Understanding the Sentiment and moods of the Customer Reviews in Restaurants using Natural Language Processing and Machine learning Techniques

Author: Sundeep Vemulapally

Introduction and overview of the issue

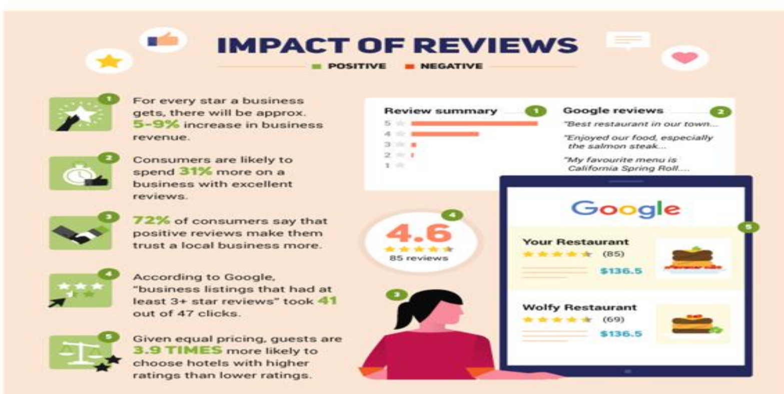
An overview of the issue

In the modern landscape, the food industry is playing a key role not only in the economy but also in the lives of the people with the quality of food being served on their plates. Customers also play an equally important role in the restaurant industry. There has been significant change in this trend in recent years where customers are more inclined to give their views on social media platforms. It is observed that 71% of online users use platforms like Facebook, and Twitter to decide on the restaurant they visit(Dobrilla, 2022).

Identifying the issue on the World Wide Web

The reviews have a greater impact on the restaurants as shown in Figure 1(Arevalo, 2017). Review hosting platforms like YELP, Tripadvisor, Twitter and restaurant-ji have an impact on customers' sentiments. In our case, we have used the reviews from "**www**.***restaurant-website.com***" to understand the customer's moods and sentiments. According to author Dobrilla (2022), reviews with no positive sentiment can give the business an understanding of the areas to focus on.

Figure 1. Impact of reviews on Restaurants.



Note. Sourced from(Arevalo, 2017).

Applying the Machine Learning (ML) and Natural Language Processing (NLP) Techniques.

This invoked a question of whether modern ***Natural language Processing(NLP)*** techniques through the application of ML Algorithms can be applied to understand the mood and tastes of customers from the posted reviews through the application of methods like ***Latent Dirichlet Allocation (LDA)*** and ***RNN***-based sentiment analysis using ***FLAIR*** package. The authors Zibarzani et al. (2022) observed the reviews on TripAdvisor using a hybrid approach that utilises machine learning and the theoretical model to inspect the hotels' qualitative factors and customer satisfaction during the pandemic period.

The authors Lee et al. (2021), based on the reviews from the ***YELP***, applied ML algorithms like ***Xgboost***, ***RandomForest***, ***NLP*** toolkit, ***LDA*** and***TextBlob*** to calculate the sentiment scores on the customer reviews. The study found the features generated from user reviews can be more helpful than the user ratings and also noticed high-starred reviews having negative comments and low starred reviews having the positive comments.

Web Crawler Building and Other Considerations

website consumed for crawling

The website chosen for the crawling www.restaurant.com which hosts their reviews and ratings along with Google, Facebook, Tripadvisor and FourSqaure etc reviews. According to the website similarweb (2023), restaurant website is top website in the ***food and drink*** category.

The data used for building the model are ***user reviews***, ***ratings*** and relative ***Indian restaurants*** in the area of Dublin, California. Only the top 10 restaurant reviews are extracted which will be further discussed in the data section.

Copyrights consideration and other aspects during the crawling

The website copyright and privacy policy is checked and the data that is retained is NOT in violation of their policies and no user's personal information is retained knowingly. It was observed that a US Ninth Circuit verdict in the "***Clearview AI***" case relied on the Supreme Court ruling of "***gate-up, gate-down***" where data publicly accessible requires no authorisation (Whittaker, 2022). The website is checked for other hindrance aspects like "***Cloudflare***" and "***robots.txt***" and no restrictions about copyright were in place.

The methodology applied in the scraping and methods to store the harvested data.

One of the popular methods or tools used in web scraping is the use of ***Selenium***, ***scrapy*** and **Beautifulsoup** with **requests**. In our case, the website has some content that is dynamically generated on the clicks and page scrolling which cannot be handled with BeautifulSoup(ZenRows, 2023). Selenium is designed for automated testing and is used in retaining the information from the "***webdriver***" which is a cross-platform testing method that can control the browser remotely(Rungta, 2023).

Table 1 Chrome webdriver options

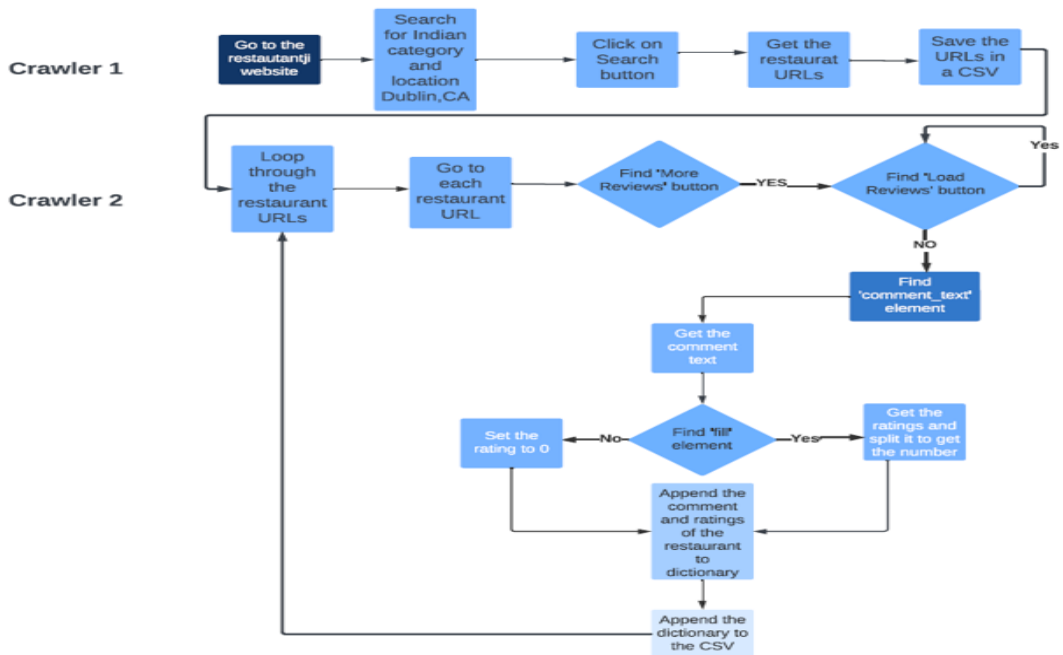
Chrome Webdriver options and their descriptions	
Option arguments	Description
--ignore-certificate-errors	Checking the acceptance of certificate
--incognito	Force the incognito mode, history will not be saved
--headless	Runs with out the UI and display server dependencies
-no-sandbox	Processes that are sandboxed will be disabled and required for headles mode
-disable-dev-shm-usage	If the development is small in VM will cause the chrome to fail or crash with error messages and will stop that
-disable-geolocation	Ignores the geo-popup which the site has enabled

Note. Sourced from (Beverloo, 2023).

The Python version used for the task is 3.9.16 with the Selenium package (version 4.8.3)along with the ***Chrome webdriver*** support. The ChromeDriver options chosen are shown in **Table 1**. Some of the options like "***—disable-geolocation***" are used to ignore the location pop-up in the website.

The data harvest was done through the use of two crawlers, one for harvesting the "***URLs***" for the restaurants and storing them in a CSV file and the other for retrieving the ***reviews, ratings, and restaurant names*** and storing them in a secondary CSV file for model building. The main data of interest is the reviews, and the other supplemented meta-data includes restaurant names and ratings which can be further used in the classification problems either by ratings or restaurant-specific topic modelling. The two crawlers' method was used to not stress the site. To mimic human interaction, a ***time.sleep()*** method is used. To overcome the pattern recognition by the site, the sleep times are random.

