Programming for Data Analytics

POE

Kieran Glezer-Jones
ST10041889

June 2022

Contents

1 Introduction

Determining sentiment from text is something that would be extremely useful to a lot of companies. And a perfected model that can accurately detect a positive or negative response could help many companies grow.

The assignment demonstrates the actions taken to create a model that achieves this with a single dataset. The dataset was collected which hosts two important columns, a review column, and a rating column. From the rating column, we can grab the sentiment of the review text as most reviews paired with a rating of 1 and 2 would be negative and any review paired with a 4 and 5 would be positive. With this sentiment defined we can tokenize and clean the data before using it to train a model. After testing many different models and techniques a superior model and technique can then be chosen to predict the sentiment of real-life, never-before-seen, data. This process is predictive analysis.

The dataset in question contains reviews from a specific woman's clothing retailer. This retailer in question could gain a lot from a successful version of this model. An example of how they could use this model revolves around social media, such as Facebook, Twitter, and Instagram. Generally, the company would need to have a customer buy a product before they give feedback on it, but these platforms will allow the retailer to gain an idea of what people think of a product before they buy it. Let's say the company in question wants to experiment with a new line of clothing. They create a post for it and then post it on social media. After an amount of time, they can collect all the comments on the posts from these different platforms and gain sentiment from them. If the majority of the sentiments are positive, then it's safe to launch the product. However, if the majority of the sentiments are negative then a strategic cancellation would be the wiser move. It could also give the company an idea of what volume they can expect to sell based on how dominant the positive ratio is.

It becomes apparent quite quickly how being able to gain a sentiment of a product before it has even been rated is a good way to validate some difficult and risky business decisions. And with the power of more accurate decisions, a company stands to generate immense amounts of value with the model.

2 Background

This section explores the background of the report and looks at what is known beforehand.

2.1 The dataset

The dataset comes with quite a few columns to work with, but from the start, for this assignment, the only two variables that will be used within the dataset are the [Review Text] and [Rating] features.

The review text column is a qualitative column that holds the reviews from many different customers. This feature will be the main focus point of the assignment as we will manipulate it in such a way that an algorithm can eventually process it and conclude. There are 19541 unique reviews to work with.

The rating column can be used to give meaning to the review column. Since the aim of the model is to predict sentiment, we need to have some sentiment to initially train it. This column will allow us to do just that. The ratings are qualitative and ordinal, which allows us to group them into two groups, positive ratings, and negative ratings. After grouping them the column

can be converted into a one-hot binary feature which makes it ideal for classification algorithms.

We are left with some text in the [Review Text] column and the sentiment behind the text in the [Rating] column.

2.2 Text Classification

Classification is a type of supervised machine learning algorithm that will take an X variable and use it to predict a Y output. The important distinction between regression and classification is that the goal of regression is to output a continuous value while the goal of classification is to output a category. The output from a classification model can be numeric, but it will have to be a discrete variable.

When it comes to text however it becomes a bit more complicated. It's difficult to just input an entire string into an algorithm and expect it to give you an accurate model. There are many different meanings surrounding different words which our current algorithms can't handle. Therefore, a decent amount of cleaning and processing has to be done on the data before it can be inputted into an algorithm, such as stemming, lemmatization, and tokenization.

2.3 Descriptive Analysis

Descriptive analytics revolves around taking raw historical data and then data mining is to discover trends and relationships (The Independent Institute of Education, 2022).

There isn't too much of this to be done on text data, however, there might be certain words that come up quite often in a certain context that might be unexpected and provide decent insight into what the model sees and how to maybe manipulate it in a way that it performs better.

2.4 Sentiment Analysis

Sentiment analysis is a form of text mining at aims to gain an insight into people's opinions, emotions, and attitudes towards something (Medhat, et al., 2014). It is a type of predictive analysis which aims to predict sentiment.

The model being built will aim to ultimately take input text and predict the opinions and emotions associated with it. More specifically it will predict if the sentiment is positive or negative.

2.5 Predictive Analysis

Where predictive analysis is often said to take a large amount of historical data to find the patterns, trends, and relationships in an attempt to predict "what is going to happen next" (Cote, 2021). With Sentiment analysis, it's looked at a bit differently.

We are not predicting what will happen next necessarily. Instead, the model aims to offer a prediction of if it's positive or negative. That isn't to say, however, that the model can't be further used in both predictive and prescriptive analysis. As previously stated, the output from the model can be used to forecast whether something will sell well or not, and then prescribe an appropriate action.

The predictive analysis will be performed with the addition of a classification algorithm and model that will take text input and predict whether the sentiment was positive or negative.

2.6 Useability

Ultimately a classification model will be used where data will be used to train both the input and the output. It will ultimately produce either an 0 or a 1 for positive and negative respectively. To make sure the dataset is appropriate for this process a few things need to be considered.

2.6.1 Legitimate Source

If the model is trained using made-up data that doesn't represent real-life scenarios, then it is very unlikely that the model will perform accurately in the real world. This is because it will likely suffer from underfitting or overfitting. For the model to accurately get the sentiment from real-life data, it needs to be trained on real-life data. The way different people speak, and their mannerisms need to be considered and data constructed from a single person or algorithm is unlikely to include these differences.

The data was collected from real-life reviews from a specific retailer and many different people. They all include different mannerisms and words, and what's even better is that the model will be constructed for a clothing company. Therefore, all the reviews will be in a clothing company context. This means that certain words which have multiple meanings are unlikely to have meaning outside the clothing context. An example is that the word brace will very likely only be trained under the assumption that it's a piece of clothing.

