Sentiment Analysis of Supermarket Reviews: Insight for Business Strategy

Table of Contents

Table of Figures

1.1 Introduction

Customer opinion is one of the most important factors that can enhance the quality of service (Bhatia, Chaudhary, & Dey, 2020). With online review platforms like Trustpilot, customers can easily voice their opinions about businesses. Smart businesses can leverage these platforms to gain insights into consumer preferences and address customer critiques quickly and effectively to foster stronger relationships and loyalty.

Sentiment analysis is one of the most impactful applications of text mining, used to determine the opinion or emotion expressed in a text (Guia, Silva, & Bernardino, 2019). Several machine learning algorithms have been applied to sentiment analysis each with unique strengths. The Naive Bayes algorithm is one of the most common with its simplicity and efficiency in classifying texts (Kristiyanti et al., 2020). Applications of these techniques are crucial for businesses aiming to understand customer perspectives and improve service.

This study uses sentiment analysis to evaluate customer feedback from five major supermarket chains in the United Kingdom and derive insights to improve service quality and customer satisfaction.

1.1.1 Research Question

How effective are text classification and sentiment analysis in classifying reviews, and will this lead to actionable business insights?

1.1.2 Relevant works

Sentiment analysis has proven to be effective in understanding customer reviews. Lim et al. (2019) analysed customer reviews for the top ten retailers in the U.S. and demonstrated the importance of sentiment analysis in identifying customer preferences. Latif et al. (2020) explored sentiment analysis for evaluating products through customer reviews. Gan et al. (2016) adopted the use for analysing online restaurant reviews. These studies emphasize the critical role of sentiment analysis in enhancing business performance by understanding customers.

1.1.3 Methodology

This study adopts the CRISP-DM (Cross Industry Standard Process for Data Mining) methodology, which provides a structured and systematic approach to handling data mining tasks. This process is divided into five phases: Business Understanding, Data Understanding, Data Preparation, Modelling and Evaluation (Larose & Larose, 2015).

1.2 Dataset

The dataset used was scraped from Trustpilot, a popular online review platform using code from an open-source Google Colab notebook on GitHub (Analytics with Adam, 2024). It contains 5,000 reviews from five of the UK's biggest supermarket chains: Sainsbury's, Tesco, Lidl, Aldi and Asda.

*Table **Error! No text of specified style in document.**.1 Description of data content*

FEATURE	DESCRIPTION
COMPANY	Name of supermarket (Sainsbury's, Tesco, Lidl, Aldi and Asda)
USERNAME	Unique identifier of the reviewer
LOCATION	Location the review is from
REVIEW	Text describing experience with the supermarket
RATING	Score given form 1-5 (1 being the lowest and 5 the highest)

```
# Loading dataset
company_reviews = pd.read_csv('company_reviews.csv',index_col=0)
```

```
# Anonymizing the data
company_reviews = company_reviews.drop(['Username','Location'],axis=1)
```

*Figure **Error! No text of specified style in document.**.1 Anonymizing the dataset by dropping identifiers*

1.2.1 Ethical / Social /Legal Issues

Ethical: Data was anonymized by removing identifiable fields such as username and location to protect customer privacy.

Social: This analysis provides insight into customer preference to aid better service.

Legal: The dataset was scraped, raising potential issues, but the dataset has been anonymized in compliance with the General Data Protection Regulation (GDPR).

1.3 Exploratory Data Analysis and Preprocessing

Before any analysis can be carried out, a proper understanding of the data is a must. The anonymized dataset now has 5,000 rows and 4 columns. The next step is to explore the data and prepare for analysis.

```
# number of rows and columns of the data
company_reviews.shape
```

```
(5000, 4)
```

*Figure **Error! No text of specified style in document.**.2 Number of rows and columns of the review's dataset*

1.3.1 Data Cleaning

This is the process of identifying and correcting issues within a dataset (Côté et al., 2024). The following processes were performed to clean the dataset.