Figure 2. Flow Chart of the data harvested using the crawler.



WebDriverWait is also used to mimic the human behaviour to wait for the elements to be loaded. Some sections of the page are dynamically updated in the dropdown menu which are handled by the input keys like ***keys.ARROW_DOWN & Return.*** The elements in the page are handled by functions "***BY***" from selenium and ***XPATH*** or ***CLASS_NAME*** and so on. The flow chart of the crawler's methodology is shown in **Figure 2**. A "***try and exception***" block is used to catch the errors in crawlers. The crawler function is shown in ***code snippet 1***.

Crawler Limitations in the Domain

- The website's initial search selection is initiated which directs to a new page with a list of restaurants which makes the crawler useless, this might be related to checking or stopping the robots. This led to the creation of 2 crawlers one for URL retrieval and other for the retrieving reviews.
- Not every review has a rating, and this would lead the crawler to fail. To handle this issue, a value of "0" is assigned to the rating when an element is not found.
- Only the top 10 Indian restaurants are taken in the area, a prioritization of certain restaurants is given over the others which can induce bias in the models.

- The website construction is simple except for the geo-location bypassing, and auto-complete fields.
- The Crawler is run in multiple phases and during off-peak hours, to not stress the site. The program is not run continuously, to not get flagged or blocked.

Code Snippet 1:

```
def getrestaurantreview(restreview_url):
    """
    The function takes the restaurant URL as an argument and returns all the
    reviews, ratings and restaurant names as a list of dictionaries.
    :param restreview_url:
    :return: ls_rest-reviews (list of dictionaries)
    """

    #Setting Chrome Driver Options
    options = webdriver.ChromeOptions()
    options.add_argument('--ignore-certificate-errors')
    options.add_argument('--incognito')
    options.add_argument('--headless')
    options.add_argument('-no-sandbox')
    options.add_argument('-disable-dev-shm-usage')
    options.add_argument("--disable-geolocation")
    driver = webdriver.Chrome(executable_path='C:\Chrome
Driver\chromedriver_win32\chromedriver',
                              chrome_options=options)

    ls_rest_reviews = []
    ls_rest_dict = {}

    ##Getting the URL for the website to retrieve the comments
    time.sleep(2)
    #Getting the restaurant name
    restname = restreview_url.split("/")[5] .replace("-"," ")

    print("Getting reviews for "+restname)
    driver.get(restreview_url)
    print ("Went to the url")
    t_wt = random.randint(6,12)
    time.sleep(t_wt)
    wait = WebDriverWait(driver, 10)

    #Clicking More Reviews button
    driver.find_element(By.PARTIAL_LINK_TEXT,"More Reviews").click()
    print("Load more reviews clicked")
    time.sleep(7)

    # Click the Load More Reviews button if it exists
    while(True):
        try:
            # print ("Try for the more reviews button")
            button = wait.until(EC.element_to_be_clickable((By.ID,
'getMoreComments')))
            if(button):
```

```
                # print ("Review Button found in iteration")
                driver.execute_script("arguments[0].scrollIntoView();",
button)
                t_wait = random.randint(4,15)
                time.sleep(t_wait)
                driver.execute_script("arguments[0].click();", button)
                print ("Load Reviews button clicked")
                time.sleep(4)
            else:
                print ("No more reviews button found")
                break
        except Exception as e:
            # Exit the loop if the button doesn't exist
            print ("No button exists as there are no reviews to load.")
            break

    try:
        ## Getting the reviews
        rest_all_reviews = driver.find_elements(By.CLASS_NAME,"comment-
inner")
        print("Number of reviews:"+str(len(rest_all_reviews)))
        for rest_review in rest_all_reviews:
            #Getting the individual review
            review = rest_review.find_element(By.CLASS_NAME,"comment-text")

            #Getting the review's rating. If the element is not found, set
rating = 0
            try:
                rest_rating = rest_review.find_element(By.CLASS_NAME,"fill")
                rating =
rest_rating.get_attribute('style').split(":")[1].split("%;")[0]
            except:
                print("Rating element is not found. Setting the rating to
0.")
                rating = str(0)
                pass

            #Storing the reviews into a dictionary
            ls_rest_dict= {'restaurant_name': restname,
'restaurant_review':review.text,'rating':rating}
            ls_rest_reviews.append(ls_rest_dict)
    except Exception as exp:
        print(exp)
        pass

    t_wait1 = random.randint(3,16)
    time.sleep(t_wait1)
    # Quitting the chrome driver
    driver.quit()
    print("...Driver Quitted...")

    return ls_rest_reviews
```

Data Wrangling and Data Summarisation

Data Wrangling methods and Cleaned Corpus retainment

Figure 3. Shape of the data after removal of missing data

```
**************************************************
The shape of the restaurant review data: (2271, 3)
**************************************************
The number of reviews with no ratings: (44, 3)
***** The final shape of the data *****
Final Shape of the data: (2227, 3)
```

The data was loaded into the environment and checked for any missing values. The data has an initial dimension of ***2271 rows * 3 Columns*** of which there are some missing data of ***44*** rows that are removed from the analysis and are in the permissible range. The final data shape is shown in ***Figure 3***. The data is plotted for "reviews" distribution in restaurants and ratings. It is observed that the data is not uniformly

Figure 4. Reviews Distribution in Restaurants and Ratings

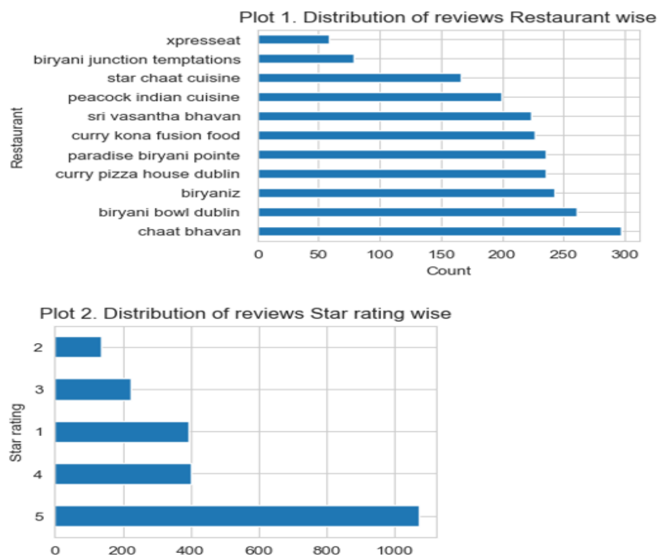


Figure 5. Descriptive statistics of the ratings

```
The descriptive statistics of the ratings
****************************************
count      5.000000
mean     445.400000
std      367.404818
min      138.000000
25%      223.000000
50%      394.000000
75%      401.000000
max     1071.000000
Name: count, dtype: float64
****************************************
The Skewness in the data: 1.7187106862615886
The Kurtosis in the data: 3.2890719426880715
```

distributed which can be seen in ***Figure 4*** and descriptive statistics show skewness in ***Figure 5***. The final data and the derived variables are shown in ***Table 2***.

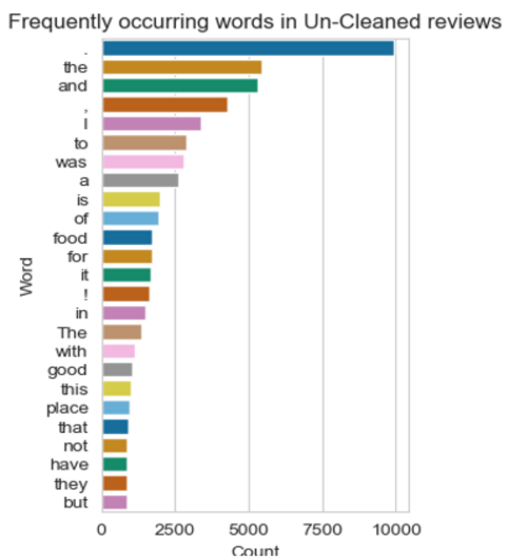
Table 2. Variable Names and Data types

Variable Names Data type	
Harvested Variable Names	**Description and data type**
restaurant_name	Categorical Nominal String-Object
reviews	Corpus = Bag of Words or TFIDF representation **Variable of interest**
ratings	Categorical Numeric representation
Generated data Variable Names	**Data Modified to fit needs**
star_rating	Categorical- multiple of 20 converted to star ratings
sentiment	Categorical – Dichotomous Calculated **response**
Sentiment_score	Numerical Continuous - float

Corpus Cleaning Text generation

The cleaning of the corpus is done utilizing the package ***spacy.*** The process involved is tokenization, normalising the text, removal of punctuation, lemmatising and recognising the Nouns and Verbs.