2.6.2 Categoric Target

There needs to be an output that the model can understand. Random text can't just be thrown at the model in hopes that it will tell you if it's positive or negative reliably. It can be attempted with Latent Dirichlet Allocation, but it's not very reliable. Instead, an appropriate categoric variable has to be supplied so that the algorithm knows what it's working towards when you train it.

The dataset in question comes with a rating variable which can be converted into either a 0 or a 1 for positive and negative. Ratings 1, 2, and 3 and be considered negative while ratings 4 and 5 can be considered positive. Therefore, we have an appropriate 1 hot binary variable that can be used as the output.

2.6.3 Consistent

The text needs to be in a single language for the best results. This is especially important for pre-processing the data with concepts like lemmatization and stop words which take the language of the word into account. Different languages would also reduce the accuracy of the model as now you're introducing a completely new set of possible words with possible meanings that may overlap with the English words the model will be using.

If not all the records, then at the very least the vast majority of the records are in English as the company is presumably an English company in an English-speaking country.

2.6.4 Enough Data

Too few records might result in the model not being able to grasp the real trend. The model is likely being trained on a sample of the full population. This means that a sample size that is too small might not accurately represent the full population. Thus, your model may not perform well in production or even testing.

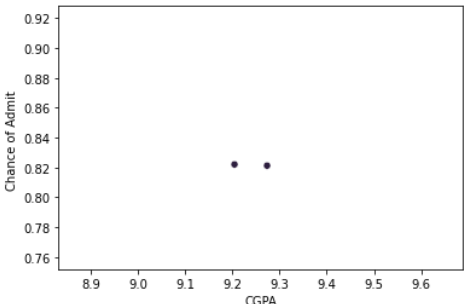

Figure 1 Too little data on a scatterplot

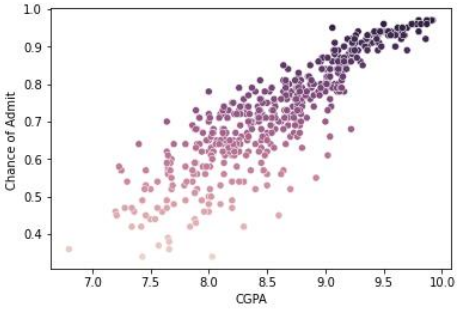

Figure 2 Enough data on a scatterplot

For this assignment specifically, there needs to be enough reviews to correctly get an idea of how each word should be weighted. Without enough records, certain words which have a negative connotation, such as the word hate, could appear in a single good review along the lines of 'I hate how much I love this'. In this instance, the word hate will be weighted as positive.

Luckily, however, the dataset comes with over 20 thousand records and reviews to be analysed. 5000 of which would be set as negative. Therefore, there are enough negative and positive records.

3 Pre-processing

The following section dictates the steps taken to process data that will ultimately be used during the predictive analysis phase of the assignment.

3.1 Data Collection and Cleaning

The first step of the process is to collect data and make sure it's clean and ready to be fed into the model. Any null or incorrect values need to be fixed or dropped, as they make create inconsistencies within the data which might throw off the accuracy.

We start by collecting the data from the dataset in question. For this assignment, the dataset is the Women's E-Commerce Clothing Reviews dataset from Kaggle. The CSV for the dataset is downloaded and loaded into the notebook using the panda's library. Irrelevant columns are then dropped so that we are left with the Review Text and the Rating features.

We can create a matrix that will visually show us if there are any null records within the dataset.

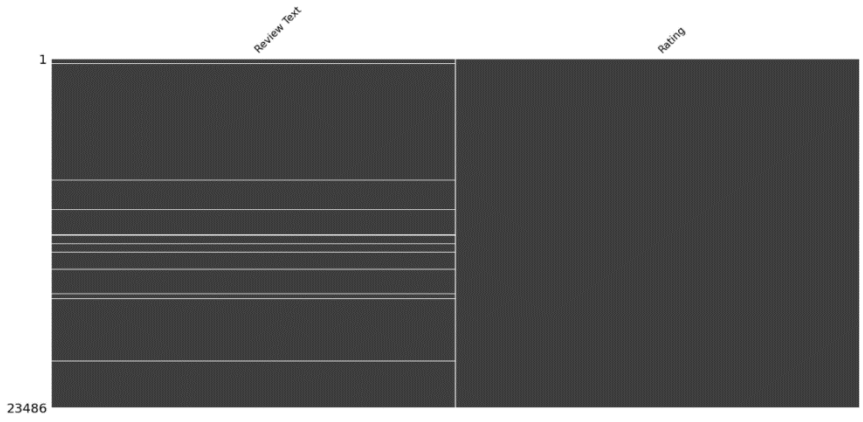

Figure 3 Visualising Null Values in the DataFrame

This figure is useful because it not only shows us if there are any null values in either of the records, but it also shows if there are any groupings between them. Assuming the dataset is in chronological order, it can show if there was a period where null values were able to be entered, this could indicate that something was wrong during that time. What we get from the figure above however is that there is a spread of missing values, which indicates that there is a way to input null values into the review field. These null values will be dropped.

After that, we can get information about the DataFrame and convert any object types into string types.