Duplicate Handling: A duplicated observation was identified and removed to maintain data integrity.

```
# Checking for Duplicates
company_reviews.duplicated().sum()
```

```
1
```

```
# Handling duplicates
company_reviews = company_reviews.drop_duplicates()
company_reviews.shape
```

```
(4999, 4)
```

*Figure **Error! No text of specified style in document.**.3 Code to check for and remove duplicated observation*

Handling Missing Values: Missing values can raise significant issues during the modelling process if not handled properly.

```
# checkinf for missing values
company_reviews.isna().sum()
```

```
Company    0
Date       0
Review     0
Rating     0
dtype: int64
```

*Figure **Error! No text of specified style in document.**.4 Output of code showing no missing values*

Formatting Datatypes: Each feature must be in the appropriate datatype (e.g. Company formatted as a string, Rating formatted as an integer)

```
# Understanding the datatypes
company_reviews.dtypes
```

```
Company    object
Date       object
Review     object
Rating      int64
dtype: object
```

*Figure **Error! No text of specified style in document.**.5 Output showing the data types of each feature*

1.3.2 Exploratory Data Analysis

1.3.2.1 Univariate Analysis

This is a statistical method that describes a single variable in the dataset. Each significant variable was visualized to gain some insight into it.

Company: This variable consists of 5 major supermarket chains in the United Kingdom. Each supermarket has 1000 observations in the dataset as shown in the bar chart.

```
# Count of Companies
plt.figure(figsize=(3,2))
sns.catplot(data=company_reviews, y='Company', kind='count',palette='Set2')
plt.title('Company distribution')
plt.show()
```

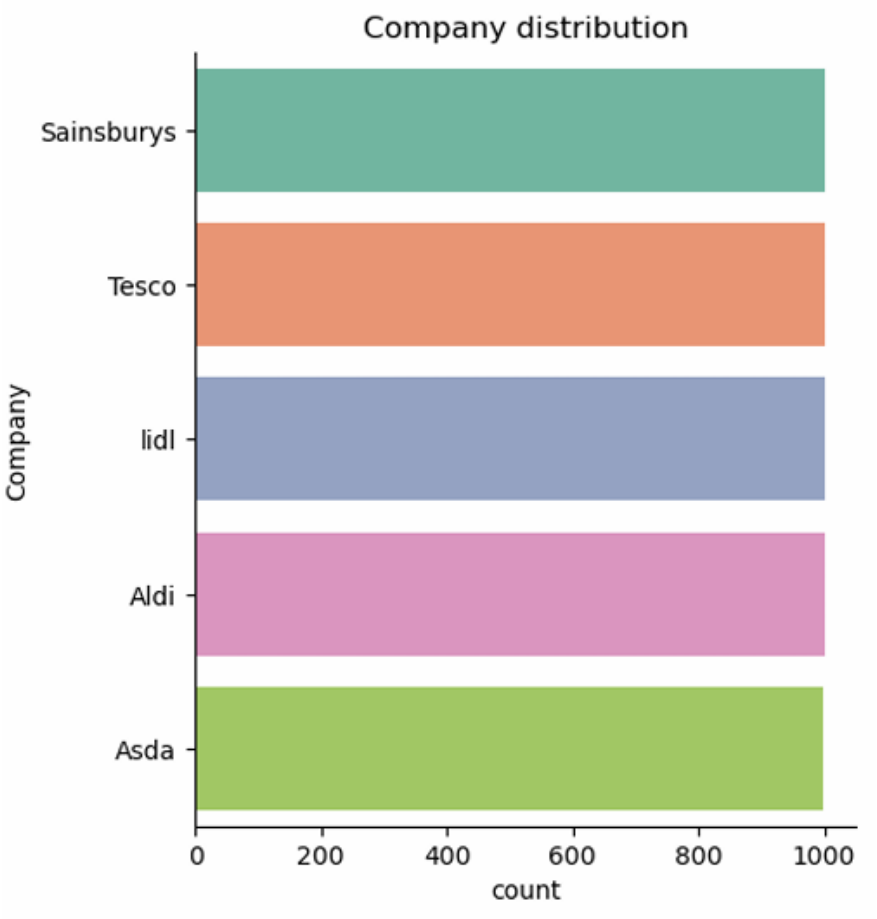

*Figure **Error! No text of specified style in document.**.6 Bar chart showing the count of reviews per company*