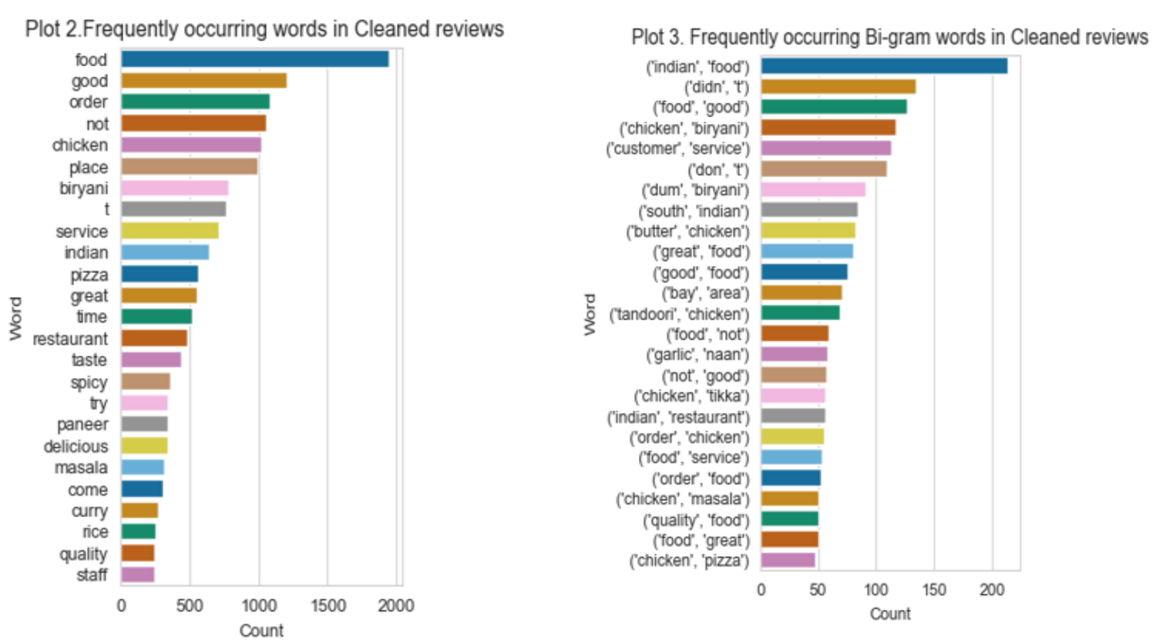
Figure 6. Frequently distributed words in Uncleaned Reviews



The Corpus contains many unwanted words or noise in the form of symbols or punctuation or frequent words like determiners. The Zipf law states that in natural language, the most frequent words occurring in the corpus are inversely proportional to their rank(Piantadosi, 2014). The “reviews” are plotted with the ***NLTK.FreqDist()*** function to understand the most frequent words which can induce noise in the model as shown in ***Figure 6***. There is harmful noise that can impact models and useful noise like the words “***didn’t or wouldn’t***” with punctuation in between can decide polarity scores when interpreting the reviews (Al Sharou et al., 2021).

The words like “***no and not***” are excluded from the stop words and lemmatized with the spacy inbuilt lemmatizer which used the rule-based method using pipeline (Honnibal et al., 2020).

Figure 7. Top 25 frequent words in cleaned corpus unigram and bigrams.



Rather than using Named Entity Recognition(***NER***), relations are extracted through ***chunking*** and part-of-speech (***POS***) tagging like ***Nouns(NN***), ***Verbs(VBG, VBN***) and ***Adjectives(JJ***)(Bird et al., 2009). The frequent words after cleaning are plotted through frequency plots and wordcloud shown in ***Figures 7*** and ***8***. The code is shown in ***code snippet 2***.

Figure 8. Word Cloud of the top 50 words in the Cleaned and Uncleaned reviews

Code Snippet 2:

```
def pos_lemtize(text, pos_tags =["NOUN", "PROPN",  "ADJ", "VERB", "ADV"]):

    """ Takes sentences, adds POS tagging
    and lemmatizes and returns the text.
     """
    # print(text)
    pos_tags = ["NNP", "NN", "VBG", "VBN", "VBD", "JJ", "RB"]
    stp_words = STOP_WORDS - set(['no', 'not'])
    nlp = spacy.load("en_core_web_sm", disable=['parser', 'ner']) # Loading
the spacy language for lemmatization.
    pos_text = [] # Initialising the pos_text
    for word in nlp(text): # A for loop to loop over the text "NLP".
        word_text = "" # Empty string
        # IF word is not in stop words or words is part of "Part of  Speech"
tags retain
        if word.lemma_ not in stp_words:
            if word.tag_ in pos_tags:    # Checking if the words exist in
Parts of speech tagging
                word_text = word.lemma_
                pos_text.append("".join(word_text))
        else:
            pos_text.append("".join(word_text))

    return pos_text
```

Feature extraction

The model building is done in two phases:

1. Topic modelling is done using ***LDA*** and Non-Negative Matrix Factorization(***NMF***). A ***Bag of Words*** approach is used with the entire corpus as a sample for training.
2. For the Classification model, Term Frequency to Inverse Document Frequency (***TFIDF***), the sample is split into ***80:20 stratified*** split between ***train*** and ***test*** data on the categorical sentiment variable discussed in the ML section. The stratified sampling has more statistical power as each class is a stratum and the sub-population is homogenously relative to the population(Simkus & Mcleod, 2023).

Bag of word (BOW) approach. In this approach, the document is represented in vector form and a **gensim** model is used to convert the dictionary representation by filtering words and making the model learn ***bi-gram words*** and collocations that occur using the **"phrases"** function. The phrases method learns the patterns out of the sequence of words using average log probability and softmax function to recognise the closely occurring vectors to the centre word and this would lead to higher accuracies in the prediction(Mikolov et al., 2013). The parameters are shown in ***Table 3***.

TFIDF approach. The term t occurring in document d is assigned with weight and the documents containing the term are inversely proportional and defined as $tf * idf(log\frac{N}{dft})$ (Manning et al., 2008). The TFIDF is trained by using the "***fit_transform***" on training data and "***transform***" on the test data to a sparse matrix. The hyperparameters are shown in ***Table 3***.

Table 3. Hyper-parameter and parameter tuning for the Phrase (BOW gensim) and TFIDF

Model and Hyper Parameters
Hyper-parameters for Gensim Bag-of-words and Phrases
Phrases • **Scoring:** A pointwise Mutual information (PMI) is chosen. Low-frequency bigrams that are more correlated a given more weights (Bouma, 2009) • **Min_count**: ignore the words below threshold (2) chosen. • **Threshold**: A lower score is chosen at 3 which would give more phrases and a higher score reduce phrases and is dependent on ***scoring***. Note. Sourced from gensim models documentation (Řehůřek & Sojka, 2010).
Hyper-parameter for the TFIDF model
Max_df: Terms above this threshold are ignored (0.8), a higher threshold can also induce noise into the model. **min_df:** The number of terms below the threshold are ignored, they do not carry information. (0.3) **max_features:** The number of features chosen or the document matrix feature representation or column space. **ngram_range:** chosen either bigram or unigram, the boundary of the collocation. (1,2) norm: ***"l2"*** is chosen which is based on the cosine similarity to handle the anomaly in the varying length corpus descriptions (Manning et al., 2008).

Note. For BOW sourced from gensim models documentation (Řehůřek & Sojka, 2010) and (Bouma, 2009).