#	Column	Non-Null Count	Dtype
0	Review Text	22641 non-null	string
1	Rating	22641 non-null	int64

Figure 4 Info on the Data

We see that we are working with 22641 records, which is a bit less than the 23486 records we started with but is still enough to accurately train a specialized model. The string will be used as input and the int will be used as the output.

```
<b>I love, love, love!</b> this jumpsuit. it's fun, flirty, and fabulous! <u>every time i wear it</u>
```

Figure 5 Raw text

The above record in the dataset shows a few things that need to be removed from the dataset. We need to remove HTML elements, punctuations, capital letters, and stop words. Stop words that contribute nothing towards the meaning of the sentence, such as ‘I’, ‘him’, or ‘she’. These words are very common and would also throw off the weightings of our model, so we get rid of them. It’s not too important to do this step here, as when we apply lemmatization or stemming, we can pass a list of stop words we want to ignore, but this process is done now for demonstration purposes. The result is as follows.

```
love love love jumpsuit fun flirty fabulous every time
```

Figure 6 Processed Text

We can now take a look at how the ratings have been distributed to conclude what we need to do with reviews that have a rating of 3.

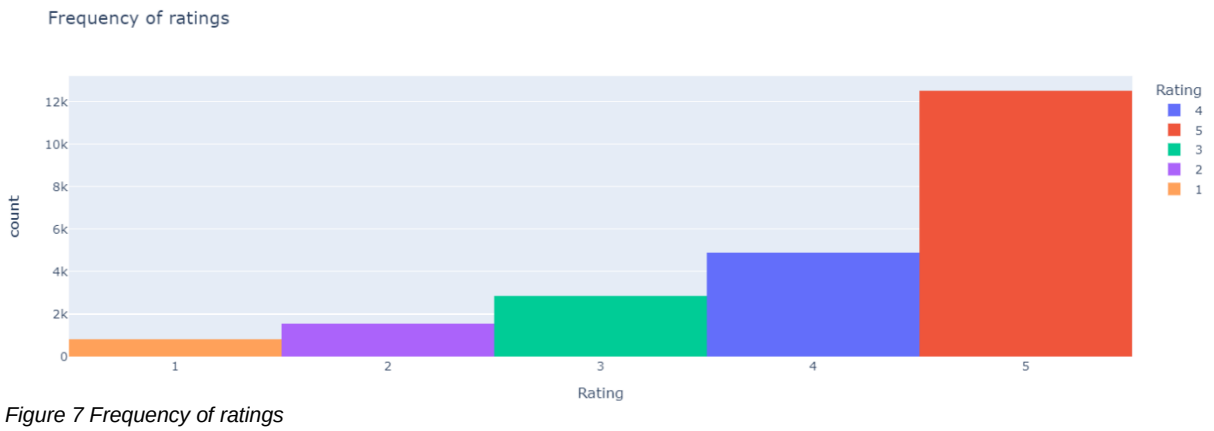

Figure 7 Frequency of ratings

We see that the products are generally quite highly rated as there are quite a few more positive records of 4 and 5 compared to the rest. The problem is deciding what to do with records that have a rating of 3. The first reaction would be to disregard all records that have a rating of 3 as they are typically neutral reviews that didn't love or hate the product. However, we are then left with way too few negative reviews, and in testing, the recall of the negative reviews was extremely low, SMOTE was also applied to balance them, but the accuracy didn't improve. When 3 was taken as a negative review, however, the recall of the negative reviews increased significantly, which shows that the majority of the 3-star reviews don't have many nice things to say either.

Therefore, ratings of 1,2, and 3 are converted to a 0 for negative, and ratings of 4 and 5 are converted to a 1 for positive.

3.2 Descriptive Analysis and Visualising the Data

The descriptive analysis that will be done on the data will be to attempt to find which words are used more commonly for negative and positive reviews respectively. To do this two-word clouds will be created, one for positive reviews and one for negative reviews. They will show us how frequently certain words appear for each of the sentiments. This way we will know which words hold the highest weight in the algorithms. This step should likely be done after lemmatization as well further down the analysis process.

