Investigating the effects of Classification Models in Natural Language Processing

To what extent is Logistic Regression an effective classification model in accurately predicting sentiment in movie reviews when compared to Linear Regression?

Computer Science Extended Essay

Word count: 3992

1. Introduction

As language-speaking humans, determining the tone of writing is simple. We are taught to analyze the diction of the writing; we regard some words to have negative connotations, and some to have positive connotations. This is known as sentiment analysis (Raj, 2021). Nowadays, we humans are not the only ones who can discern the tone of writing -- computers can too. Sentiment analysis can be done through a procedure called Natural Language Processing (NLP) (Raj, 2021). This is a form of machine learning that has several applications, the most popular being smart assistants like Apple's Siri (Tableau).

NLP follows two main phases, data preprocessing and algorithm development (Lutkevich, 2021). Data preprocessing involves simplifying text to make it easier for the computer to analyze. Afterwards, classification models like Logistic Regression are used to classify whether the preprocessed text is positive or negative in terms of its tone. The issue with these classification models is that language is very inconsistent and it is hard to determine a consistent trend where a specific word/writing directly corresponds with a particular sentiment. Thus, this paper aims to determine if Logistic Regression, as a classification model, can be more accurate than Linear Regression in predicting the sentiment of movie reviews.

2. Background

2.1 Machine Learning

Natural language processing is a form of machine learning. It is a process where computers analyze and learn from data to make accurate predictions about a topic (Heath, 2020). The main difference between machine learning and traditional software is that machine learning models do not have code that directly explains how to classify dissimilar data. Instead, large datasets are used to train the program such that it can accurately distinguish between data (Heath, 2020). For example, NLP models can analyze the words that correlate with a review being positive and be able to classify positive reviews it has never seen before.

To reiterate, NLP involves data preprocessing, also known as feature extraction, to make it easier for a classification model like Logistic Regression or Linear Regression to find a general trend based on the frequency of keywords that contribute to the tone of writing. Different classification models, Logistic Regression and Linear Regression will be compared in this paper.

2.2 Feature Extraction Algorithms

Text feature extraction creates numerical values to represent the words used in the movie review dataset. Converting strings or words to numbers in this manner will make it easier for classification models to work with them and create a more accurate classification (Maksoud, 2019). There are many different ways of representing a word as a number, and each way can produce different results. In this experiment, two different representations were used: Bag of Words (BOW) and Term Frequency - Inverse Document Frequency (TF-IDF).

Bag of Words (BOW)

The BOW model determines how frequently a word appears in a document (Brownlee, 2017). As the name suggests, BOW stores the count of words without concern for the order in which the words come (Brownlee, 2017). This means that phrases like “not good” will not be seen as a negative evaluation of a movie review; it will instead be seen as two separate words, “not” and “good”, which may cause the program to run into accuracy problems. This is important to note because even if the classification models being tested in the investigation are flawless, the accuracy of the program will not be perfect because language is complex as it involves expressions, sarcasm, and context that cannot be interpreted through a single word.

Term Frequency - Inverse Document Frequency (TF-IDF)

TF-IDF is a more advanced feature extraction algorithm. While BOW simply counts the number of occurrences of a specific word, TF-IDF does the same but also determines how important the same word is to a document within the entire dataset. This is done by counting how many times it appears in other documents (Huilgol, 2020). Usually, when a word appears often in a review, it becomes harder to classify documents based on this common word, so TF-IDF makes this word less relevant in classifying the review (Borcan, 2020). To summarize, for each word, its TF-IDF value increases with every appearance of the word in one document; however, it decreases with every appearance in other documents (Maneja, 2021).

TF-IDF consists of two components: term frequency (*tf*) and inverse document frequency (*idf*). These two components multiply with each other to produce a meaningful TF-IDF score for any given word. High TF-IDF scores can help classification models to attribute certain words to certain reviews, allowing these models to use these features to uniquely identify other similar reviews. Term frequency determines how important a word, *t,* is to a specific document, *d*. That is, the ratio of the number of times a word occurs, to the total number of words in the same document. This is represented by the following formula:

$$tf(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}}$$

$f_{t,d}$ is the number of occurrences of a word, *t*, in review, *d*; $\sum_{t' \in d} f_{t',d}$ is the total number of words in *d*; *tf(t,d)* is the term frequency value of the word, *t*, in document, *d* (Maneja, 2021).

Inverse Document Frequency determines how common a word, *t*, is among all other documents in a dataset, *D*. It is given by the following formula:

$$idf(t, D) = log \frac{|D|}{|\{d \in D : t \in d\}|}$$

Where |*D*| is number of reviews in the dataset; |*{d ∈ D : t ∈ d}*| is the number of documents that contain the word, *t*; *idf(t, D)* is the inverse document frequency of the word, *t*, throughout the dataset (Maneja, 2021).

The ratio in the logarithm is the *reciprocal* of *(the number of documents that contain the word, t) / (total # documents)*. The reason for this is to give greater IDF scores to words that are less common among other documents (Maneja, 2021). For example, if there was a dataset of two documents, and only one of the documents contained the word “bad”, the IDF score of “bad” would be log(2/1). If this were to be compared to the word “the” which appears in both the hypothetical documents, then it will have an IDF score of log(2/2) which is less than log(2/1). Thus, the word “bad” is more meaningful than the word “the.”

Additionally, the IDF formula uses a logarithmic scale to ensure that one word does not overpower other words (Maneja, 2021). For example, if there were two terms with IDF scores of 2 and 4 respectively. They will both be considered as a meaningful feature within the dataset. However, if these IDF scores were, instead, 20 million and 40 million, only the greater score will be considered a meaningful feature as it is significantly greater than the other score (Maneja, 2021). The logarithmic scale is meant to put TF-IDF values on “equal playing fields” (Maneja, 2021).

After *tf* and *idf* are measured, they can be multiplied with each other to produce the TF-IDF score.

2.3 Classification models

Classification models aim to plot the extracted features (BOW and TF-IDF values) from the training data to a graph. For any given word, its feature value is plotted on the x-axis and the sentiment of its respective movie review is plotted on the y-axis -- if the movie review sentiment is positive, the y-value is 1 and if it is negative, the y-value is 0. Afterwards, a classification model is used to find a relationship between the features and sentiment. Based on this relationship, the model would be able to classify movie reviews that it has never seen before. The classification models for this paper are: Linear Regression which is used to plot a straight line to the data points and Logistic regression which is used to plot an “s”-shaped curve called a sigmoid function.

Linear Regression

Linear regression is a statistical algorithm and a classification model that plots a line of best fit to training data. This reveals a trend in the data which can be used to predict values of features it has never seen before. For this paper, Linear regression will be used to calculate how features such as the Bag of Words (BOW) values and Term Frequency-Inverse Document Frequency (TF-IDF) values relate to the sentiment of the text (Wolff, 2020). This will help determine the relationship between certain words in the movie reviews and its sentiment.

Linear regression follows the basic mathematical formula of a line,

$$f(x) = w_1x + w_0$$

(w_1 is the slope of the line, w_0 is the y-intercept, x is the inputted feature which will help find the sentiment of the text, f(x) (Al-Masri, 2019)).

The following graph is an example of a line fitted to feature extracted values of training data (the visual would be the same for TF-IDF values).

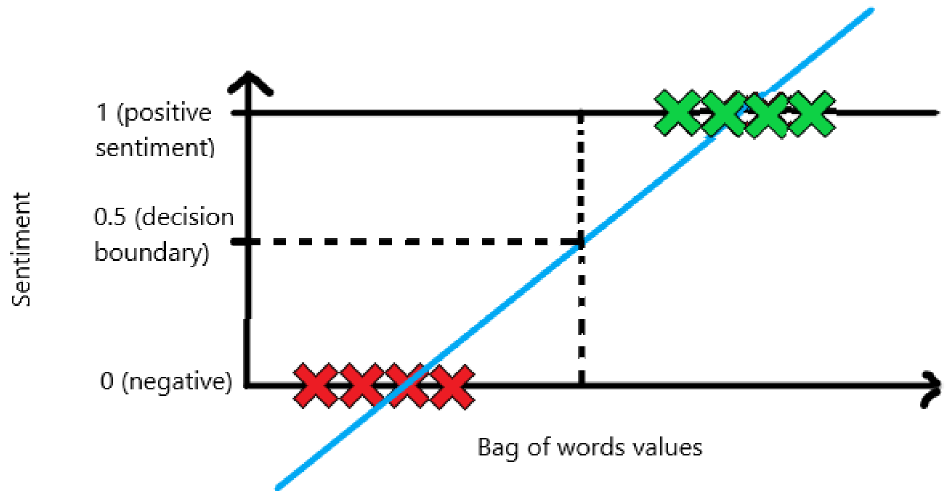


Figure 2-1: Graphical representation of Linear Regression for one word

Because there are several different words within a list of movie reviews, there will be several different linear regression lines for each word or feature. To combine all these regression lines, the following equation can be used to encompass all relevant words (add all the slope and feature values to one equation) (Great Learning Team, 2020):

$$f(x) = w_0 + w_1x_1 + w_2x_2 + \ldots + w_nx_n$$

(w_n and x_n represents the slope and feature values respectively for one word in a movie review) When classifying new movie reviews, the BOW or TF-IDF values for each word is calculated, then substituted in as one of the x values to find the overall f(x) sentiment value. An f(x) value of over 0.5, the decision boundary, indicates positive sentiment and below 0.5 indicates negative sentiment.