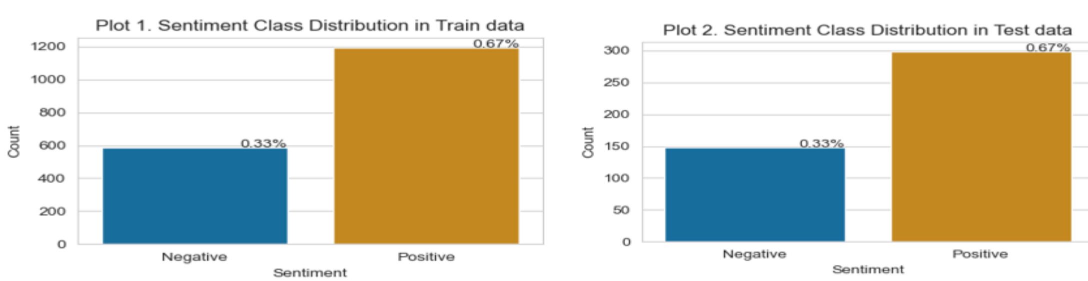
Note. The TFIDF model parameter is adopted from the sci-kit Learn (Pedregosa et al., 2011).

Bias in the data, corpus limitation and data distribution

The word distribution is checked in the sample, train, and test and all the data showed positive skewness and this could be partly attributed to the outliers in the data. As the data is from the real world, no artificial smoothing is done and the data is normalised using cosine based "***L2***" method in TFIDF and the words that fit in certain parameters are retained. The ***Plots from 2 to 4*** show the distribution and descriptive statistics as shown in ***Figure 9.***

Bias can be induced into the models in the form of data bias where the data is different from ideally collected data and selection bias where the users are free to rate their views (Chen et al., 2020). In our case, the data bias is in the form of the top 10 restaurants chosen in the crawler which has more positive reviews distribution. The ratings have some ambiguity where the users have given 5 stars with negative reviews. It is noticed that some of the 5-Star ratings had negative reviews and some 1&2 Star ratings had positive reviews. The sentiment analyzer with star ratings is plotted to understand the distribution of sentiment as shown in ***Figure 9***, ***Plot 1*** shows that high star rating also has negative ratings and some low star ratings had positive reviews discussed in the results section. The sentiment analyser can induce machine bias as it is based on the **BOW** approach where reviews are analysed based on the transformer method exposure to the reviews. The class distribution of sentiment is shown ***in Figure 10***. No methods like Synthetic Minority Over-Sampling Techniques (SMOTE) or Over-Sampling or Under-Sampling are employed which can further induce bias or uncertainty. The only limitation to the corpus is single words like "ok and great" or similar words that can induce noise into the data which are handled with feature normalisation.

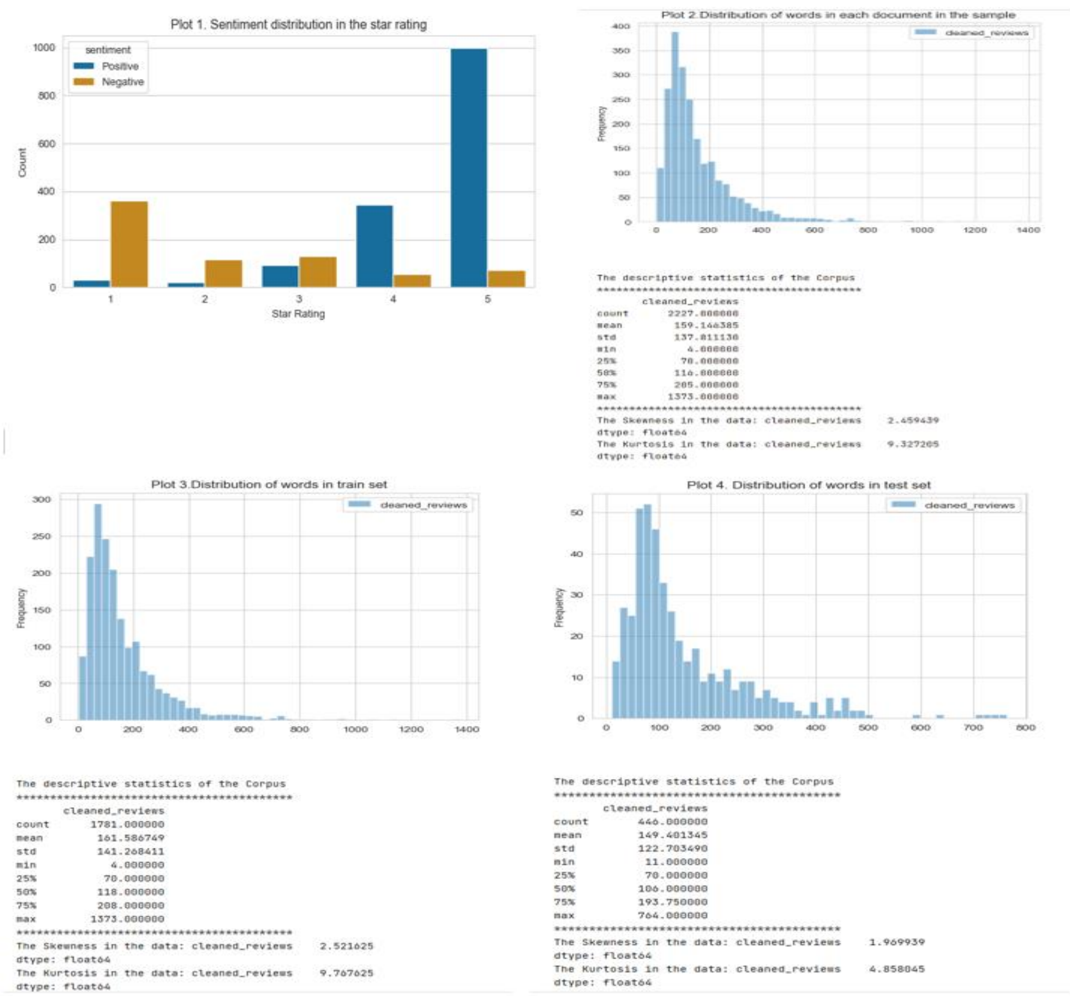
Figure 10. Class distribution of the Sentiment in Train and Test data




```
The distribution of classes and their counts
**********************************************
sentiment
Positive    1192
Negative     589
Name: count, dtype: int64
**********************************************
```

```
The distribution of classes and their counts
**********************************************
sentiment
Positive    298
Negative    148
Name: count, dtype: int64
**********************************************
```

Figure 9. Word Distribution in Sample, Train and Test. Sentiment Distribution in Ratings.



Machine learning model structure, Development and Evaluation

The machine learning aspect of the reviews is assessed in two parts:

- Topics modelling using probabilistic models like ***LDA*** and dimensionality reduction methods like ***NMF***.
- Assessing the sentiment using the transformer-based method using the ***RNN Flair package*** and assessing the key features in the ***XGB*** classifier through grid search method and assessed the model evaluation metrics.

Topic Modelling

The idea is that each document is made up of topics that are made up of words and is an automatic method of detecting topics from the data. The topic modelling or text similarity is done through the LDA and NMF rather than BERT or DOC2VEC as they might be a good performer on short topics like tweets and cannot perform on bigger documents with a mixture of multiple topics(Albanese, 2022). The topic modelling would give the restauranteurs an understanding of the topics based on customer reviews and their interests which can be either positive or negative representing a specific topic about the restaurants.

LDA, NMF modelling and hyper-parameter setting

LDA (Latent Dirichlet Allocation). is a probabilistic model where each item of the collection is modelled on the underlying topics, which are modelled as an infinite mixture of topic probabilities. The underlying idea is the documents which are a mixture of latent topics based on the underlying distribution of words (Blei et al., 2003, pp. 994-996).

The formula is mathematically denoted by:

$$p(\theta, z, W|\alpha, \beta) = p(\theta|\alpha) \prod_{n=1}^{N} p(z_n\,|\theta) p\left(w_n \middle| z_{n,}\beta\right)$$

Note. Equation sourced from(Blei et al., 2003, p. 997).

Where α and β are the parameters related to the apriori of the corpus. Variables θ and Z are related to the multinomial distribution of documents and words.