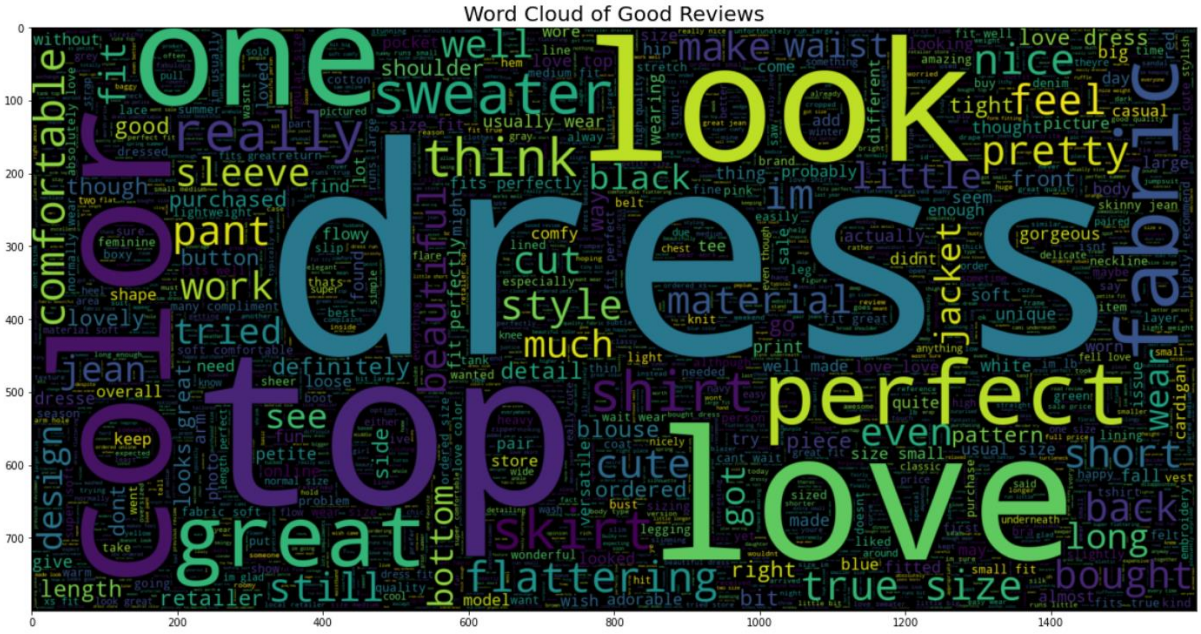

Figure 8 Word cloud of good reviews

We see words like 'dress' and 'top' taking up a large portion of the words, while more notable words such as 'love', 'perfect', and 'great' are also quite notable in the positive reviews. 'Color' and 'look' are also quite frequent, which leads us to believe that the majority of the positive reviews are based on their aesthetic.

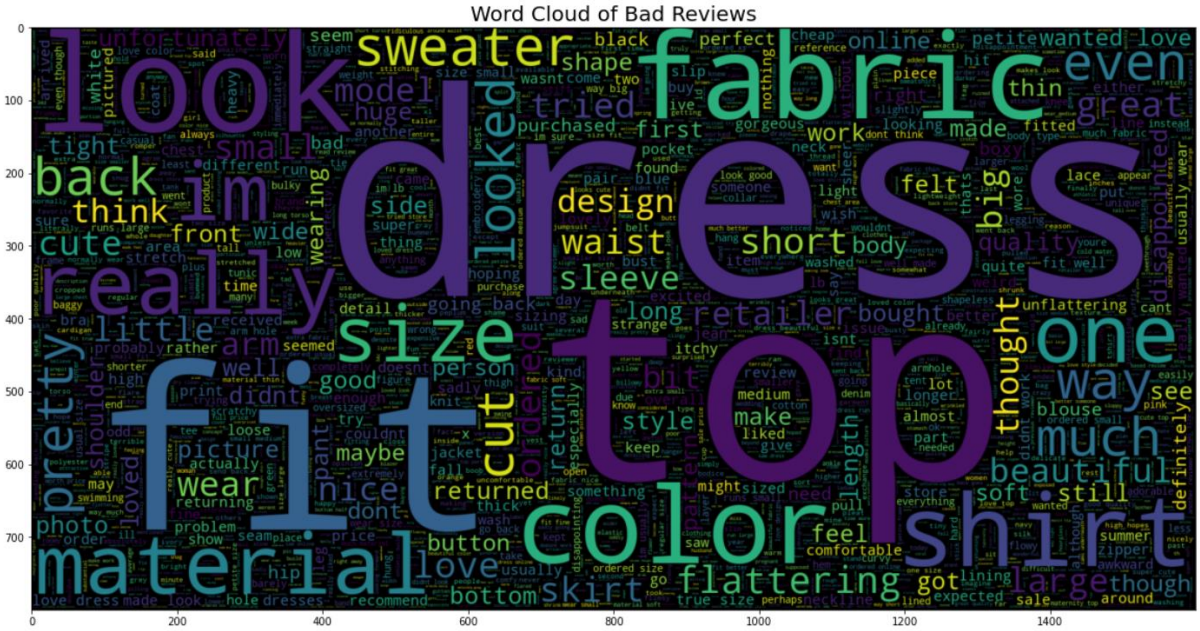

Figure 9 Word cloud of bad reviews

For the bad reviews, we also see ‘dress’ and ‘top’ being very present but interestingly enough we see the word ‘fit’ quite often in the negative reviews. This means it’s quite likely that the majority of negative reviews are due to the clothing not fitting properly.

We conclude that the large majority of positive votes are based on how much people loved the look of the clothing while a large majority of the negative votes are a result of the clothing not fitting nicely.

3.3 Tokenization

Tokenization refers to taking raw data and breaking it up into smaller pieces (Webster & Kit, 1992). For our dataset, it will involve taking the raw reviews from each of the columns and then breaking them up into individual words. An example of this process is where we take the words from the record demonstrated in section 3.1 and tokenize it so it outputs the following.

```
['love', 'love', 'love', 'jumpsuit', 'fun', 'flirty', 'fabulous', 'every', 'time']
```

Figure 10 After tokenization

We see that it returns a list of the words in the review. This is important and at the core of both the bag of words model and the term frequency-inverse document frequency method, which use this concept to convert the words into a numerical form. We do not have to manually tokenize the data as these functions will do it for us.

3.4 Splitting the Data

The data needs to be split into an X variable and a Y variable. The X variable will be the Review Text which has been processed in the cleaning phase and the Y variable will be the sentiment which is either a 0 or a 1. Once these variables have been created, they can now be split into training and testing data. 80% of the records will be segregated into data which will be used to train the model, and the remaining 20% of the data will be kept away from the

model until it's time to test it. It's important to note that the testing data will not be used during cross-validation as we don't want to validate the model with data that it has seen before.

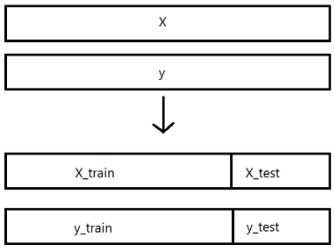

Figure 11 Splitting Data

3.5 Creating Pipelines

Pipelines are a great way to save time and make the process of handling data significantly easier. They link together multiple steps, each takes data, manipulates it, and then passes that manipulated data to the next step. An obvious advantage is that it doesn't change the data you put into it. Any cleaning processes can take place within the pipeline and not affect the original dataset. Custom components can be created for the pipeline by making a class that has a fit and transform method.