Finding the optimal line

While finding the slopes of the equation is irrelevant to the actual experiment, it is still important to understand how these values are found to create the most optimal linear regression model.

The problem now is that, although we have the formula to find the sentiment of a review, the values, w-values are still unknown. The classification model needs the most optimal values of these variables in order to fit an accurate line to the data. To find these optimal values, the concept of cost can be used. Cost tells us how much a line deviates from the data. For linear regression, the Squared Error Cost, $J(w_0, w_1)$, is used (Al-Masri, 2019). For any given word, for example, "good", the Squared Error Cost is calculated by finding the distance between the predicted sentiment given by this word, $f(x_i)$, and its actual sentiment given by a training data point, y_i. This distance is then squared in order to make this value positive. The same is done for every movie review containing the word "good" in the dataset, and the sum of these values is calculated. Afterwards, it is divided by the total number of movie reviews, n, to get the average cost of the word. Below, is the formula (Al-Masri, 2019):

$$J(w_0,\ w_1) = \frac{1}{2n}\sum_{i=1}^{n}(f(x_i) - y_i)^2$$

With the Squared Error Cost formula, it is necessary that the w-values that produce the smallest cost are found. To do this, gradient descent can be used. The below figure graphs w_0 against cost (A similar parabolic graph can be graphed between w_1 and cost):

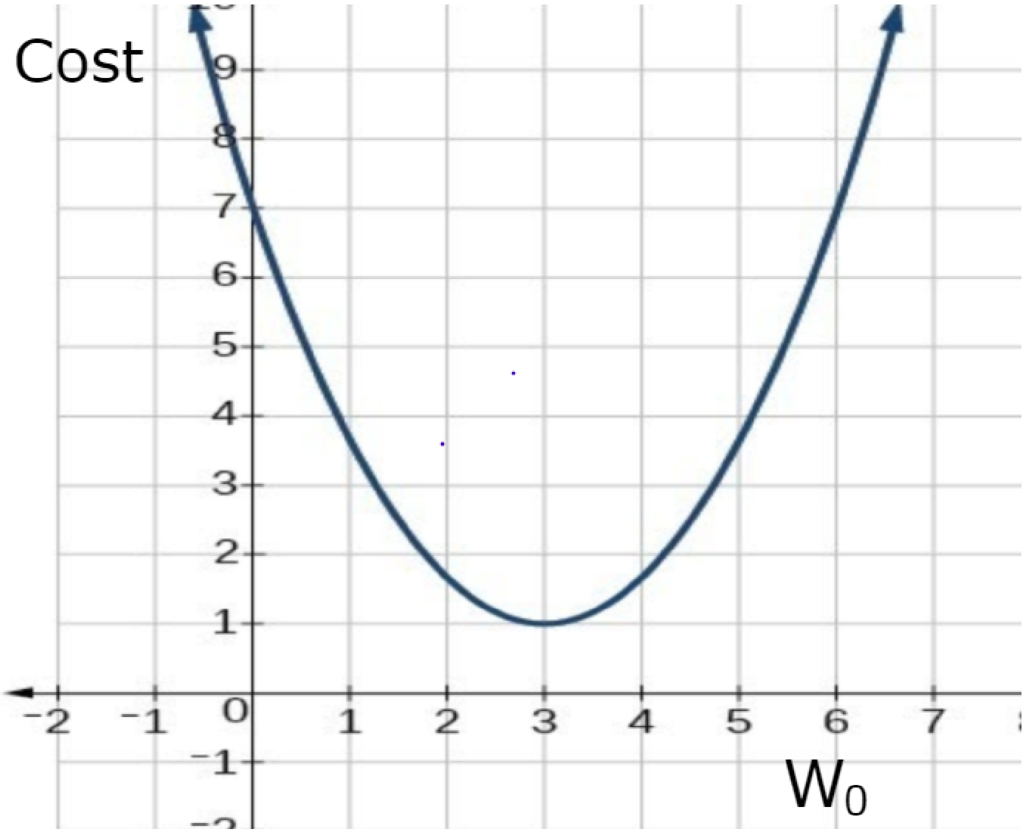


Figure 2-2: w_0 vs cost (Pai, 2021)

Initially, the values of w are set to be 0 as their optimal values are yet to be determined. An arbitrary value, L, the learning rate will be used to control how much w is changed in each iteration of the algorithm (Banerji, 2021). Usually the learning rate is a small value like 0.01 so that the w values are changed little by little each iteration, increasing accuracy (Banerji, 2021).

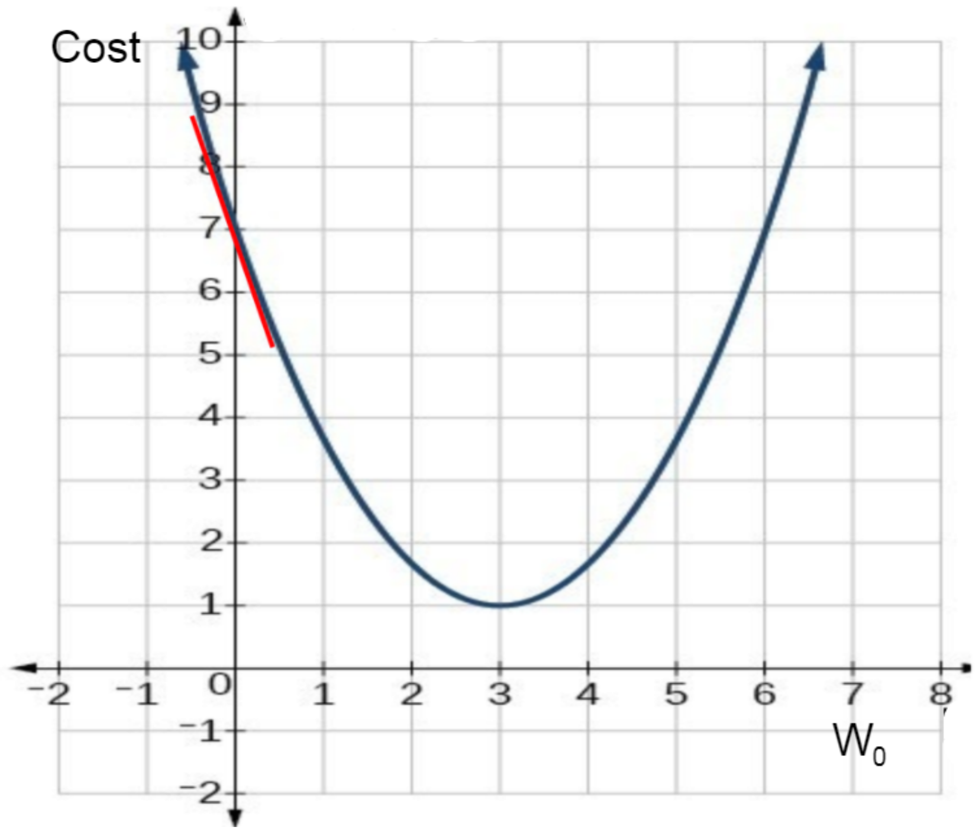


Figure 2-3: The slope of the graph at the current value of w_0 (Pai, 2021)

The algorithm starts by taking the slope, D_w, of the graph in Figure 2-2 at the current values of w. This slope is seen on Figure 2-3 as a red line. This will help figure out by how much the algorithm should change the current values of w (Banerji, 2021). The learning rate is multiplied by the slope, and this product is subtracted from the current values of w. The process is shown by the figure below:

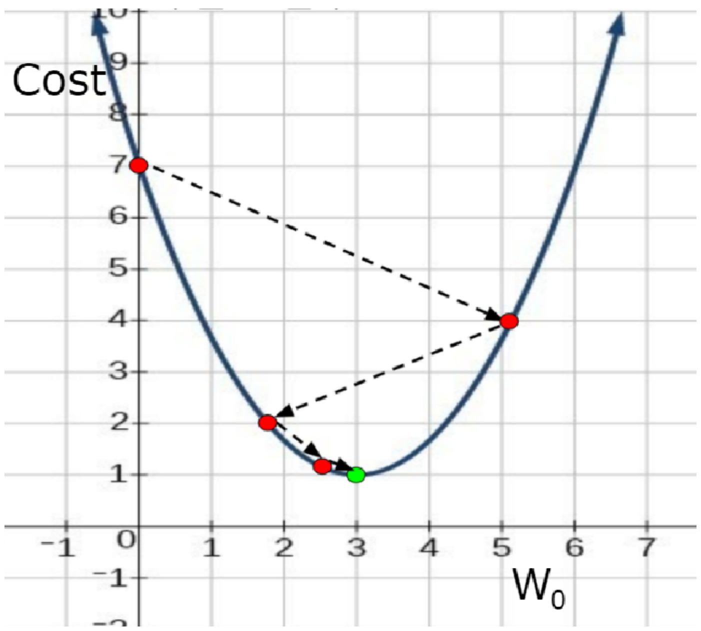


Figure 2-4: w_0 vs. cost, visualization of gradient descent (Pai, 2021)

Each arrow represents one iteration of the algorithm. As explained above, each iteration involves the following formulas (Banerji, 2021):

$$w_0 := w_0 - L \cdot D_{w_0}$$

$$w_1 := w_1 - L \cdot D_{w_1}$$

These formulas will continue to iterate until the slope reaches 0 at which point the values of w are the values that produce the minimum cost, resulting in the most optimal linear regression line (Banerji, 2021).

Logistic Regression

Logistic regression plots its training data on a graph and creates a “sigmoid” function which returns a probability value between 0 and 1 (Pant, 2019). The shape of the sigmoid function can be seen in the figure below.

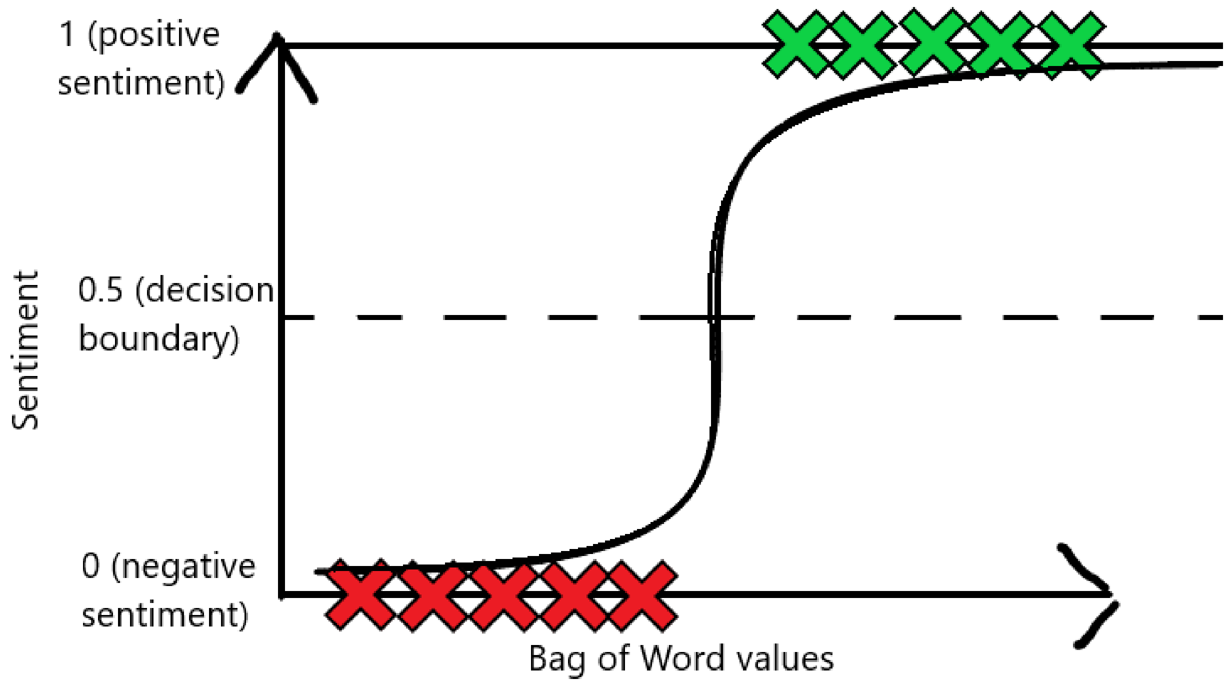


Figure 2-5: Graphical representation of Logistic Regression for one word

It then uses a decision boundary which is a value between 0 and 1, on the y-axis, that shows the model how to interpret the probability value (Pant, 2019). Usually, this value is set to 0.5 indicating that if the probability value of the text document exceeds 0.5, then the document is 1 or positive, and if it is below 0.5, then it is 0 or negative.

The function for logistic regression is the following (Pant, 2019):

$$f(x) = \frac{1}{1+e^{-(w_0+w_1x)}}$$

(f(x) is the probability that a movie review is positive)

Similarly to linear regression, if there are more than one word that must be considered in determining whether a movie review is positive or negative, the equation will combine all the slopes and variable features (Hoffman, 2019):

$$f(x) = \frac{1}{1+e^{-(w_0+w_1x_1+w_2x_2+\ldots+w_nx_n)}}$$

This will encompass all n possible words that can appear in a movie review.

Finding the optimal curve

Logistic regression similarly performs gradient descent in tandem with a cost function to find its w values. However, using the simple square error cost formula for a sigmoid function would not produce a simple parabola like in Figure 2-2 when graphing w against the cost, making it harder to find the minimum cost (Pant, 2019). Thus, the following formula, called Cross-Entropy, can be used to find cost, J(f(x), y) (Gupta, 2019):

$$J(f(x_i), y_i) = x = \begin{cases} -log(f(x_i)) & y_i = 1 \\ -log(1 - f(x_i)) & y_i = 0 \end{cases}$$

(y_i is the actual sentiment of the i'th movie review, f(x_i) is the prediction sentiment of the i'th movie review)

If y_i is 1, then it finds the log of the probability indicating that x_i relates to a positive review, according to the current function f(x_i), and vice-versa for when y_i is 0. This formula can then be combined into the following equation (Gupta, 2019):

$$J(w_0, w_1) = -\frac{1}{n}\sum_{i=1}^{n}[y_i log(f(x_i)) + (1 - y_i) log(1 - f(x_i))]$$

The average cost throughout all movie reviews in the dataset can be determined by summing the costs and dividing it by the number of movie reviews, n, as seen in the formula. The same gradient descent formulas from the linear regression section can be used to find the minimum cost of each value of w.

2.4 Comparison of the models

Linear regression's main flaw when doing binary classification such as sentiment analysis is when there is an outlier in the training data (Narkhede, 2018). The linear model overcompensates for the outlier and, in turn, fails to correctly classify some data points.

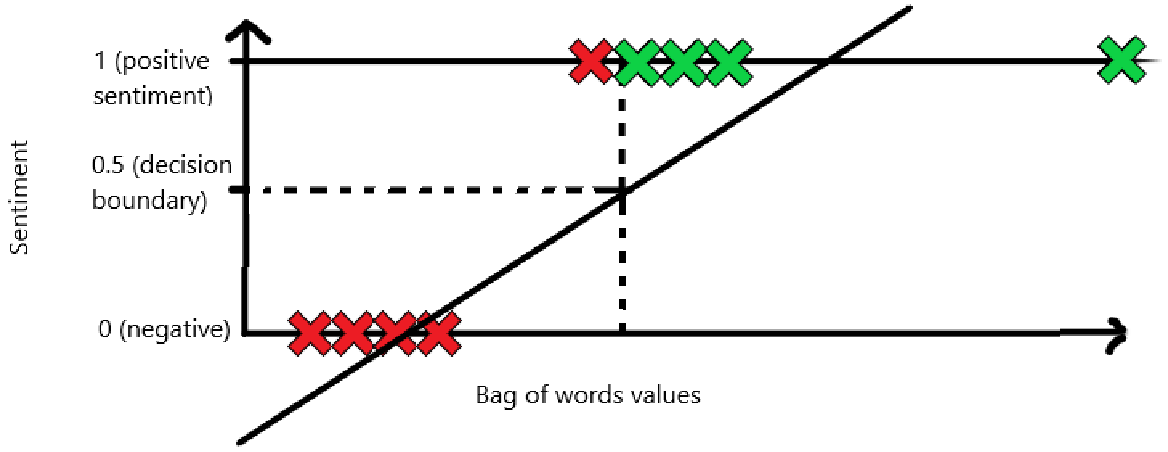


Figure 2-6: Linear regression with an outlier

As seen in Figure 2-1, a graph with a decision boundary of 0.5 and no outliers, the linear regression model fits the training data very well. It correctly identifies data points in *y*=1 to be the positive class and data points in y=0 to be the negative. However, when encountered with an outlier in Figure 2-6, the line accommodates for the outlier which causes it to incorrectly identify some y=1 data points to be negative. On the other hand, logistic regression is able to accommodate the outlier and also keep its evaluation of the normal data:

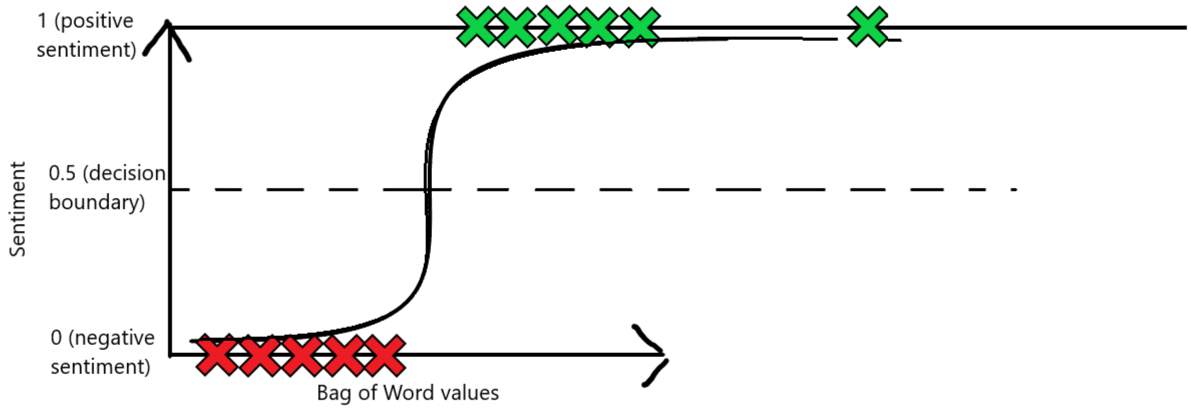


Figure 2-7: Logistic regression with an outlier

Thus, the goal is to determine how much of an impact logistic regression has on sentiment analysis on movie reviews.

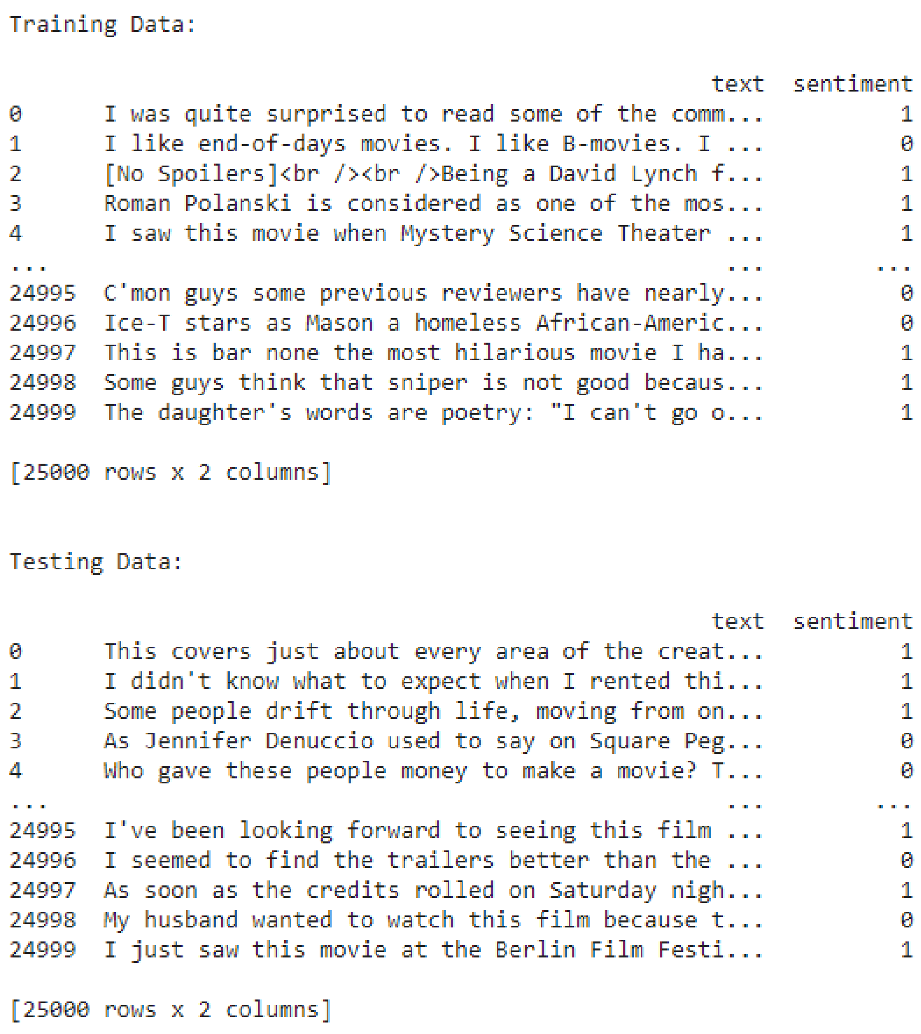
3. Experiment Methodology

This paper investigates how the accuracy of logistic regression compares to that of linear regression when classifying movie reviews. To do this, there are some required procedures and implementation.

The code for this experiment was relies heavily on (Gajavelli, 2019, see citation).

3.1 The Dataset

The data set that was used for the experiment is a list of 50,000 positive and negative movie reviews provided by (Maas, 2011, see citation). Half of the movie reviews were used for training and the other half was used to test the classification models. To organize the data, a DataFrame was created using the Pandas library. It involves putting data into a table so that the machine learning models can easily use and manipulate them. The figure below shows the DataFrame for the dataset used:

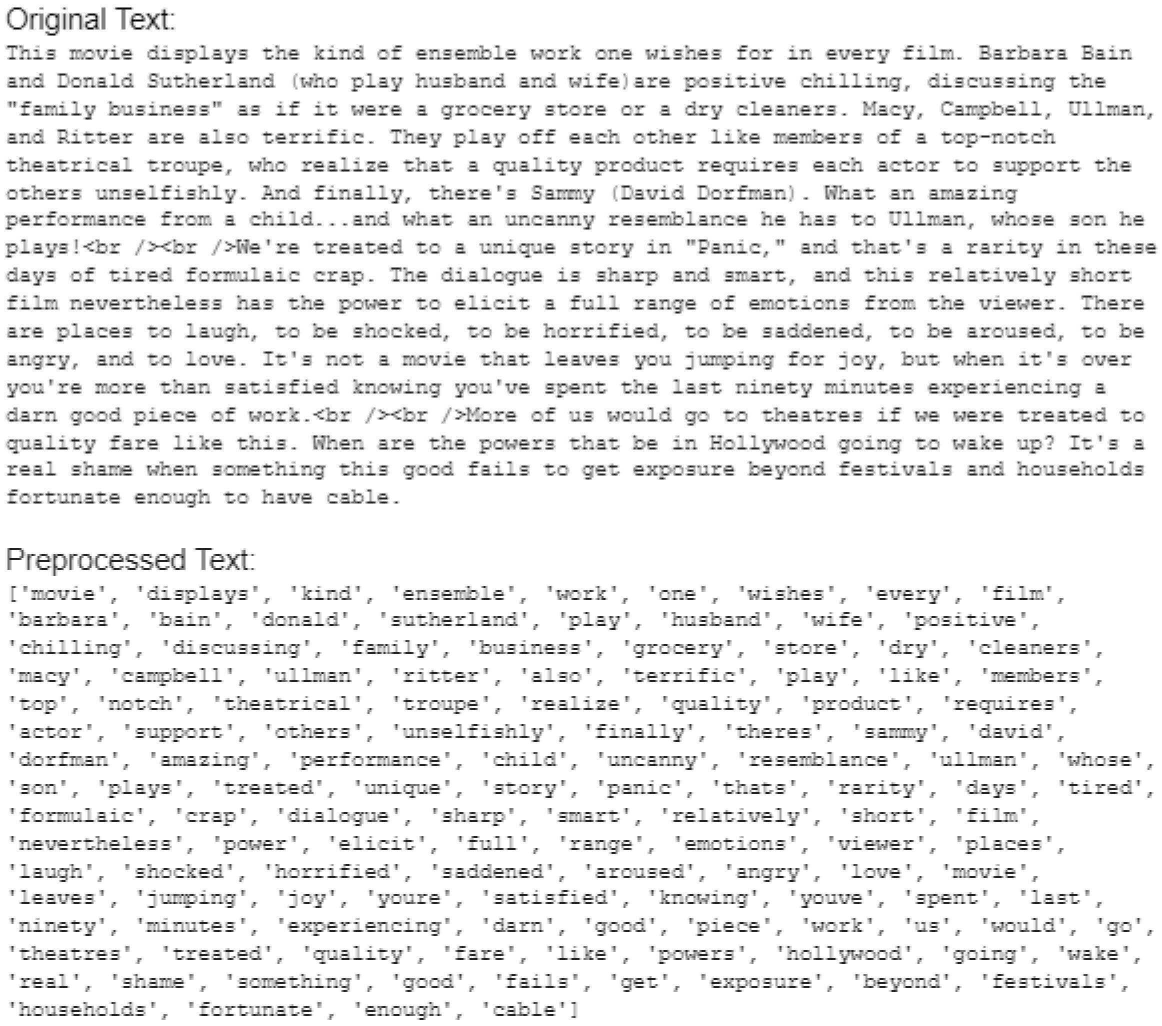


```
Training Data:

                                                    text  sentiment
0      I was quite surprised to read some of the comm...          1
1      I like end-of-days movies. I like B-movies. I ...          0
2      [No Spoilers]<br /><br />Being a David Lynch f...          1
3      Roman Polanski is considered as one of the mos...          1
4      I saw this movie when Mystery Science Theater ...          1
...                                                  ...        ...
24995  C'mon guys some previous reviewers have nearly...          0
24996  Ice-T stars as Mason a homeless African-Americ...          0
24997  This is bar none the most hilarious movie I ha...          1
24998  Some guys think that sniper is not good becaus...          1
24999  The daughter's words are poetry: "I can't go o...          1

[25000 rows x 2 columns]


Testing Data:

                                                    text  sentiment
0      This covers just about every area of the creat...          1
1      I didn't know what to expect when I rented thi...          1
2      Some people drift through life, moving from on...          1
3      As Jennifer Denuccio used to say on Square Peg...          0
4      Who gave these people money to make a movie? T...          0
...                                                  ...        ...
24995  I've been looking forward to seeing this film ...          1
24996  I seemed to find the trailers better than the ...          0
24997  As soon as the credits rolled on Saturday nigh...          1
24998  My husband wanted to watch this film because t...          0
24999  I just saw this movie at the Berlin Film Festi...          1

[25000 rows x 2 columns]
```