Review: This is the feedback given by the customer. The review consists of letters, numbers and punctuation.

```
company_reviews['Review'].iloc[3]
```

```
'great quality food at great prices with…great quality food at great prices with a
wide choice always available. We also think you have outdone every other supermark
et with the BFG Christmas advert, remembering those who may not have such a good C
hristmas.  Well done Sainsburys - you have absolutely smashed it !Date of experie
nce: November 14, 2024'
```

*Figure **Error! No text of specified style in document.**.7 An example of a customer review*

Rating: Figure **Error! No text of specified style in document.**.9 shows the feedback scores on a scale of 1 to 5, with a rating of **1** being the most frequent. This indicates that customers mostly give negative feedback, suggesting dissatisfaction.

```
# Count of ratings
plt.figure(figsize=(3,2))
sns.catplot(data=company_reviews, x='Rating', kind='count',color='lightblue')
plt.title('Ratings distribution')
plt.show()
```

*Figure **Error! No text of specified style in document.**.8 Code to plot the distribution of ratings*

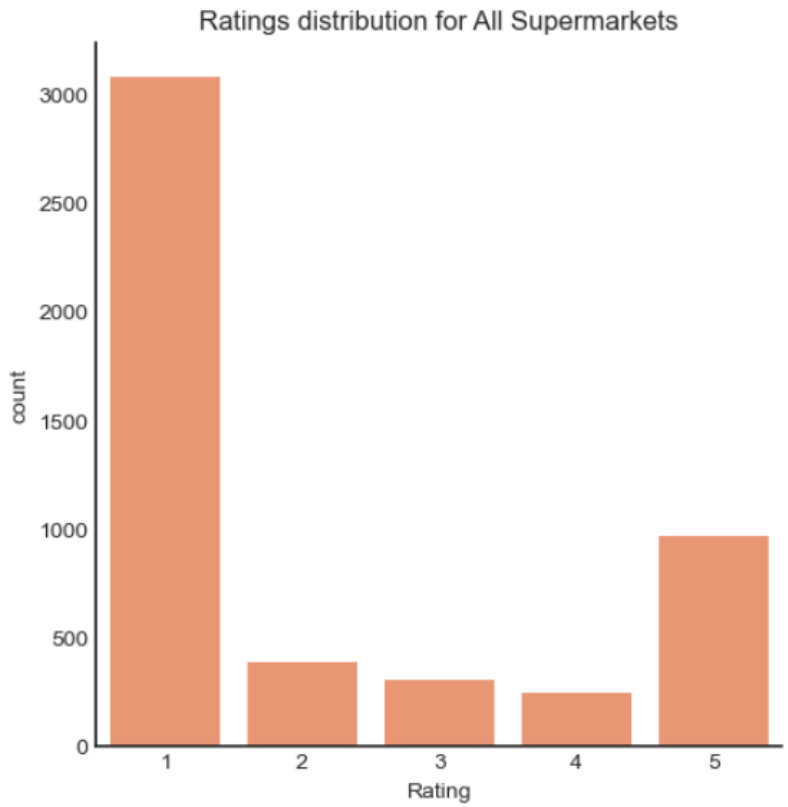

*Figure **Error! No text of specified style in document.**.9 Column chart showing the distribution of overall ratings*

1.3.2.2 Bivariate Analysis

This is the analysis of two variables, bar plots of each company's ratings were plotted. The result shows that Sainsbury's has the highest amount of 5's while Asda has the highest amount of 1's. While Tesco, Aldi and Lidl have a similar distribution of ratings among the five rating scores.

```
# Rating of each Company
sns.catplot(data=company_reviews, x='Rating', kind='count',col='Company',
            col_wrap=2,palette='Set2', hue='Company')
plt.suptitle('Company Ratings')
plt.tight_layout()
plt.savefig('company ratings.png')
plt.show()
```

*Figure **Error! No text of specified style in document.**.10 Code to plot the distribution of ratings per company*