The below headings are examples of processes that will be used inside a pipeline along with the actual model.

3.5.1 Stemming

Stemming involves reducing all the words with the same root, down to their common form. This is commonly done by removing the derivational and inflectional suffixes from the words (Lovins, 1968).

We can see this in action when we apply it to our training data.

```
bought shirt midjuly washed twice cold delicat...
paigepilcro lover years mcguire jeans went sal...
runs almost two sizes small plan wear jacket s...
gorgeous pattern even prettier person runs sli...
love top tried store loved great transitional ...
```

Figure 12 Before Stemming

```
bought shirt midjuli wash twice cold delic hun...
paigepilcro lover year mcguir jean went sale l...
run almost two size small plan wear jacket sle...
gorgeou pattern even prettier person run sligh...
love top tri store love great transit piec ric...
```

Figure 13 After stemming

We see some drastic changes to words, such as 'July' becoming 'Juli' and 'delicate' becoming 'delic'. From this we also see why it would be so important to make sure that any further data which is used as an X should be stemmed before the model sees it, this is to data consistent enough to receive a decent accuracy.

3.5.2 Lemmatization

Lemmatization is a more realistic approach to stemming, where it works on a very similar concept, however, instead of breaking the word down into its stem, it instead breaks it down into a normalized version of the word. An example of this is that stemming would take the words ‘computes’, ‘computing’, and ‘computed’ and break them down to be ‘comput’. Where stemming will take the normalized version of the word which is ‘compute’ (Plisson, et al., 2004).

Another benefit of lemmatization is that before it breaks the word down, it will first take the dictionary meaning of a word and converts the word to its base version of it. We can see this below.

Figure 14 Before and after Lemmatization

We see that the word corpora get converted to corpus while the word better get converted to well. This method is preferred over stemming as not only does it reduces the number of words the algorithm needs to know, which stemming does as well but it considers the concept of the word as well and covers many possible words down to one word which means the same thing. This further reduces the number of words the algorithm needs to know and makes it a lot more reliable in production when it’s likely people will use words the algorithm has never seen before.

Therefore, if the algorithm was trained with the word ‘well’, and then someone typed they were feeling ‘better’, without lemmatization this word would hold no weight for the algorithm, but with lemmatization ‘better’ gets converted to ‘well’ and holds weight.

3.5.3 Bag of Words

The bag of words model involves taking multiple sentences of records and breaking the words present into a matrix that looks at all the possible words given if they were present in a sentence, and its frequency (Zhang, et al., 2010).

After creating the matrix the words are then tallied and put into a ‘bag’ where they have the word and its frequency. Words with a higher frequency will hold more weight. If we take two records from the dataset we can display this.

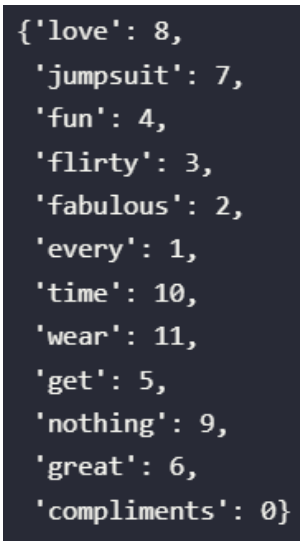

Figure 15 Bag of words bag

This is a technique to help the algorithm work with data. Since it isn't capable of making a prediction based on text, it instead passes the weight to the formula which is something it can work with. It produces a vector that the model can work with.

3.5.4 Term Frequency – Inverse Document Frequency

In direct competition to the Bag of Words model, the TF-IDF model not only looks at the frequency of the words but also looks at the rarity of them too. It looks at the term frequency and the words as well as the inverse of the number of documents (Sewwandi, 2019).

The output is comparable to the bag of words model where you get a vector that the model can work with. The difference is that the importance of a word is also considered.

3.5.5 The model

The data will be sent through the pipeline and will go through a combination of these methods to be processed and eventually sent to the model which will take the vector as an input. The model to highlight will be the Naïve Bayes model which is known to do quite well when it comes to text-based classification.

4 Sentiment Analysis

The following section highlights the steps taken to create a model which will take raw text and highlight the sentiment behind it. Sentiment Analysis is a type of predictive analysis where the goal is to predict sentiment from text.

4.1 Fitting the Model

To fit the model multiple pipelines were used to create models are varying structures. Certain models were fit with the Bag of Words concept while others fit with the TF-IDF. Within these distinctions, Stemming was compared to Lemmatization and other multiple different algorithms were used. This was to gain an understanding of exactly which combination worked best. The use of pipelines made this process significantly faster as the combination was as simple as putting the pieces together within them.

4.2 Overfitting and Underfitting considerations

Overfitting occurs when a model is too closely fit to the training data. This prevents the model from predicting accurately as it is affected by the noise in the data set. A underfit model cannot make accurate predictions as it cannot recognize the trend of the data (Muller & Guido, 2017).

4.2.1 Overfitting

It is quite difficult to avoid overfitting the data if the dataset contains too much bias. The naïve Bayes algorithm should not suffer from overfitting as it doesn't follow the trend of the data regardless. It independently makes assumptions based on each feature.

4.2.2 Underfitting

Bad data, to begin with, can cause underfitting. There needs to be a decent number of records to work with that contribute value to the model.