NMF (Non-negative Matrix Factorisation). is a dimensionality reduction method that recognises the latent topics representing the higher dimensionality matrix into lower dimensions matrixes as an approximation of the document term matrix represented by $M = WH$ (Helan & Sultani, 2023). Where W is word weightage matrix for each column and $H$is the row matrix of words embedded. The hyper-parameters of the two models are shown in ***Table 4.*** Both the models are trained on the entire corpus to produce topics.

Table 4. Hyper-parameter and parameter tuning for LDA and NMF

Model and Hyper Parameters
Hyper-parameters for LDA Model
Coprus: A sparse matrix representation of terms in the document. **Distributed:** used for multi-thread processing, not used due to inconsistent results despite seed setting. **Update_every:** Number of documents learned in each iteration 0 is chosen for batch learning. 1 is for online learning which can handle large document sets since our corpus is a small batch is chosen (Hoffman et al., 2010). (0= mini-batch method chosen) **Chunksize:** determines the documents processed in the algorithm, and influences the quality of the model based on the learning parameter (Hoffman et al., 2010, p. 7.). (100 = size) **Passes**: The number of times the algorithm experiences a line before converging. (20 passes) **Alpha and Eta:** The learning parameters of A-PRIORI based on the document and topic distribution. ***Note.*** Sourced from gensim models documentation (Řehůřek & Sojka, 2010).
Hyper-parameter for NMF Model
Eval_every: The number of batches after which the “l2” norm is calculated. (100) **Passes:** passes over the training set, chosen at 20.

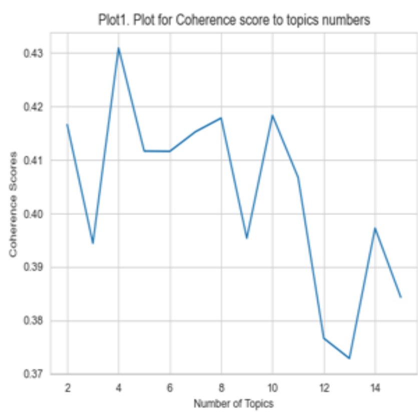
Kappa: The learning parameter and step size of the gradient descent chosen at 0.01, higher values can lead to non-convergence. ***Chunksize:*** the number of training documents used in each chunk. (50) ***Note.*** Sourced from gensim models documentation (Řehůřek & Sojka, 2010)

Topic modelling outputs and results discussion

The optimal models are retained based on the iterative approach where the topics are chosen based on the top coherence and perplexity score. A **perplexity** is used as a measure in modelling tasks for assessing the model geometric mean per word (information) and a lower score indicates a better model (Blei et al., 2003, p. 1008). The LDA model has a lower score(-5.87). For the NMF model, perplexity is not assessed and both models are assessed with a Coherence score for comparison. The models and scores are shown in ***Figure 11***. The code is shown in the ***code snippet 3*** and the same methodology is followed for NMF.

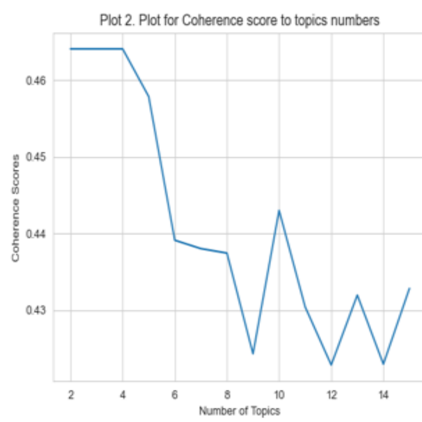
Fugure 11. Coherence Score and topic numbers

Plot 1. Coherence Scores to Topcis LDA




```
******************* Optimal model, number of topics and Coherence Score *******************

Topics needed to produce optimal model: 4
Topics Optimal Coherence score achieved for the model: 0.4309507886469897
*****************************************************************************************
```

Plot 2. Coherence Scores to Topics NMF




```
******************* Optimal model, number of topics and Coherence Score *******************

Topics needed to produce optimal model: 3
Topics Optimal Coherence score achieved for the model: 0.46405226454240406
*****************************************************************************************
```

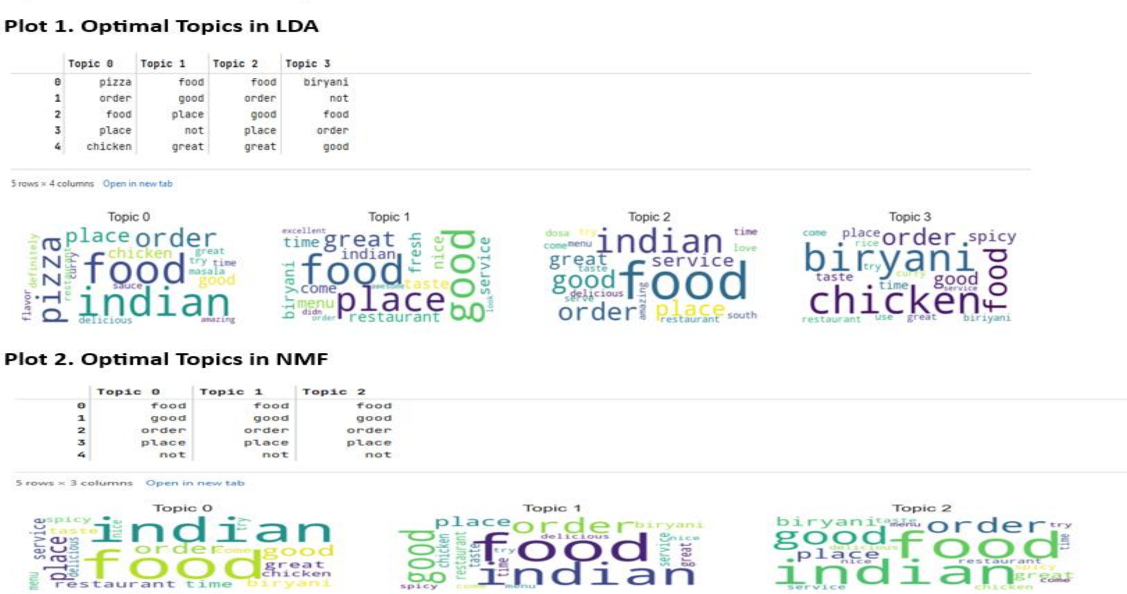
Code Snippet 3:

```
coherence_score = []
for n_topics in tqdm(range(str_num, lmt_num, step_num)):
    lda_model_optimal = gensim.models.LdaModel(corpus=corpus_text, # Cleaned corpus from the above code.
                          id2word=id2word,
                          num_topics=n_topics, # Assembling the words into topics.
                          random_state=172, # A set seed is done for the reproducibility of the code.
                          update_every=0, # The documents that are iterated and used is "mini-batch iterative learning".
```

```
                          # workers= 3, # number of parallel processes
                          chunksize=100, # documents processed in the algorithm.
                          passes=20, # Documents processed at an instance in the
algorithm
                          alpha='auto', # Learning parameters " A priori of the
document distribution".
                          eta= 'auto', # learning parameters "A priori of the word
distribution"
                          per_word_topics=True # Model computation of the topics
in descending order.
                                                        )
     coherence_scre_dict =
calc_coherence_score(optimal_model=lda_model_optimal,
                                                            id2word=id2word,
                                                            bigram_text = bigram_text,
                                                            coherence_typ= 'c_v',
                                                            n_topics = n_topics)
     coherence_score.append(coherence_scre_dict)

optimal_cohrnce_score_df = pd.DataFrame.from_dict(coherence_score)
```

From the Coherent scores, it is observed that NMF is marginally high when compared to LDA. When the word clouds in topics are assessed, they speak a different story about the topics descriptions as LDA can differentiate between topics and NMF failed to distinguish. The wordcloud for the topics is shown in ***Figure 12***. The partial failure of the results in NMF can be attributed to its robust nature of impact to handle the outliers in the mini-batch setting as noted in these articles(Stevens et al., 2012; Zhao & Tan, 2017). The LDA on the other hand is dependent on learning parameters α and β and forms topics from the words that co-exist frequently. These models also require an understanding of the number of topics in advance.