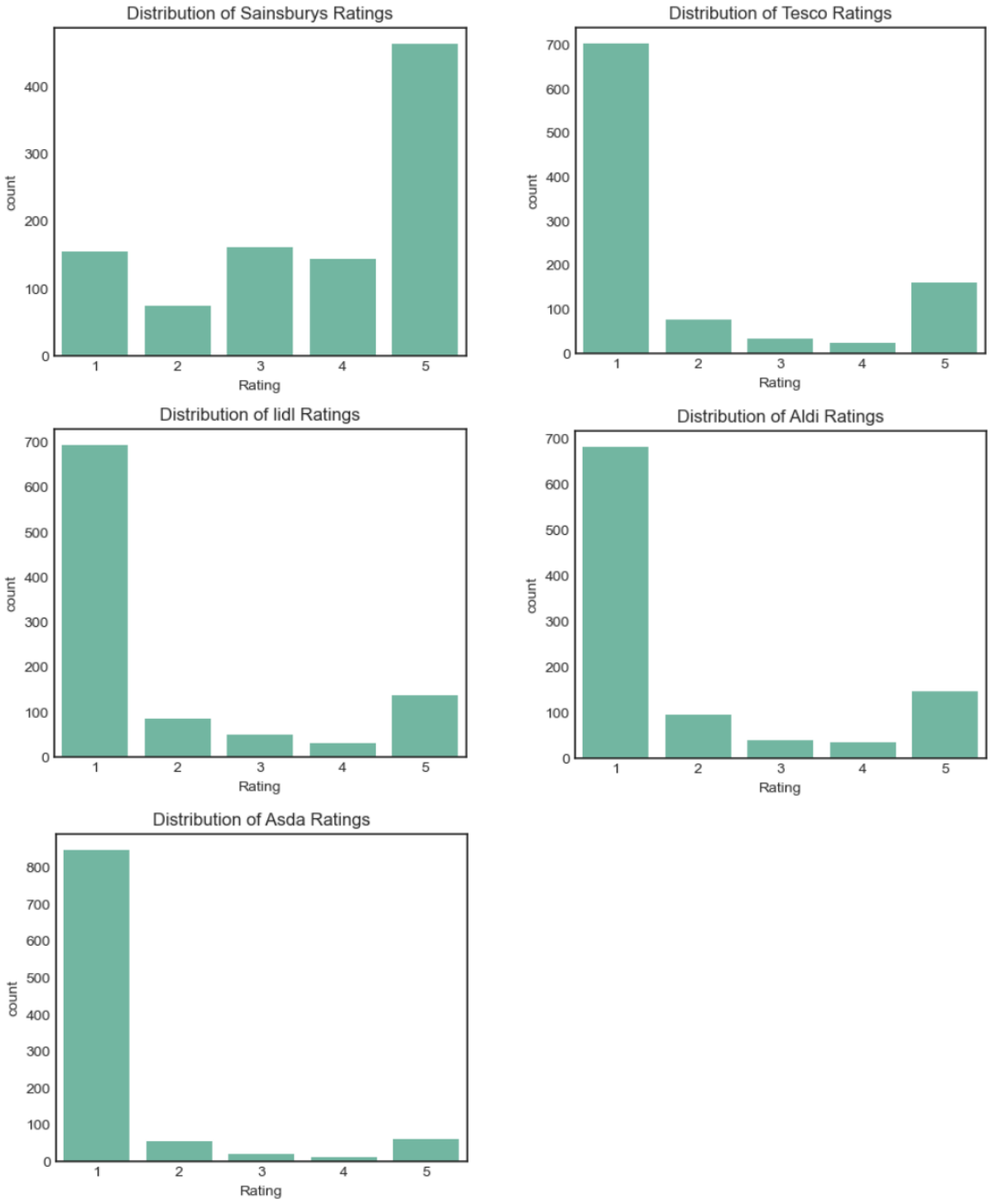

*Figure **Error! No text of specified style in document.**.11 Column charts showing the ratings of each company*

1.3.3 Preprocessing

To enhance the quality of the data, and prepare data for the modelling stage, the following data preprocessing steps are carried out:

- Tokenization: Splitting the reviews into smaller words because algorithms cannot directly process texts.
- Formatting words: Converting all words to lowercase, to have uniformity.
- Remove stop-words: Eliminating commonly used words like “the”, and “is”, that do not add meaningful context.
- Stemming: Taking the words back to their roots (e.g. “shopping” to “shop”), using Porter Stemmer.