If you don't have enough data, then more needs to be collected. It's also likely that the data gets underfit if there are too few of the two possible target outcomes. This can be compensated with SMOTE.

Unclean data can also result in underfitting. Dropping too many null values may leave you with too few records. It might be beneficial to interpolate the missing data instead of dropping it.

Bad parameters can also cause underfitting, it's recommended to use hyper parameters if you can try and find the best possible fit.

4.2.2.1 Testing for overfitting or underfitting

It's easy to test for overfitting or underfitting. You can split the dataset into training and testing data. After that, you segregate the testing data and only use the training data to train the model. After the model is constructed, you can introduce the testing data to validate it. This will simulate real-life under the condition that the entire dataset isn't too clean. If the whole dataset is too clean, then human intuition is required.

4.3 Analysing and Verifying the Results

Verifying the data will be done using a concept similar to the k-folds validation concept. This breaks the dataset into k number of folds and can then iterate k times through the data, each time retraining the model with a different combination of folds and the 1 remaining fold to test it. This reduces the worry that the output accuracy was a fluke. The testing data previously created will not be used in the validation technique as it would just be testing the data that was already seen before. Instead, the testing data will be used to test the model later on, after it has been completed. The k-folds technique to validate that the data isn't random between accuracy outputs.

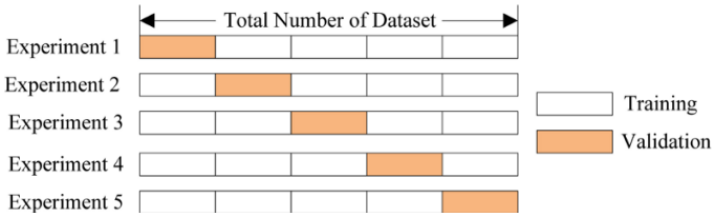

Figure 16 K-folds method (Koehrsen, 2018)

After all the iterations have been run, you can get an average score that will correctly represent the consistent accuracy. We will not manually write the logic for the iterations, we will instead use the cross_val_score() method from the sci-kit learn library which does the same thing automatically for you. The cross_val_score() library shuffles your data before you run it as well.

The pipelines of the different combinations will be fed into the cross_val_score() method to see which combination performed the best. The first and most consistent algorithm to test is the Naïve Bayes algorithm which has the following outputs from the different combinations.

```
------------------------- BAG OF WORDS MODEL-------------------------
With no stemming or lemmatization:
[0.86092715 0.84547461 0.84547461 0.85651214 0.85966851] Average : 0.854

With Stemming:
[0.86092715 0.84216336 0.85209713 0.85651214 0.84861878] Average : 0.852

With Lemmatization:
[0.85761589 0.84326711 0.84437086 0.85430464 0.85082873] Average : 0.85

With Both Lemmatization and Stemming:
[0.86313466 0.84216336 0.84878587 0.85651214 0.84861878] Average : 0.852
------------------------- TF-IDF MODEL -------------------------
With no stemming or lemmatization:
[0.76600442 0.76490066 0.76600442 0.76379691 0.76464088] Average : 0.765

With Stemming:
[0.76821192 0.76490066 0.76600442 0.76710817 0.7679558 ] Average : 0.767

With Lemmatization:
[0.76821192 0.76490066 0.76710817 0.76600442 0.76353591] Average : 0.766

With Both Lemmatization and Stemming:
[0.76821192 0.76490066 0.76600442 0.76600442 0.76906077] Average : 0.767
```

Figure 17 Accuracy from Naive Bayes

We see the Bag of Words model is performing significantly better than the TF-IDF model and there is little to no difference between Lemmatization and Stemming. However, the accuracy scores could be deceiving. Only 30% of the records in the training data were negative reviews, so if the algorithm predicted positive for all the reviews, even the negative ones, then it would still get a 70% accuracy. The cross-validation scores only showed us that the scores are very consistent, to get a better idea of how reliable the model is, we can print the classification report and confusion matrix.

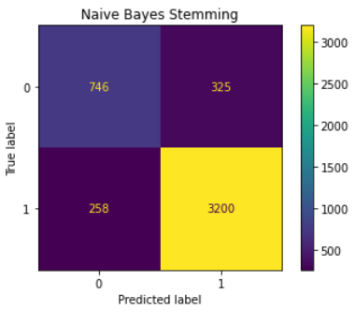

Figure 19 Confusion Matrix for stemming

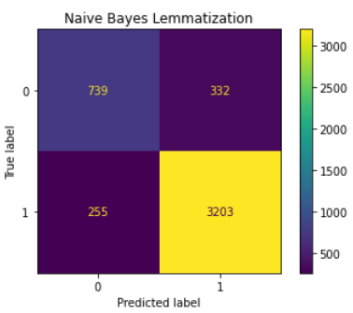

Figure 18 Confusion Matrix for Lemmatization

```
----------------
Stemming report:
----------------
              precision    recall  f1-score   support

           0       0.74      0.70      0.72      1071
           1       0.91      0.93      0.92      3458

    accuracy                           0.87      4529
   macro avg       0.83      0.81      0.82      4529
weighted avg       0.87      0.87      0.87      4529


----------------
Lem Report:
----------------
              precision    recall  f1-score   support

           0       0.74      0.69      0.72      1071
           1       0.91      0.93      0.92      3458

    accuracy                           0.87      4529
   macro avg       0.82      0.81      0.82      4529
weighted avg       0.87      0.87      0.87      4529
```

Figure 20 Classification Report

We see almost no difference between Stemming and Lemmatization. In this instance, we would use Lemmatisation for the remainder of the model, as the lack of difference is likely due to the data being too clean and not giving Lemmatization any opportunity to shine. However, in production, the inputs will be a lot messier and less consistent which will allow lemmatization to truly come out ahead.