Figure 12. WordCloud of Topics from LDA and NMF



Plot 1. Optimal Topics in LDA

	Topic 0	Topic 1	Topic 2	Topic 3
0	pizza	food	food	biryani
1	order	good	order	not
2	food	place	good	food
3	place	not	place	order
4	chicken	great	great	good

5 rows × 4 columns Open in new tab

Plot 2. Optimal Topics in NMF

	Topic 0	Topic 1	Topic 2
0	food	food	food
1	good	good	good
2	order	order	order
3	place	place	place
4	not	not	not

5 rows × 3 columns Open in new tab

It can be inferred from ***Plot 1***, LDA that there is some distinction in topics (topic 1 about pizza ordering, topics 2 and 3 about the place and topic 4 about Biryani) and the distinction is not seen in NMF. pyLDAvis is shown in ***Appendix A, Figure 1***.

Sentiment Analysis and Classification with XGBoost

Flair sentiment analyser

Sentiment analysis is used to understand the customer's views with the use of NLP and polarity scores of the given text. In our case, we used the ***Flair*** package sentiment classifier which is based on the Long-short-term-memory (LSTM) neural network model which projects itself marginally better than the ***Google Auto ML*** model by the authors (Akbik et al., 2019; Magajna, 2021). The polarity score of the sentences is either positive or negative and not neutral which makes the classification problem dichotomous.

The model is also a pre-trained model on IMDB and Amazon datasets which also use transformer wrappers. Polarity scores and classification is done on entire cleaned reviews and compared to ratings as shown in ***Figure 9, plot 1***. The code is shown in ***code snippet 4***.

Code Snippet 4:

```
def text_classifier_sentiment(corpus):
    """
    The function takes the classifier from the flair package  which can be
either
    transformer-based or RNN-based method which could sense the sentiment in
reviews
    using the ML methods of word embedding.
    : param corpus:
    :return: classification_val: returns the values of classification.
    """

    # tagger is based on the transformer which has high accuracy
    tagger = Classifier.load('sentiment-fast')
    class_val = []
    class_score = []
    for ind in tqdm(range(0, len(corpus))):
        sentence = corpus[ind]
        sentence_test = Sentence(sentence)
        tagger.predict(sentence_test)
        if str(sentence_test.labels[0].value) == "POSITIVE":
            classification_val = "Positive"
            classification_score = sentence_test.score
        # else str(sentence_test.labels[0].value) == "NEGATIVE":
        else:
            classification_val = "Negative"
            classification_score = sentence_test.score
        # Appending the value to the list
        class_val.append(classification_val)
        class_score.append(classification_score)

    return class_val, class_score
```

Ensemble-based methods for classification

There is a class imbalance noted in the model and tree-based methods which don't make any assumptions and are non-parametric can handle the imbalance efficiently. Xgboost is based on the greedy sequentially learnt Regularised Gradient Boosting Model with added parameters in the second order$\Omega(ft)$ equation to make the algorithm quickly learn patterns in larger datasets and can handle distributed machine learning settings (Chen & Guestrin, 2016). The XGBoost applied here is based on the grid search on the classification problem. A Random Forest model is used, which is not included part of the discussion. The hyper-parameters and grid search are shown in ***Table 5.*** The code snippet for XGBoost is Shown in ***Code snippet 5***.

Table 5. Hyper-parameter and Grid Search Settings XGBoost

Model and Hyper Parameters-values
XGBoost Model
Learning_rate: The step size of the shrinkage, stops the model from over-fitting, lower values are better. `[0.1]` **gamma**: minimum loss reduction at the split, a higher value can make the model overfit. `(1)` **max_depth**: The depth of trees at the split, higher values can make the model overfit. `(3-6)` **min_child_weight:** The minimum hessian weights in a child before the partition is given up.`(1, 3, 5)` **subsample:** The weight of the sample considered at each iteration. `(0.6, 0.7)` **colsample_bytree:** subsample considered at each tree. `(0.8)` **"objective function" = "binary: logistic"** The problem is a dichotomous logistic problem. "**scale_pos_weights**": There is a class imbalance an inbuilt class imbalance method handling is used. `(len(y_train.sentiment[y_train.sentiment == 'Negative'])/len(y_train.sentiment[y_train.sentiment == 'Positive']))**0.5` Data is label-encoded for the response variables.
Grid Search Parameters
"scoring": **"roc_auc"** for evaluation metrics. **"CV":** A 5 fold cross-validation method is used

Note. The parameters are adopted based on the XGBoost Developers page (xgboost-developers, 2023)

Code Snippet 5

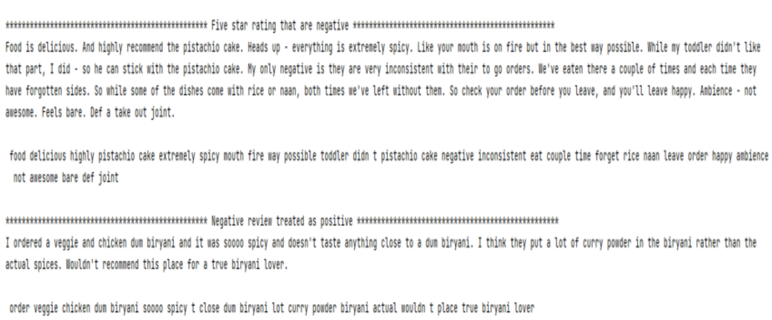
```
weight_val = (len(y_train.sentiment[y_train.sentiment ==
'Negative'])/len(y_train.sentiment[y_train.sentiment == 'Positive']))**0.5
## Grid Parameters for the XGBoost
param_grid_xgb= {
    "learning_rate": [0.01], # Step size of the shrinkage, stops the model
from over-fitting.
    "gamma": [1], # minimum loss reduction at the split.
    "max_depth": [3, 4, 5, 6], # maximum depth of the tree
    "min_child_weight": [1, 3, 5], # minimum instance hessian weight in the
tree.
    "subsample": [0.6,0.7],    # sub-sample occurs once in each boosting and
prevents over-fitting.
```

```
    "colsample_bytree": [0.8],  # sub-sample ratio for every tree
constructed.
}

#  XGBoost classifier initialised
xg_boost_cl = XGBClassifier(n_estimators=300, # number of estimator at
gradient boosting
                         objective = "binary:logistic", # The class is
dichotomous a "binary logistic model is used"
                         seed = 2391, # Setting the seed to reproduce
results.
                         nthread = 4, # number of thread assigned.
                         max_delta_step = 1,
                        scale_pos_weight = weight_val # Class weight
proportionate to class distribution.
                        )

# Grid Search model initiation to find the optimal model to be used to build
better model.
grid_search_clf = GridSearchCV(estimator=xg_boost_cl,
                               n_jobs = -1,                  # No parallel
processing is used
                               param_grid = param_grid_xgb,  # Parameters in
the grid search
                               scoring = 'roc_auc',          #  Area under
the curve for evaluation metric
                               cv = 5,                       # cross
validation parameter.
                               verbose = 1)
```

Results and Discussion on Classifier

The classification done using the sentiment is not an accurate representation of the positive or negative classes. The sentiment analyser was able to judge the semantic ambiguity in a sentence and was able to judge a 5-Star rated negative review and on the other hand, failed to account for the 1-Star negative review and accounted it as "positive" as shown in ***Figure 13.*** This could lead to machine-induced bias in the models. The failure in the second case can be attributed to lemmatised words not having any negative sentiment or complete negative words.