A function was created to carry out all these preprocessing tasks at once and apply them to the Review column.

```
# stop waords
stop_words = nltk.corpus.stopwords.words('english')
```

```
# function to preprocess text
def preprocess_text(text):
    tokenized_document = nltk.tokenize.RegexpTokenizer('[a-zA-Z0-9\']+').tokenize(text)
    cleaned_tokens = [word.lower() for word in tokenized_document if word.lower() not in stop_words]
    stemmed_text = [nltk.stem.PorterStemmer().stem(word) for word in cleaned_tokens]
    return ' '.join(stemmed_text)
```

*Figure **Error! No text of specified style in document.**..12 Function to preprocess the text*

```
# creating a copy of the data
df = company_reviews.copy()
```

```
# Applying the test preprocessing function
df['Review'] = df['Review'].apply(preprocess_text)
df.head()
```

	Company	Date	Review	Rating
0	Sainsburys	17 hours ago	first rate buffet food friend suggest order bu...	5
1	Sainsburys	2 days agoInvited	sainsburi shop experiencesi'v alway shop sains...	5
2	Sainsburys	2 days agoInvited	shop onlin josetteveri conveni shop onlin conv...	5
3	Sainsburys	3 days agoInvited	great qualiti food great price great qualiti f...	5
4	Sainsburys	2 days ago	great way shop easi manag get everyth want nee...	5

*Figure **Error! No text of specified style in document.**..13 Preprocessing the text using the preprocess_text function*

Binning Ratings: Figure **Error! No text of specified style in document.**..9 reveals that the rating column containing 5 class labels is highly imbalanced. To address this, binning was applied to group the ratings into broader categories. A new feature, Rating_category was created. Table **Error! No text of specified style in document.**..2 describes the new category.

*Table **Error! No text of specified style in document.**.2 Describing the categories in Rating_category*

CATEGORY LABEL	DESCRIPTION
BAD	Ratings of 1-2
GOOD	Ratings of 3-4
EXCELLENT	Ratings of 5

```
# creating a rating category
rating_map = {
    1: 'Bad',
    2: 'Bad',
    3: 'Good',
    4: 'Good',
    5: 'Excellent'
}
```

```
df['Rating_category'] = df['Rating'].replace(rating_map)
```

*Figure **Error! No text of specified style in document.**.14 Code to bin the ratings category*

Assignment: The processed review feature(independent) and the Rating_category target (dependent) were assigned to X, and y respectively.

```
# Target and feature for text classification
X = df['Review']
y = df['Rating_category']
```

*Figure **Error! No text of specified style in document.**.15 Assigning the target and features*

Train-Test Split: The data was split into 80% training and 20% testing set. This makes sure that the model learns on the train and can be evaluated on the test. Stratification was applied to maintain the proportion of the categories in the split.

```
# Splitting to train and test
X_train, X_test, y_train, y_test = train_test_split(
    X,y,train_size=0.8,test_size=0.2, random_state=random, stratify=y
)
print(X_train.shape)
print(X_test.shape)
```

```
(3999,)
(1000,)
```

*Figure **Error! No text of specified style in document.**.16 Code showing the train-test split*

Text Vectorization: The reviews in the X variable were transformed into a numerical format using CountVectorizer so that machine learning models can process them. The result contained 12,288 unique tokens.

```
# instantiating the vectorizer
vectorizer = CountVectorizer()
# fitting and transforming
X_train = vectorizer.fit_transform(X_train)
X_test = vectorizer.transform(X_test)
```

```
print(X_train.shape)
print(X_test.shape)
```

```
(3999, 12288)
(1000, 12288)
```

*Figure **Error! No text of specified style in document.**.17 Code to vectorize the text*

1.4 Implementation

There are two tasks to be carried out in this modelling section; text classification and sentiment analysis, each with different algorithms.

1.4.1 Text Classification - Multinomial Naive Bayes Classifier

Multinomial Naive Bayes Classifier: Scikit-learn Developers (2024) highlight that Multinomial Naive Bayes is suitable for text classification because it effectively handles discrete features like word counts or term frequencies. To address the imbalanced nature of the data set, two models will be developed and compared:

- Model with imbalanced data.
- Model balanced using SMOTE.