The other noticeable thing about the report and classification matrix is that the recall and precision of the negative reviews are a bit low. This could be due to a lack of negative records. The accuracy overall is quite good due to the algorithm picking positive excessively and getting a good score by default since the chance of the record being positive is so much higher than the chance of it being negative. To fix this imbalance we can try using SMOTE, which is a method that will look and your dataset and procedurally generate more of the minority data. The results from the SMOTE are as follows.

```
-----------
Stemming:
-----------
              precision    recall  f1-score   support

           0       0.32      0.44      0.37      1071
           1       0.80      0.71      0.75      3458

    accuracy                           0.64      4529
   macro avg       0.56      0.57      0.56      4529
weighted avg       0.69      0.64      0.66      4529


-----------
Lem Report:
-----------
              precision    recall  f1-score   support

           0       0.31      0.44      0.36      1071
           1       0.80      0.69      0.74      3458

    accuracy                           0.63      4529
   macro avg       0.55      0.57      0.55      4529
weighted avg       0.68      0.63      0.65      4529
```

Figure 21 Accuracy after SMOTE

The ability to call the negative data got significantly worse after introducing generated negative reviews. Therefore, we won't use SMOTE for the remainder of the assignment.

Logistic regression, Random Forest Classification, and Linear Support Vector Machines were tested and documented in the Jupyter Notebook file provided. Many other algorithms such as K-nearest neighbour and decision trees were tested, however, their scores were so bad that they would just waste processing time in the notebook, so they weren't included.

4.4 Improving the Model

To improve the model, the concept of hyper parameters was used to get the best-known parameters for the model. You supply the function with a list of parameters you wish to test, and it will return the parameters which scored the best. We can use Logistic Regression to demonstrate this.

```
The Best Parameters for Logistic Regression are:

  C = 0.1
  max_iter = 1000
```

Figure 22 Best Parameters for Logistic Regression

Unfortunately, the naïve Bayes class doesn't come with many possible parameters to use and the dataset itself doesn't seem to benefit much from the process.

	precision	recall	f1-score	support
0	0.73	0.66	0.70	1071
1	0.90	0.93	0.91	3458
accuracy			0.86	4529
macro avg	0.82	0.79	0.80	4529
weighted avg	0.86	0.86	0.86	4529

Figure 24 Before Hyper Parameters

	precision	recall	f1-score	support
0	0.76	0.63	0.69	1071
1	0.89	0.94	0.91	3458
accuracy			0.87	4529
macro avg	0.83	0.79	0.80	4529
weighted avg	0.86	0.87	0.86	4529

Figure 23 After Hyper Parameters

We see that after hyper parameters, the accuracy of the model went up by 1%, but this was at the cost of the recall of 0. This is the flaw in the hyper parameter system, it chases the highest possible accuracy, which gets difficult when there is an imbalance between 0s and 1s. An accuracy of 70% where both 0s and 1s have a recall of 70% will perform much better in production compared to a model which got a 90% accuracy by only guessing 1s the whole time. An example is a failed attempt at the Random Forest hyper parameters.

	precision	recall	f1-score	support
0	0.00	0.00	0.00	1071
1	0.76	1.00	0.87	3458
accuracy			0.76	4529
macro avg	0.38	0.50	0.43	4529
weighted avg	0.58	0.76	0.66	4529

Figure 25 Random Forest Accuracy Demonstration.

We see that it got the best accuracy when it wasn't even predicting any 0s.

5 Latent Dirichlet Allocation

The dataset that was used for the application was very convenient in the sense that it allowed us to break down reviews into good or bad based on the rating. However, this is not likely to always be the case. In an instance where only text is available with no way of distinguishing them, the Latent Dirichlet Allocation (LDA) can be attempted.

LDA is an unsupervised topic model that attempts to break down the data into a n number of topics that can be recognized and assigned (Blei, et al., 2003). To demonstrate this, we can run through a quick version of the assignment where we use only the [Review Text] feature.

We fit the feature into the LDA model and specify that we want two topics. The hope is that it breaks it down into good and bad. We can generate the most frequent words that seem to be present in each of the topics.

```
Topic 1:
['size', 'dress', 'like', 'fit', 'im', 'small', 'fabric', 'love', 'really', 'ordered', 'little', 'look', 'large', 'color', 'bit', 'petite', 'wear',
'beautiful', 'shirt']


Topic 2:
['love', 'dress', 'wear', 'great', 'fit', 'perfect', 'size', 'color', 'comfortable', 'like', 'jeans', 'im', 'bought', 'cute', 'soft', 'flattering',
'look', 'pants', 'fabric']
```

Figure 26 LDA Topics

The topics aren't very distinguishable, but we see words like 'size' and 'fit' are very present in the first topic. Earlier we concluded that clothing not fitting correctly is the primary reason for a bad review. This makes us believe there is a good chance that the first topic is negative. The second topic holds words such as 'love', 'great', and 'perfect', which indicates that the second topic is positive.