Figure 13. Sentiment Analyser and Ambigiuty



```
************************************************ Five star rating that are negative ************************************************
Food is delicious. And highly recommend the pistachio cake. Heads up - everything is extremely spicy. Like your mouth is on fire but in the best way possible. While my toddler didn't like
that part, I did - so he can stick with the pistachio cake. My only negative is they are very inconsistent with their to go orders. We've eaten there a couple of times and each time they
have forgotten sides. So while some of the dishes come with rice or naan, both times we've left without them. So check your order before you leave, and you'll leave happy. Ambience - not
awesome. Feels bare. Def a take out joint.

 food delicious highly pistachio cake extremely spicy mouth fire way possible toddler didn t pistachio cake negative inconsistent eat couple time forget rice naan leave order happy ambience
  not awesome bare def joint

************************************************ Negative review treated as positive ************************************************
I ordered a veggie and chicken dum biryani and it was soooo spicy and doesn't taste anything close to a dum biryani. I think they put a lot of curry powder in the biryani rather than the
actual spices. Wouldn't recommend this place for a true biryani lover.

 order veggie chicken dum biryani soooo spicy t close dum biryani lot curry powder biryani actual wouldn t place true biryani lover
```

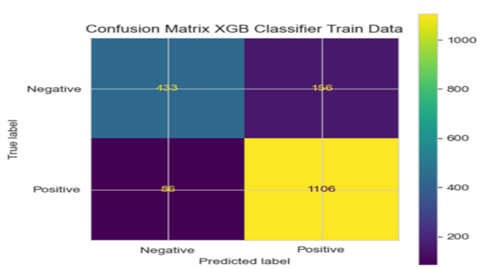
A good classifier is not too aggressive or conservative in decision-making. In the information retrieval domain, precision and recall scores determine the effectiveness of the classifier. Precision can be defined as how often the model can predict a positive class as positive and recall is the number of positive predictions that the model missed(Lantz, 2013, p. 310). There is a trade-off between precision and recall and in an effective system, the precision is traded off for recall when the number of positive classes increases (Manning et al., 2008, p. 144). In our case, both the training and test models performed on par with the precision and recall in macro-averages showing that the model can effectively differentiate positive from negative, and the ***negative class (reviews) is our priority*** in this case. An effective measure is the harmonic average of precision and recall which is the ***F1 Score*** that showed good results in train and testing as shown in ***Figure 14.*** As the problem is a classification problem, a Receiver Operating Characteristic (***ROC***) plot is plotted and the results are shown in ***Figure 14, plot 3***. A model score of AUC between 0.7 and 0.8 is fairly acceptable and in our case, it is 78% (Lantz, 2013, p. 313)

In a frequency-based term selection with normalisation, the model features to select the words that have no information and can sometimes contribute to an optimised model and these models can be robust to skewed distribution(Manning et al., 2008, p. 257). There are some outliers in the data in the form of frequent text as shown in ***Figure 9*** which might impact the model that is handled with the features range chosen in TFIDF and contribute to better model predictions.

Figure 14. Evaluation Metrics for XGBoost Train and Test, ROC Curve

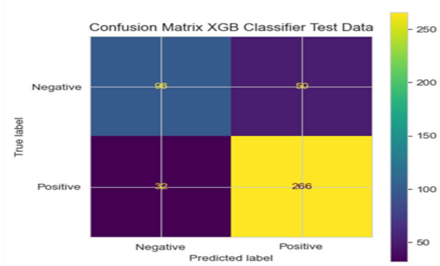
Plot 1. Evaluation Metrics XGB train

	precision	recall	f1-score	support
class 1	0.74	0.83	0.78	519
class 0	0.93	0.88	0.90	1262
accuracy			0.86	1781
macro avg	0.83	0.86	0.84	1781
weighted avg	0.87	0.86	0.87	1781



Plot 2. Evaluation Metrics XGB test

	precision	recall	f1-score	support
class 1	0.66	0.75	0.71	130
class 0	0.89	0.84	0.87	316
accuracy			0.82	446
macro avg	0.78	0.80	0.79	446
weighted avg	0.83	0.82	0.82	446



Plot 3. ROC Curve for the XGBoost Test data

Conclusion

The model building is only the tip of the iceberg, many more models can be pipelined to understand the customer moods and sentiments and what can be improved to make effective business decisions. The model can be extended to other review models and can be evolved to make better predictions.

Limitations:

- One of the notable limitations is that the "spacy" lemmatizer is extremely time-consuming at 22 mins for 2k reviews and based on the transformer model, NLTK can achieve that in 20 seconds. This is the sacrifice of accuracy to time.
- The other limitation is the limited word count of the report and that doesn't give room to talk about other models like Random Forest or how the models can be evolved.

References

Akbik, A., Bergmann, T., Blythe, D., Rasul, K., Schweter, S., & Vollgraf, R. (2019, 2019/6). *FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP* Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations), http://dx.doi.org/10.18653/v1/N19-4010

Al Sharou, K., Li, Z., & Specia, L. (2021). *Towards a Better Understanding of Noise in Natural Language Processing*. https://doi.org/10.26615/978-954-452-072-4_007

Albanese, N. C. (2022, September 2022). *Topic Modeling with LSA, pLSA, LDA, NMF, BERTopic, Top2Vec: a Comparison*. https://towardsdatascience.com/topic-modeling-with-lsa-plsa-lda-nmf-bertopic-top2vec-a-comparison-5e6ce4b1e4a5

Arevalo, M. (2017, April 26). *The Impact of Reviews on the Restaurant Market*. Modern Restaurant Mangement. https://modernrestaurantmanagement.com/the-impact-of-reviews-on-the-restaurant-market-infographic

Bird, S., Ewan, & Loper, E. (2009). *Natural language processing with Python: Analyzing text with the natural language toolkit*. O'Reilly. https://www.nltk.org/book/

Blei, D. M., Ng, A. Y., & Jordan, M. I. (2003). Latent Dirichlet Allocation. *J. Mach. Learn. Res.*, *3*(null), 993–1022 , numpages = 1030. https://dl.acm.org/doi/10.5555/944919.944937

Bouma, G. J. (2009). *Normalized (pointwise) mutual information in collocation extraction* https://svn.spraakdata.gu.se/repos/gerlof/pub/www/Docs/npmi-pfd.pdf

Chen, J., Dong, H., Wang, X., Feng, F., Wang, M., & He, X. (2020). Bias and Debias in Recommender System: A Survey and Future Directions. *ACM Transactions on Information Systems*. https://doi.org/10.48550/arXiv.2010.03240

Chen, T., & Guestrin, C. (2016). *XGBoost: A Scalable Tree Boosting System* Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, https://doi.org/10.1145/2939672.2939785

Dobrilla, A. (2022). Social Media Restaurant Statistics to Profit from. https://www.gloriafood.com/social-media-restaurant-statistics

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del Río, J. F., Wiebe, M., Peterson, P., . . . Oliphant, T. E. (2020). Array programming with NumPy *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

Helan, A., & Sultani, Z. N. (2023). Topic modeling methods for text data analysis: A review. *AIP Conference Proceedings*, *2457*(1). https://doi.org/10.1063/5.0118679

Hoffman, M. D., Blei, D. M., & Bach, F. (2010, 2010). Online Learning for Latent Dirichlet Allocation. Proceedings of the 23rd International Conference on Neural Information Processing Systems - Volume 1,

Honnibal, M., Motani, I., Landeghem, S. V., & Boyd, A. (2020). Industrial-strength Natural Language Processing in Python. https://doi.org/10.5281/zenodo.1212303

Lantz, B. (2013). *Machine Learning with R*. Packt Publishing.

Lee, M., Kwon, W., & Back, K.-J. (2021). Artificial intelligence for hospitality big data analytics: developing a prediction model of restaurant review helpfulness for customer decision-making. *International Journal of Contemporary Hospitality Management*, *33*(6), 2117-2136. https://doi.org/10.1108/IJCHM-06-2020-0587

Magajna, T. (2021, December 07). *Text Classification with State of the Art NLP Library — Flair*. https://towardsdatascience.com/text-classification-with-state-of-the-art-nlp-library-flair-b541d7add21f

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations of Words and Phrases and their Compositionality. *CoRR*, *abs/1310.4546*. http://arxiv.org/abs/1310.4546

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., & Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python *Journal of Machine Learning Research*, *12*, 2825-2830. https://scikit-learn.org/stable/about.html

Piantadosi, S. T. (2014). Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic Bulletin & Review*, *21*(5), 1112-1130. https://doi.org/10.3758/s13423-014-0585-6

restaurantji. (2023). *Restaurant Review Data* [Reviews Dataset]. https://www.restaurantji.com/

Řehůřek, R., & Sojka, P. (2010, May 22). *Software Framework for Topic Modelling with Large Corpora* Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks, http://is.muni.cz/publication/884893/en