1.4.1.1 Imbalanced Data

```
# Pie chart to visualize target.
plt.figure(figsize=(6, 5))
y_train.value_counts(normalize=True).plot.pie(
    autopct='%1.0f%%',
    startangle=100,
    colors=['lightblue', 'blue', 'darkblue'],
    labels=['Bad', 'Excellent', 'Good'],
    explode=[0.03, 0.03,0.03])
plt.title('Distribution of Ratings (Imbalanced)')
plt.ylabel('')
plt.show()
```

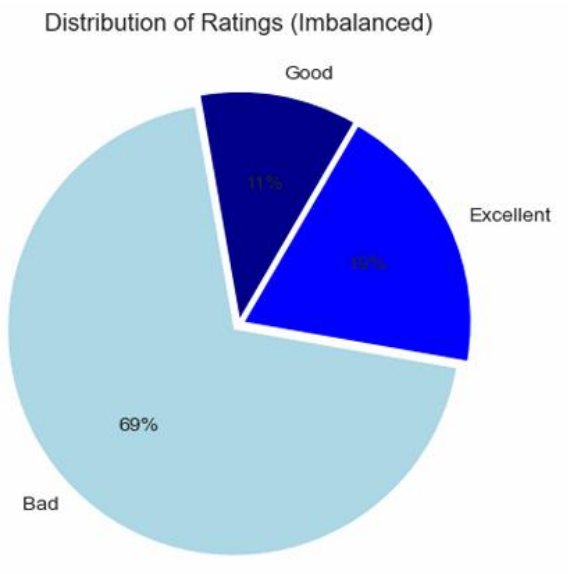

*Figure **Error! No text of specified style in document.**..18 Distribution of the imbalanced Ratings_category*

The distribution shows that the bad class is significantly larger than the other two classes. The following are steps needed to carry out text classification using Multinomial Naive Bayes:

Preprocessing

- Assigning features and target
  - Features (X): Text reviews.
  - Target (y): Rating category.
- Splitting data into train and test split (80% - 20%).
- Vectorizing the reviews (X train and X test) using ConuntVectorizer.

Modelling

- Instantiating the Multinomial Naive Bayes model from sklearn.
- Training the model by fitting the vectorized train data (X_train) and the target category train data (y_train).
- Using the trained model to make predictions on the X_test which is unseen by the model.

```
# Instantiating and fitting  naive bayes model
mnb = MultinomialNB()
mnb.fit(X_train,y_train)
```

```
MultinomialNB
MultinomialNB()
```

*Figure **Error! No text of specified style in document.**..19 code used in fitting the MultinomialNB model*

```
# predictions
y_pred = mnb.predict(X_test)
```

*Figure **Error! No text of specified style in document.**..20 Predicting the X_test*

Predictions: Model predictions can be compared with the actual values by using a confusion matrix.

```
# confusion matrix
confusion_matrix_i = confusion_matrix(y_test,y_pred)
plt.figure(figsize=(5, 3))
sns.heatmap(confusion_matrix_i, annot=True, fmt='d', cmap='Blues',cbar=False,
            xticklabels=['Bad', 'Excellent', 'Good'],
            yticklabels=['Bad', 'Excellent', 'Good'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for reviews classification')
```

*Figure **Error! No text of specified style in document.**..21 Code for plotting the confusion matrix*