We can then get a ratio of how likely each review is to belong to a certain topic. From that ratio, we can programmatically supply each review with the highest likely topic. Now we are left with a [Review Text] column and a parallel binary column that hopefully holds the sentiment. We can test to see how accurately a model can guess the topic.

```
---------------
Stemming report:
---------------
                precision    recall  f1-score   support

           0       0.94      0.94      0.94      2338
           1       0.93      0.93      0.93      2191

    accuracy                           0.94      4529
   macro avg       0.94      0.94      0.94      4529
weighted avg       0.94      0.94      0.94      4529
```

Figure 27 Logistic Regression LDA accuracy

We see that the topics are quite consistent in the sense that the algorithm can seem to get a good idea of what's happening, but there is no guarantee that it's sentiment that the algorithm is predicting, it might be a completely different topic. To test how accurately it guessed the sentiment, we can test the correlation between the generated sentiment and the actual sentient.

```
0.00511696910707505
```

Figure 28 LDA correlation

We were looking for either a very high positive or negative correlation, instead, we got basically a 0 which means no correlation. Therefore, the LDA was unable to accurately distinguish between positive or negative sentiment. It likely guessed a completely different topic that we haven't identified.

6 Conclusion

Data was collected from thousands of reviews from a women's clothing shop. It was found that most of the negative reviews seem to be from women who didn't like the fit or feel of the clothing while the majority of the positive votes seem to be from women who liked the look of the clothing.

Sentiment analysis was then conducted to build a model which could accurately predict if the inputted text was positive or negative. Sentiment analysis is also a form of predictive analysis which is used to predict the sentiment of the text. It was found that the Bag of Words model was outperforming the TF-IDF model, so it was preferred. Stemming and Lemmatization both performed equally well, but lemmatization will be utilized in production as it has more benefits than stemming which might be realized with more drastic input.

Ultimately the model performed well, being able to predict both positive and negative sentiment, and can be used in the future to help make informed business decisions.

.

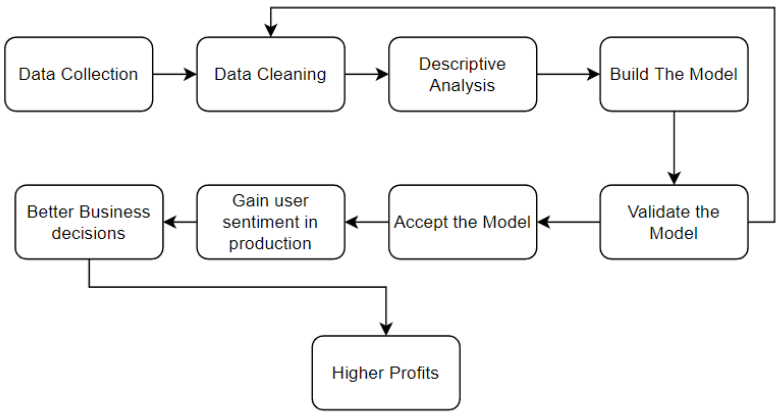

Figure 29 Sentiment Analysis Process

7 Recommendations

More negative data needs to be collected for the model to perform very well. There are a lot of problems with NLP that come into play, such as ambiguity or people using words with a negative connotation for positive reviews. An example of this is 'I hate how much I can't live with this now'. With more negative records it will become easier for the model to accurately distinguish between the two options. SMOTE attempted to compensate for the imbalance but failed to grasp the trend of the data.

As for how useful the model is, the best use for the model would be to show customers a new product on social media, collect the comments and tally how many were good and how many were bad. If the vast majority of comments are good, then you know that product will do well.

If it's pretty close then an appropriate business decision can be made, but it's almost certain that it's not worth heavily investing in.

The model, however, does have a few flaws, the most notable one is that it struggles to recognise negative reviews. Another notable flaw is that most of the negative reviews came from people not liking the feel of the clothing. These kinds of reviews are unlikely to appear on social media when only a picture is displayed.

To compensate for this the company should look at taking appropriate measures to prevent any issues with size or feel and make it a core process.

Lastly, the model can keep getting trained on new data that it receives so that it constantly gets better and better.

8 References

Blei, D., Ng, A. & Jordan, M., 2003. Latent dirichlet allocation.. *Journal of machine Learning research,* Issue 993-1022, p. 3.

Cote, C., 2021. *Predictive analytics.* [Online]
Available at: https://online.hbs.edu/blog/post/predictive-analytics
[Accessed 22 April 2022].

Koehrsen, W., 2018. Overfitting vs. underfitting: A complete example. *Towards Data Science.*

Lovins, J., 1968. Development of a stemming algorithm.. *Mech. Transl. Comput. Linguistics,* 11(2), pp. 22-31.

Medhat, W., Hassan, A. & Korashy, H., 2014. Sentiment analysis algorithms and applications: A survey. *Ain Shams engineering journal,* 5(4), pp. 1093-1113.

Muller, A. & Guido, S., 2017. *Introduction to Machine Learning with Python.* 1st ed. United States of America: O'Reilly.

Plisson, J., Lavrac, N. & Mladenic, D., 2004. A rule based approach to word lemmatization. *Proceedings of IS,* Volume 3, pp. 83-86.

Sewwandi, U., 2019. *bow vs tfid in information retrieval.* [Online]
Available at: https://medium.com/@sewwandikaus.13/bow-vs-tf-idf-in-information-retrieval-a325b5e61984
[Accessed 27 June 2022].

The Independent Institute of Education, 2022. *Data Analytics Module Manual ,* s.l.: s.n.

Webster, J. & Kit, C., 1992. Tokenization as the initial phase in NLP. *COLING 1992,* Volume 4.

Zhang, Y., Jin, R. & Zhou, H., 2010. Understanding bag-of-words model: a statistical framework. *International journal of machine learning and cybernetics,* 1(1), pp. 43-52.