Figure 3-1: DataFrame for the Dataset used in the experiment

3.2 Using the Dataset

The movie reviews were put into a DataFrame where it was then preprocessed or cleaned. In the code, this was done by the “clean_text(text)” function. The function follows four main operations: converting all characters to lowercase characters, removing stopwords, removing all punctuation and unnecessary symbols, and removing HTML tags that were left in the reviews. The last two operations in the above list use Regular Expressions (RegEx). Its function, re.sub(), allows the program to replace unwanted characters, such as punctuation, with whitespace. Lastly, tokenization will be needed to make it easier to work with each individual word. This process involves turning the text into a list of words rather than a long, continuous string. With these preprocessing operations done, feature extraction and classification can be performed. Below is an example of preprocessing. Refer to Appendix A for this implementation.



Original Text:

```
This movie displays the kind of ensemble work one wishes for in every film. Barbara Bain
and Donald Sutherland (who play husband and wife)are positive chilling, discussing the
"family business" as if it were a grocery store or a dry cleaners. Macy, Campbell, Ullman,
and Ritter are also terrific. They play off each other like members of a top-notch
theatrical troupe, who realize that a quality product requires each actor to support the
others unselfishly. And finally, there's Sammy (David Dorfman). What an amazing
performance from a child...and what an uncanny resemblance he has to Ullman, whose son he
plays!<br /><br />We're treated to a unique story in "Panic," and that's a rarity in these
days of tired formulaic crap. The dialogue is sharp and smart, and this relatively short
film nevertheless has the power to elicit a full range of emotions from the viewer. There
are places to laugh, to be shocked, to be horrified, to be saddened, to be aroused, to be
angry, and to love. It's not a movie that leaves you jumping for joy, but when it's over
you're more than satisfied knowing you've spent the last ninety minutes experiencing a
darn good piece of work.<br /><br />More of us would go to theatres if we were treated to
quality fare like this. When are the powers that be in Hollywood going to wake up? It's a
real shame when something this good fails to get exposure beyond festivals and households
fortunate enough to have cable.
```

Preprocessed Text:

```
['movie', 'displays', 'kind', 'ensemble', 'work', 'one', 'wishes', 'every', 'film',
'barbara', 'bain', 'donald', 'sutherland', 'play', 'husband', 'wife', 'positive',
'chilling', 'discussing', 'family', 'business', 'grocery', 'store', 'dry', 'cleaners',
'macy', 'campbell', 'ullman', 'ritter', 'also', 'terrific', 'play', 'like', 'members',
'top', 'notch', 'theatrical', 'troupe', 'realize', 'quality', 'product', 'requires',
'actor', 'support', 'others', 'unselfishly', 'finally', 'theres', 'sammy', 'david',
'dorfman', 'amazing', 'performance', 'child', 'uncanny', 'resemblance', 'ullman', 'whose',
'son', 'plays', 'treated', 'unique', 'story', 'panic', 'thats', 'rarity', 'days', 'tired',
'formulaic', 'crap', 'dialogue', 'sharp', 'smart', 'relatively', 'short', 'film',
'nevertheless', 'power', 'elicit', 'full', 'range', 'emotions', 'viewer', 'places',
'laugh', 'shocked', 'horrified', 'saddened', 'aroused', 'angry', 'love', 'movie',
'leaves', 'jumping', 'joy', 'youre', 'satisfied', 'knowing', 'youve', 'spent', 'last',
'ninety', 'minutes', 'experiencing', 'darn', 'good', 'piece', 'work', 'us', 'would', 'go',
'theatres', 'treated', 'quality', 'fare', 'like', 'powers', 'hollywood', 'going', 'wake',
'real', 'shame', 'something', 'good', 'fails', 'get', 'exposure', 'beyond', 'festivals',
'households', 'fortunate', 'enough', 'cable']
```

Figure 3-2: The preprocessing of a movie review

3.3 The Experiment

Independent Variable

The independent variable of this experiment is the classification models (linear regression and logistic regression). Both of these models were implemented twice with either BOW or TF-IDF. This means that there were four different trials.

To implement the classification models and feature extraction algorithms, the SciKit-Learn library was used. Firstly, the program uses the SciKit-Learn library's TfidfVectorizer to calculate the TF-IDF values. These TF-IDF values were used in LogisticRegression() and LinearRegression(), from the SciKit-Learn Library, in two separate trials. Moreover, the CountVectorizer was used to produce the BOW values of the dataset and the classification models were also used with these values. Refer to Appendix B for this implementation.

Dependent Variable

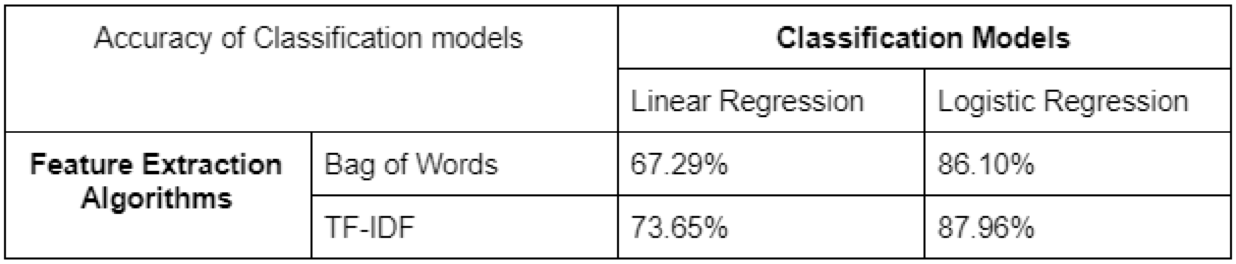
The accuracy or ability to classify texts was recorded in each of the four trials. This was done after training the classification models and was calculated by taking the number of correct classifications and dividing it by the total number of predictions.