Rungta, k. (2023, April 15). *What is Selenium? Introduction to Selenium Automation Testing*. https://www.guru99.com/introduction-to-selenium.html

similarweb. (2023, April 28). *Similarweb*. https://www.similarweb.com/website/restaurantji.com/#overview

Simkus, J., & Mcleod, S. (2023, March 07). *Stratified Random Sampling: Definition, Method & Examples*. https://www.simplypsychology.org/stratified-random-sampling.html

Sievert, C., & Shirley, K. (2014, June 01). *LDAvis: A method for visualizing and interpreting topics* Proceedings of the Workshop on Interactive Language Learning, Visualization, and Interfaces, http://dx.doi.org/10.3115/v1/W14-3110

Stevens, K., Kegelmeyer, P., Andrzejewski, D., & Buttler, D. (2012, 2012/7). *Exploring Topic Coherence over Many Models and Many Topics* Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning, https://aclanthology.org/D12-1087

Team The Pandas Development. (2020). *pandas-dev/pandas: Pandas* In (Version 1.5.3) [Python package]. Zenodo. https://doi.org/10.5281/zenodo.3509134

Van Rossum, G., & Drake, F. L. (2009). *Python 3* In (Version 3.9.16) [Python Software]. CreateSpace. https://www.python.org/downloads/release/python-390/

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., . . . SciPy, C. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python *Nature Methods*, *17*, 261-272. https://doi.org/10.1038/s41592-019-0686-2

Waskom, M., Botvinnik, O., O'Kane, D., Hobson, P., Lukauskas, S., Gemperline, D. C., Augspurger, T., Halchenko, Y., Cole, J. B., Warmenhoven, J., de Ruiter, J., Pye, C., Hoyer, S., Vanderplas, J., Villalba, S., Kunter, G., Quintero, E., Bachant, P., Martin, M., . . . Qalieh, A. (2017). *mwaskom/seaborn: v0.8.1 (September 2017)*. In (Version v0.8.1) Zenodo. http://dx.doi.org/10.5281/zenodo.883859

Whittaker, Z. (2022, April 18). *Web scraping is legal, US appeals court reaffirms*. https://techcrunch.com/2022/04/18/web-scraping-legal-court

xgboost-developers. (2023, March 01). *XGBoost Parameters*. xgboost developers. https://xgboost.readthedocs.io/en/latest/parameter.html

ZenRows. (2023). Selenium vs BeautifulSoup in 2023: Which Is Better? *ZenRows*. https://www.zenrows.com/blog/selenium-vs-beautifulsoup#beautifulsoup

Zhao, R., & Tan, V. Y. F. (2017). Online Nonnegative Matrix Factorization With Outliers. *IEEE Transactions on Signal Processing*, *65*(3), 555-570. https://doi.org/10.1109/tsp.2016.2620967

Zibarzani, M., Abumalloh, R. A., Nilashi, M., Samad, S., Alghamdi, O. A., Nayer, F. K., Ismail, M. Y., Mohd, S., & Mohammed Akib, N. A. (2022). Customer satisfaction with Restaurants Service Quality during COVID-19 outbreak: A two-stage methodology. *Technology in Society*, *70*, 101977. https://doi.org/https://doi.org/10.1016/j.techsoc.2022.101977

Appendix A

Figure 1. Topic Modelling visualisation using pyLDAvis

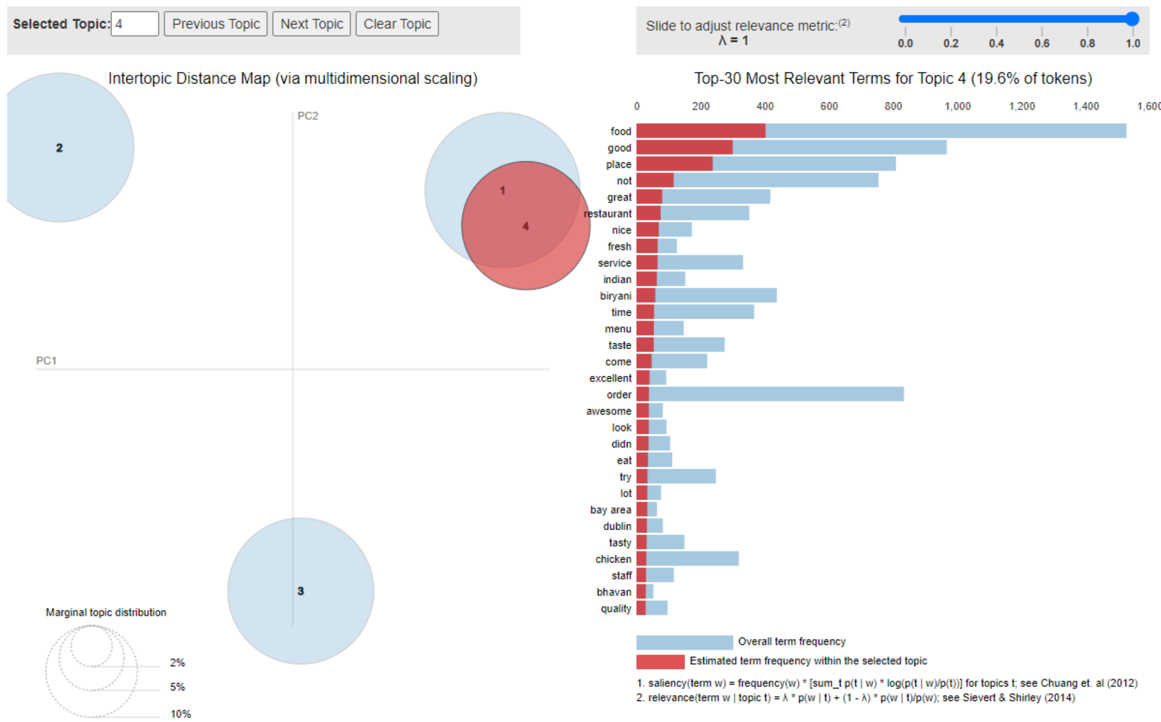


Figure 2. A negative review and its prediction which is ambiguous and result

```
# ------------------------A negative review----------------------------------
#---------------------------------------------------------------------------
# review_test_pred = ["The restaurant is good with ambience, the south-indian food is real bad. I will not go this place again."]
review_test_pred = ["I ordered a veggie and chicken dum biryani and it was soooo spicy and doesn't taste anything close to a dum biryani. I think they put a lot of curry powder
 in the biryani rather than the actual spices. Wouldn't recommend this place for a true biryani lover."]

# predicting the sentiment from the tweet.
pred_sentiment_val(review_test_pred,
                   modl= best_xgb_modl,
                   labl_encdr= label_encdr)
#Calling the function "pred_review_to_simmod_topic" for "LDA" model.
pred_review_to_simmod_topic(review_test= review_test_pred,
                            model=lda_model,
                            word_num=30, str_title = "LDA MODEL")
```

```
100%|██████████| 1/1 [00:00<00:00,  1.22it/s]

*************************The Sentiment of the model*************************
The sentiment of the model is : Negative

********************************************************************************

The Review is referring to the LDA MODEL topic: 2

I ordered a veggie and chicken dum biryani and it was soooo spicy and doesn't taste anything close to a dum biryani. I think they put a lot of curry powder in the biryani
 rather than the actual spices. Wouldn't recommend this place for a true biryani lover.
[('food', 0.07754946), ('order', 0.03617695), ('good', 0.03368118), ('place', 0.028049212), ('great', 0.023688808), ('not', 0.022811692), ('service', 0.022024399),
('restaurant', 0.017095668), ('time', 0.01563008), ('amazing', 0.011876144), ('delicious', 0.011789287), ('try', 0.010493273), ('come', 0.007998536), ('love', 0
.0077627026), ('south indian', 0.007683102), ('indian food', 0.0075999293), ('serve', 0.006949047), ('taste', 0.0067657284), ('menu', 0.00666901), ('dosa', 0.006352047),
('bad', 0.006095072), ('tasty', 0.0060085496), ('staff', 0.0059468835), ('nice', 0.004790203), ('table', 0.004626283), ('ask', 0.0045965677), ('definitely', 0.004316991),
 ('enjoy', 0.0042376295), ('thali', 0.0041889274), ('little', 0.004132783)]
```