, 10, 30, 50, 100 |

## Evaluate the Results

The following metrics were used to evaluate the results:

* Accuracy: We thought accuracy was the best parameter to evaluate our models as this is a text classification problem.

## Results

After running our models multiple times with different parameters, we found the following are the best parameters and resulted in the accuracy as mentioned below:

|  |  |  |
| --- | --- | --- |
| **Classification Model** | **Best Parameter** | **Accuracy** |
| **Logistic Regression** | * maxIter = 10 * regParam = 0.01 * elasticNetParam = 0.5 | 89.64% |
| **Decision Tree** | * maxDepth = 30 * minInstancesPerNode = 5 * minInfoGain = 0.0 * impurity = gini | 81.76% |
| **Random Forest** | * maxDepth = 30 * minInstancesPerNode = 5 * minInfoGain = 0.0 * impurity = gini, entropy | 82.52% |
| **Gradient Boosted Tree** | * maxDepth = 30 * minInstancesPerNode = 5 * minInfoGain = 0.05 * maxIter = 50 * stepSize = 0.1 | 84.38% |
| **Naive Bayes** | * smoothing = 50 | 88.23% |
| **Vader** | * no parameters | 93.62% |

## Analysis

We found that Vader, Logistic Regression and Naïve Bayes are the best models for our data and tree based models are the worst.

We trained our models on two variants of data. Initially, we trained our models to consider all the reviews with a rating of less than 3.0 to be negative and to be positive otherwise. Most of the models worked well but there we then changed our models to consider the reviews with less than 2.5 star rating to be negative and a rating of 3.5 or above to be positive and ignore the rest. Our final models use the second set of data that ignore the reviews rated between 2.5 and 3.5. This is because, we saw a significant increase in accuracy when we used the second set of data.

Also, we have to mention that Vader, LR and NB models were extremely fast in terms of execution times while the tree based models were very slow and more memory intensive with worse accuracy values. One reason we think is because, as the data grows in size, the tree structure used by the classifier grows accordingly which can be expensive in terms of memory and processing time.

Another important observation with vader is that, with the first dataset that uses all the reviews for classification, we saw the accuracy value of 85.18%. But the improvement in performance when we removed the reviews rated between 2.5 and 3.5 was huge. The accuracy improved to 93.62%. We feel the reason for this because, the reviews that are rated in the range of 3.0 comprise of largely positive sentences and the vader model is more likely to classify these reviews as positive. But in many cases, we have seen that users would have written a very good review but might have still rated the restaurant relatively low. So, eliminating the reviews in the area of overlap, improves the accuracy by a huge margin.

The advantage of using vader are in the cases where users write very long reviews. These are the reviews that have a major part of the text that is not related to the user experience as such. Vader gives the sentences that do not add context a neutral score. Thus increasing the probability of guessing the right sentiment based on the sentences in the reviews that actually add value.

## Comparing models

**Review text:**

"My goal of 2016 is to write more Yelp reviews... So here we are. My boyfriend and I just spent an hour at Target getting stuff for our new house. We were hungry and wanted to grab something quick, so we went to Steak and Shake. I would first like to note that their parking lot and sidewalks had NOT been salted... Not really the best idea considering it's been icy out for two days now. Anyways, we made it inside and it was pretty crowded. We were seated within 5 minutes of coming in. The service was OK, nothing special. I ordered the single cheeseburger and cheese fries. My boyfriend got the bacon cheeseburger and cheesy bacon fries. Our food came out pretty quickly and it was the way we ordered it (I ordered everything on my burger while my boyfriend only ordered cheese and bacon). Our food was good, I have no complaints. If you're looking for a quick and cheap bite to eat, then this is your place!"

**Vader output:**

compound: 0.3126, neg: 0.048, neu: 0.845, pos: 0.106,

**sentiment from vader** : 1 (positive review)

**sentiment from LR** : 0 (negative review)

Actual rating: 3 stars.

Vader helps in addressing the issues with users who give out lesser rating even when they write a good review. Vader performs better than LR in processing longer reviews. Some users write long reviews which have many sentences that do not add to the sentiment value. Vader assigns a neutral score to these sentences and assigns a final score based on only the sentences that have sentiment value. But LR model aggregate the sentiment values of all the words in the review regardless of their context. So, vader outperforms all other models in evaluating sentiment of longer reviews.

## Libraries

This project is implemented using Python and Spark with the following libraries:

* Pyspark Mllib
* NumPy
* Math

## Future Work

In the future, we would like to integrate other data like gender data, geographical data, , word count of reviews and how these parameters can help us in determining the overall user experience at the restaurant and hence get a better understanding of the user profile. This can be useful in building a very effective recommender system.

## Submission Structure

|  |  |  |
| --- | --- | --- |
| Purpose | File | Location |
| N/A | Report | README.pdf |
| Analyzes the data set and Predicts Sentiment | Code | Code > yelp.py  Code > vader.py |
| Explains the logic of the program | Pseudocode | Pseudocode > yelp\_pseudocode.txt |
| Results of the Best Parameters for each Model | Results | Results > results.txt |