Procedure

1. The movie review dataset was loaded and put into a DataFrame for organization
2. The reviews were cleaned and HTML tags, punctuation, and stop words were removed.
3. The words were converted into TF-IDF values using feature extraction
4. TF-IDF values for 25,000 movie reviews (training data) were plotted on a graph and linear regression was performed, graphing the relationship between the features and sentiment
5. The TF-IDF values of the remaining 25,000 movie review (testing data) was found
6. The testing data's features were plotted according to the linear regression model
7. The movie reviews that were classified with values over 0.5 were classified as positive, and those that were under 0.5 were classified as negative
8. The predicted sentiments in step 7 was compared to the reviews' actual sentiments and and accuracy percentage was calculated by dividing the number of correct predictions by the total number of testing movie reviews (25,000)
9. Step 3-8 were repeated for BOW and Logistic regression for 4 total trials

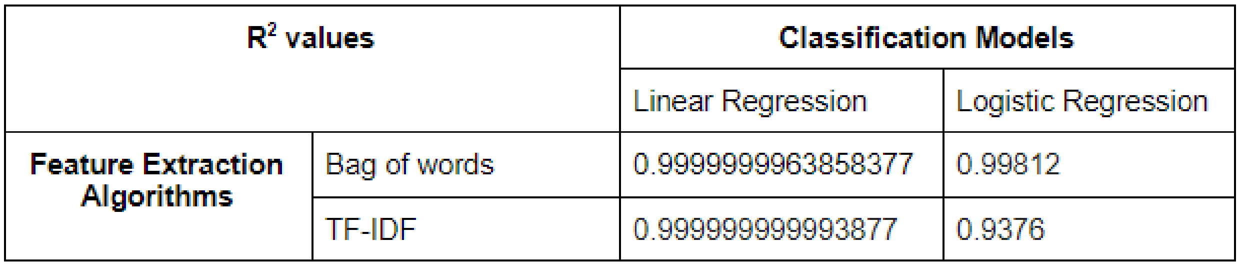
4. Experimental Results

Tables



Accuracy of Classification models		**Classification Models**	
		Linear Regression	Logistic Regression
Feature Extraction Algorithms	Bag of Words	67.29%	86.10%
	TF-IDF	73.65%	87.96%

Figure 4-1: Accuracy of classification models in each of the experiments



R^2 values		**Classification Models**	
		Linear Regression	Logistic Regression
Feature Extraction Algorithms	Bag of words	0.9999999963858377	0.99812
	TF-IDF	0.999999999993877	0.9376

Figure 4-2: R^2 values in each experiment

Confusion Matrices

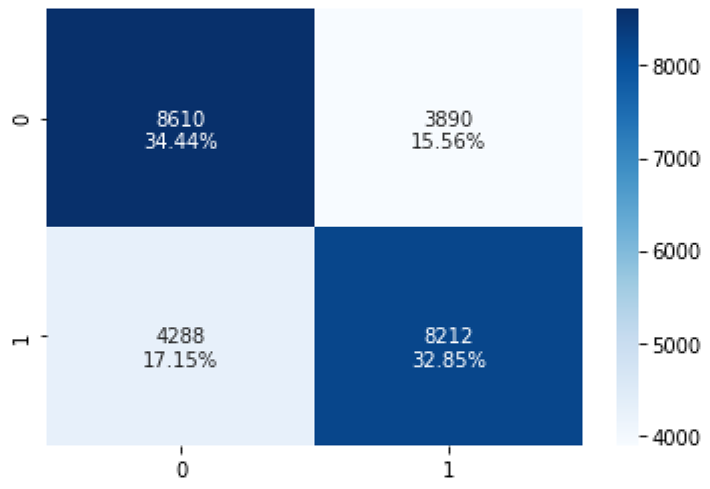


Figure 4-3: Confusion matrix of Linear regression with bag of words

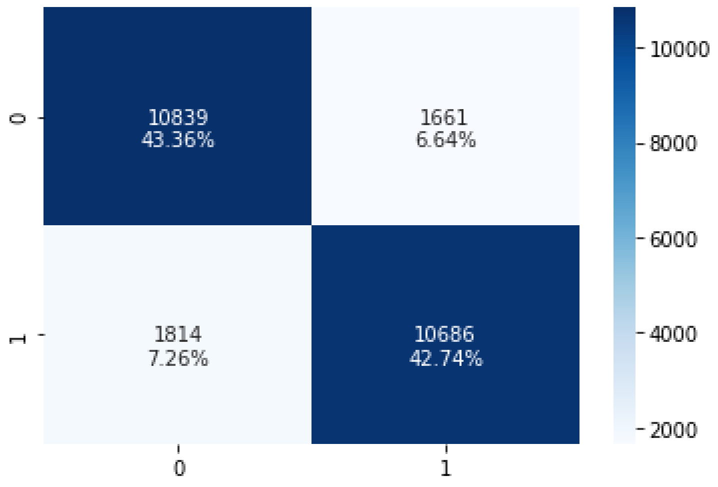


Figure 4-4: Confusion matrix of logistic regression with bag of words

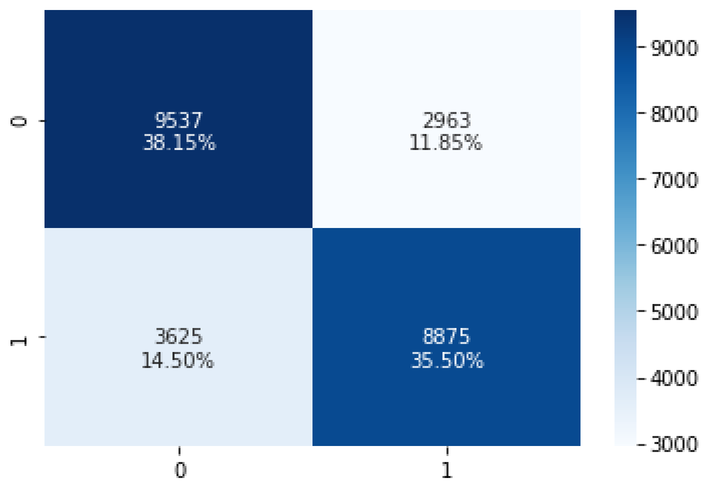


Figure 4-5: Confusion matrix of linear regression with TF-IDF

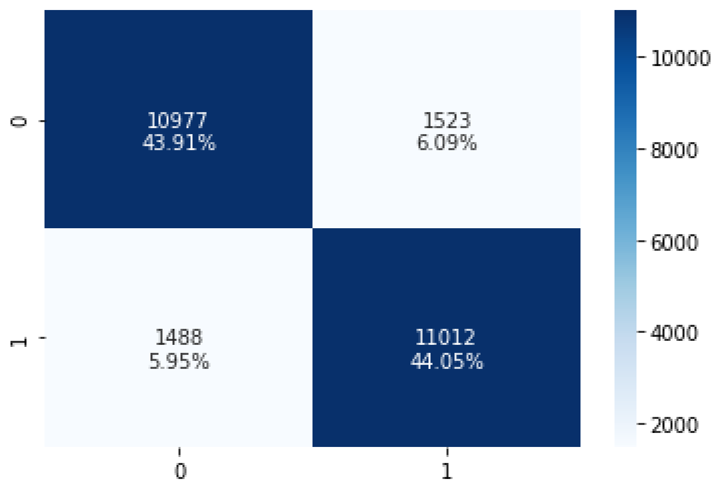


Figure 4-6: Confusion matrix of logistic regression with TF-IDF

5. Data Analysis

5.1 The results

Overall, logistic regression yields significantly higher accuracies than linear regression. As seen in Figure 4-1, the accuracy of logistic regression, when using either the Bag of Words (BOW) or Term Frequency-Inverse Document Frequency (TF-IDF) models, exceeds 85%. Contrastingly, the accuracy of linear regression is significantly worse, with its lowest accuracy at 67.29%.

The confusion matrices indicate the number of correct and incorrect predictions the models make. The vertical axis indicates the actual sentiment while the horizontal axis indicates the predicted sentiment. This means that if a classification model is accurate, there will be a blue diagonal line from the top-left corner to the bottom-right corner (Narkhede, 2018). Shown by Figures 4-3 and 4-5, the linear regression confusion matrices do not have a clean diagonal line while 4-4 and 4-6 do, suggesting that logistic regression is more accurate.

5.2 Analysis

Logistic regression gives higher accuracies than Linear regression. This is primarily due to the sigmoidal shape given by logistic regression. Recall from *Figure 2-6*, a linear regression model would fail to accommodate an outlier in the data. Unlike linear regression, seen in *Figure 2-7*, the sigmoidal shape of the logistic regression model maintains its classification of the normal features.

Additionally, it can be seen that there is a significant increase in accuracy when the Linear regression model uses TF-IDF values instead of BOW values. This can be attributed to the fact that TF-IDF values are calculated on a logarithmic scale. Logarithmic functions increase by a very small amount and thus, the TF-IDF values will be relatively close to each other on the x-axis. As a result, using TF-IDF values reduces the chances of an outlier. On the other hand, there is an increase in the accuracy of Logistic regression when it uses TF-IDF instead of BOW; however, this increase is very small because outliers are already properly considered in logistic regression.

Usually, having a high R^2 value reveals that a model fits a relationship very well. However, in this case, seen in *Figure 4-2,* the logistic regression models have lower values than the linear regression models -- which is counter-intuitive. Linear regression's high R^2 value is indicative of "overfitting" where the linear regression model fits the training data too well. This causes the model to represent the random errors or noise within the training data rather than the actual relationship between the features and sentiment (Frost, 2017). "Random noise" are things like extreme outliers. Essentially, this makes it difficult to predict features it has never seen. This issue is most common when the model is complex such as in this case where there are many features (Frost, 2017). Additionally, the sigmoid function's shape is specialized for finding the probability between two different classes whereas linear regression predicts continuous values which go beyond the values between 0 and 1 (Jing, 2020), making it less optimized for binary classification.

6. Limitations

6.1 The dataset

The issue with the dataset used in the experiment was that its training data only had highly polar examples of movie reviews. This meant that the experiment did not use reviews that are somewhat positive/negative and thus, the investigation does not actually represent a real world situation.

6.2 Preprocessing

Preprocessing could have improved as there are some aspects that may have affected the accuracy. An issue that could have arisen within the classification process is the fact that the words were not put into context, they were simply evaluated individually. This can be avoided by using n-grams which not only take words individually, but it also takes groups of consecutive words (Ganesan, 2020). This can take in phrases like “not good” and avoid wrongly classifying negative reviews containing the word “good” as positive.

Also, lemmatization could have been used. This reduces words to its root form. For example, words like “playing” and “played” are reduced to “play” (Jabeen, 2018). This will group similar words together to produce a more accurate and consistent feature.

7. Further Research Opportunities

7.1 Investigating different dataset sizes

Seeing that the large amount of training data could have caused overfitting, it would be interesting to test different dataset sizes to either avoid overfitting or making it more severe. This will show how linear and logistic regression fare against different amounts of data.

7.2 Exploring more classification models

This paper only compared logistic regression to linear regression. However, it would be very insightful to compare logistic regression to other modern classification models such as Naives Bayes which has its own advantages such as being able to get accurate results even with a small amount of training data (MonkeyLearn).

8. Conclusion

This paper explored the accuracies of linear regression and logistic regression when paired with different feature extraction algorithms, including TF-IDF and BOW. Evidently, logistic regression is more accurate in classifying sentiment than linear regression.

As a result of higher R^2 values, linear regression is subjected to overfitting. Linear regression also suffers when feature values come from a large range of values such as when it is paired with a BOW where features are not “put on the same playing field” as the other features.

Logistic regression yields much higher accuracy than linear regression because the shape of the sigmoid function allows for a better representation of two separate classes, positive and negative. It can account for outliers and large ranges of values without sacrificing the accuracy of the other features. In this specific case, its R^2 values are generally smaller than it is for linear regression but these R^2 values are still relatively high. This indicates that it avoids overfitting but stays very consistent to the relationship between the features and sentiment.

To answer the research question, “To what extent is Logistic Regression an effective classification model in accurately predicting sentiment in movie reviews when compared to Linear Regression?” logistic regression is an extremely accurate classification model as it reaches significantly higher accuracies than linear regression.

Works Cited

Al-Masri, A. (2019, March 18). *How does linear regression actually work?* Medium. Retrieved July 16, 2021, from https://towardsdatascience.com/how-does-linear-regression-actually-work-3297021970dd

Banerji, A. (2021, April 9). *Gradient descent in linear regression: Introduction to gradient descent*. Analytics Vidhya. Retrieved July 17, 2021, from http://www.analyticsvidhya.com/blog/2021/04/gradient-descent-in-linear-regression/

Borcan, M. (2020, June 8). *TF-IDF explained and python sklearn implementation*. Medium. Retrieved July 5, 2021, from https://towardsdatascience.com/tf-idf-explained-and-python-sklearn-implementation-b020c5e83275

Brownlee, J. (2017, October 9). *A gentle introduction to the bag-of-words model*. Machine Learning Mastery. Retrieved July 5, 2021, from https://machinelearningmastery.com/gentle-introduction-bag-words-model/

Frost, J. (2017). *Five reasons why your R-squared can be too high*. Statistics By Jim. Retrieved August 20, 2021, from https://statisticsbyjim.com/regression/r-squared-too-high/

Gajavelli, S. K. (2019, May 8). *Sentiment-analysis-of-IMDB-movie-reviews/untitled.ipynb at master · shivakrishna2497/sentiment-analysis-of-IMDB-movie-reviews*. GitHub. Retrieved July 23, 2021, from https://github.com/shivakrishna2497/Sentiment-Analysis-of-IMDB-Movie-Reviews/blob/master/Untitled.ipynb

Ganesan, K. (2020). *What are N-grams?* Kavita Ganesan, PhD. Retrieved August 10, 2021, from https://kavita-ganesan.com/what-are-n-grams/

Great Learning Team. (2020, August 17). *Introduction to multivariate regression analysis*. GreatLearning Blog: Free Resources what Matters to shape your Career! Retrieved November 29, 2021, from http://www.mygreatlearning.com/blog/introduction-to-multivariate-regression/

Gupta, K., Fortuner, B., Vadhadiya, P., & Genter, J. (2019, December 15). *ML-glossary/logistic_regression.RST at master · bfortuner/ML-Glossary*. GitHub. Retrieved July 22, 2021, from https://github.com/bfortuner/ml-glossary/blob/master/docs/logistic_regression.rst

Heath, N. (2020, December 16). *What is machine learning? everything you need to know*. ZDNet. Retrieved July 5, 2021, from http://www.zdnet.com/article/what-is-machine-learning-everything-you-need-to-know/

Hoffman, J. (2019). *Logistic Regression Analysis*. Logistic Regression Analysis - an overview | ScienceDirect Topics. Retrieved November 15, 2021, from http://www.sciencedirect.com/topics/medicine-and-dentistry/logistic-regression-analysis

Huilgol, P. (2020, February 28). *Bow model and TF-IDF for creating feature from text*. Analytics Vidhya. Retrieved July 17, 2021, from http://www.analyticsvidhya.com/blog/2020/02/quick-introduction-bag-of-words-bow-tf-idf/

Jabeen, H. (2018, October 23). *Stemming and lemmatization in python*. DataCamp Community. Retrieved August 25, 2021, from http://www.datacamp.com/community/tutorials/stemming-lemmatization-python

Jing, H. (2020, June 22). *Why linear regression is not suitable for classification*. Khok Hong Jing (Jingles). Retrieved August 25, 2021, from https://jinglescode.github.io/2019/05/07/why-linear-regression-is-not-suitable-for-classification/

Lutkevich, B., & Burns, E. (2021, March 2). *What is natural language processing? an introduction to NLP*. SearchEnterpriseAI. Retrieved July 5, 2021, from https://www.techtarget.com/searchenterpriseai/definition/natural-language-processing-NLP

Maas, A. (2011). *Large Movie Review Dataset*. Sentiment Analysis. Retrieved July 22, 2021, from http://ai.stanford.edu/~amaas/data/sentiment/

Maksoud, E. (2019). *Feature extraction*. Feature Extraction - an overview | ScienceDirect Topics. Retrieved July 17, 2021, from https://www.sciencedirect.com/topics/engineering/feature-extraction

Maneja, D. (2021, February 17). *How TF-IDF Works*. Medium. Retrieved July 17, 2021, from https://towardsdatascience.com/how-tf-idf-works-3dbf35e568f0

MonkeyLearn. (n.d.). *Guide to text classification with machine learning & NLP*. MonkeyLearn. Retrieved July 5, 2021, from https://monkeylearn.com/text-classification/

Narkhede, S. (2018, May 17). *Understanding logistic regression*. Medium. Retrieved July 17, 2021, from https://towardsdatascience.com/understanding-logistic-regression-9b02c2aec102

Narkhede, S. (2018, May 9). *Understanding confusion matrix*. Medium. Retrieved August 15, 2021, from https://towardsdatascience.com/understanding-confusion-matrix-a9ad42dcfd62

Pai, A. (2020, October 27). *Mean squared error cost function*. Machine Learning Works. Retrieved July 17, 2021, from http://www.machinelearningworks.com/tutorials/mean-squared-error-cost-function

Pai, A. (2021, April 17). *Gradient descent*. Machine Learning Works. Retrieved July 17, 2021, from http://www.machinelearningworks.com/tutorials/gradient-descent

Pant, A. (2019, January 22). *Introduction to logistic regression*. Medium. Retrieved July 17, 2021, from https://towardsdatascience.com/introduction-to-logistic-regression-66248243c148

Raj, N. (2021, June 15). *Sentiment analysis: Sentiment analysis in Natural Language Processing*. Analytics Vidhya. Retrieved July 5, 2021, from https://www.analyticsvidhya.com/blog/2021/06/nlp-sentiment-analysis/

Tableau. (n.d.). *8 Natural Language Processing (NLP) examples*. Tableau. Retrieved July 5, 2021, from http://www.tableau.com/learn/articles/natural-language-processing-examples

Wolff, R. (2020, April 20). *Sentiment Analysis & Machine Learning*. MonkeyLearn Blog. Retrieved July 6, 2021, from https://monkeylearn.com/blog/sentiment-analysis-machine-learning/

Appendix

Appendix A

The following are the libraries required to perform various functions such as creating tf-idf and BOW values and also using logistic and linear regression on them

```
import os
import re
import numpy as np
import pandas as pd
from sklearn.feature_extraction.text import CountVectorizer
from sklearn.metrics import accuracy_score
from sklearn.feature_extraction.text import TfidfVectorizer
import seaborn as sns
from sklearn.metrics import confusion_matrix
from sklearn.linear_model import LogisticRegression
from sklearn.linear_model import LinearRegression
```

The following function loads in the dataset such that the program can use it. The goal is to take in the dataset and return the training and testing data in a DataFrame. This process is supported by the Pandas library.

```
def load_train_test_imdb_data(data_dir):

 data = {}
 for dataset in ["train", "test"]:
     data[dataset] = []
     for sentiment in ["neg", "pos"]:
         score = 1 if sentiment == "pos" else 0

         path = os.path.join(data_dir, dataset, sentiment)
         file_names = os.listdir(path)
         for f_name in file_names:
             with open(os.path.join(path, f_name), "r") as f:
                 review = f.read()
                 data[dataset].append([review, score])
 data["train"] = pd.DataFrame(data["train"], columns=['text', 'sentiment'])
 print("Training Data: \n")
 print(data["train"])
 print("\n")
 data["test"] = pd.DataFrame(data["test"], columns=['text', 'sentiment'])
 print("Testing Data: \n")
 print(data["test"])
 return data["train"], data["test"]

train_data, test_data = load_train_test_imdb_data(data_dir="aclImdb/")
```

This is the function that cleans the text.
The following operations are done on it:

- Remove HTML tags
- Remove punctuation
- Putting all characters to its lowercase form
- Removing words that are irrelevant to the sentiment to the text, also called stopwords

```
def clean_text(text):
 #Stopwords
 stopwords =
['i','me','my','myself','we','our','ours','ourselves','you',"you're","you've","
you'll","you'd",'your','yours','yourself','yourselves','he','him','his','himsel
f','she',"she's",'her','hers','herself','it',"it's",'its','itself','they','them
','their','theirs','themselves','what','which','who','whom','this','that',"that
'll",'these','those','am','is','are','was','were','be','been','being','have','h
as','had','having','do','does','did','doing','a','an','the','and','but','if','o
r','because','as','until','while','of','at','by','for','with','about','against'
,'between','into','through','during','before','after','above','below','to','fro
m','up','down','in','out','on','off','over','under','again','further','then','o
nce','here','there','when','where','why','how','all','any','both','each','few',
'more','most','other','some','such','no','nor','not','only','own','same','so','
than','too','very','s','t','can','will','just','don',"don't",'should',"should'v
e",'now','d','ll','m','o','re','ve','y','ain','aren',"aren't",'couldn',"couldn'
t",'didn',"didn't",'doesn',"doesn't",'hadn',"hadn't",'hasn',"hasn't",'haven',"h
aven't",'isn',"isn't",'ma','mightn',"mightn't",'mustn',"mustn't",'needn',"needn
't",'shan',"shan't",'shouldn',"shouldn't",'wasn',"wasn't",'weren',"weren't",'wo
n',"won't",'wouldn',"wouldn't"]

 # remove HTML tags
 text = re.sub(r'<.*?>', '', text)
  # remove the characters [\], ['] and ["]
 text = re.sub(r"\\", "", text)
 text = re.sub(r"\'", "", text)
 text = re.sub(r"\"", "", text)
  # convert text to lowercase
 text = text.strip().lower()
  # replace punctuation characters with spaces
 filters='!"\'#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n'
 translate_dict = dict((c, " ") for c in filters)
 translate_map = str.maketrans(translate_dict)
 text = text.translate(translate_map)

 #removing stopwords
 words = text.split()
 result = ""
 for word in words:
   if word not in stopwords:
     result = result+" "+word

 text = result
 return text
```

Appendix B

Linear Regression on TF-IDF

```
# This code will convert the text from the movie reviews to TF-IDF values
vectorizer = TfidfVectorizer(stop_words="english",preprocessor=clean_text,)

#this uses the vectorizer to produce the training and testing tf-idf features
for the words
```

```
training_features = vectorizer.fit_transform(train_data["text"])
test_features = vectorizer.transform(test_data["text"])

# Training the linear regression model
model = LinearRegression()

#this plots the training features onto a plane and then plots a linear
regression line according to it
model.fit(training_features, train_data["sentiment"])
y_pred = model.predict(test_features)

# decision boundary. Predictions greater than 0.5 are positive and predictions
less than 0.5 are negative
for i in range(len(y_pred)):
 if y_pred[i]<0.5:
   y_pred[i] = 0
 else:
   y_pred[i] = 1



# Evaluation
#finding the accuracy of the linear regression model on tf-idf values by
comparing the predicted sentiments to the actual sentiments
acc = accuracy_score(test_data["sentiment"], y_pred)
print(acc*100)

# confusion matrix creation
matrix = confusion_matrix(test_data["sentiment"], y_pred)

#the following functions will help format the confusion matrices to make them
more visually appealing
group_names = ['','','','']
group_counts = ["{0:0.0f}".format(value) for value in
               matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                    matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
         zip(group_names,group_counts,group_percentages)]
```

```
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
```

Logistic Regression on TF-IDF

```
# Training the logistic regression model
model = LogisticRegression()
model.fit(training_features, train_data["sentiment"])

y_pred = model.predict(test_features)

# Evaluation

#here we continue to use the tf-idf values we found in the previous trial
acc = accuracy_score(test_data["sentiment"], y_pred)
print(acc*100)
#confusion matrix creation
matrix = confusion_matrix(test_data["sentiment"], y_pred)

group_names = ['','','','']
group_counts = ["{0:0.0f}".format(value) for value in
               matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                    matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
         zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
```

Linear Regression on BOW

```
# Instead of using the vectorizer for tf-idf values, we now use countvectorizer
which converts the words to bag of words values
vectorizer = CountVectorizer(stop_words="english",preprocessor=clean_text,)
training_features = vectorizer.fit_transform(train_data["text"])
test_features = vectorizer.transform(test_data["text"])
```

```
# Training the linear regression model according to the BOW values
model = LinearRegression()
model.fit(training_features, train_data["sentiment"])
y_pred = model.predict(test_features)

#again, we use the same decision boundary where a predicted sentiment of over
0.5 is considered positive and below 0.5 is considered negative
for i in range(len(y_pred)):
 if y_pred[i]<0.5:
   y_pred[i]=0
 else:
   y_pred[i]=1

# Evaluation
acc = accuracy_score(test_data["sentiment"], y_pred)
print(acc*100)

#confusion matrix
matrix = confusion_matrix(test_data["sentiment"], y_pred)

group_names = ['','','','']
group_counts = ["{0:0.0f}".format(value) for value in
               matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                    matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
         zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
```

Logistic Regression on BOW

```
# Training the logistic regression model according the bag of words values
model = LogisticRegression()
model.fit(training_features, train_data["sentiment"])
y_pred = model.predict(test_features)
```

```
# Evaluation
acc = accuracy_score(test_data["sentiment"], y_pred)
print(acc*100)
#confusion matrix
matrix = confusion_matrix(test_data["sentiment"], y_pred)

group_names = ['','','','']
group_counts = ["{0:0.0f}".format(value) for value in
               matrix.flatten()]
group_percentages = ["{0:.2%}".format(value) for value in
                    matrix.flatten()/np.sum(matrix)]
labels = [f"{v1}\n{v2}\n{v3}" for v1, v2, v3 in
         zip(group_names,group_counts,group_percentages)]
labels = np.asarray(labels).reshape(2,2)
sns.heatmap(matrix, annot=labels, fmt='', cmap='Blues')
```