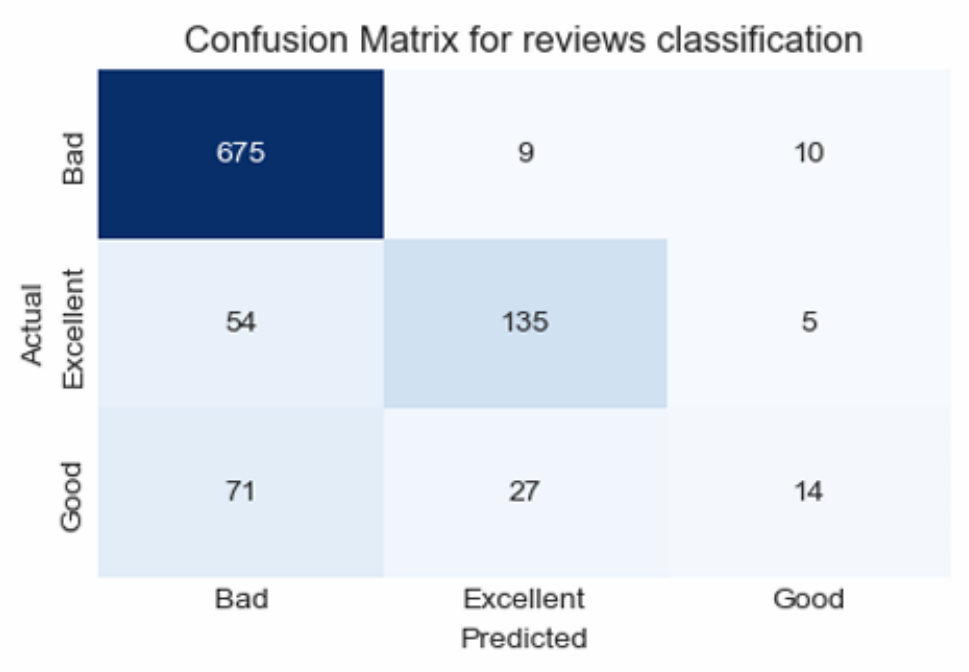

*Figure **Error! No text of specified style in document.**..22 Confusion matrix showing the model predictions for each class in the imbalanced data*

Figure **Error! No text of specified style in document.**..22 confusion matrix shows that the model correctly predicted the following in each class:

- Bad: 675 out of 694 observations
- Excellent: 135 out of 194 observations
- Good: 14 out of 112 observations

1.4.1.2 Balanced data with SMOTE

The Synthetic Minority Oversampling Technique (**SMOTE**) is a resampling method used to address class imbalance by generating synthetic examples for the minority class (Microsoft, n.d.). Smote ensures the model receives balanced training data, thereby increasing model performance.

```
# instantiating the oversampler
smote = SMOTE(random_state=random)
```

```
# Resampling the model
X_train_smote, y_train_smote = smote.fit_resample(X_train,y_train)
```

*Figure **Error! No text of specified style in document.**..23 Balancing the model by resampling using SMOTE*

```
# Pie chart to visualize target.
plt.figure(figsize=(6, 5))
y_train_smote.value_counts(normalize=True).plot.pie(
    autopct='%1.0f%%',
    startangle=100,
    colors=['lightblue', 'blue', 'darkblue'],
    labels=['Bad', 'Excellent', 'Good'],
    explode=[0.01, 0.01,0.01])
plt.title('Distribution of Ratings (Balnced)')
plt.ylabel('')
plt.show()
```

*Figure **Error! No text of specified style in document.**..24 Code used to plot the distribution of the balanced data*

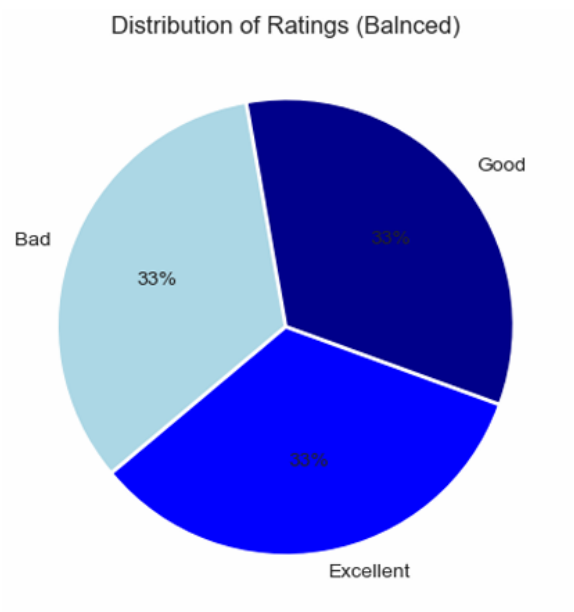

*Figure **Error! No text of specified style in document.**..25 Distribution of the balanced Ratings_category*

The same modelling processes and evaluation processes are applied to the balanced data.

Instantiate the Model and fit the oversampled train data.

```
# Instantiating and trainig the model
mnb_smote = MultinomialNB()
mnb_smote.fit(X_train_smote,y_train_smote)
```

```
MultinomialNB
MultinomialNB()
```

*Figure **Error! No text of specified style in document.**..26 code used in fitting the MultinomialNB model*

Make a Prediction from the unseen test independent feature.

```
# prediction
y_pred_smote = mnb_smote.predict(X_test)
```

*Figure **Error! No text of specified style in document.**.27 Predicting the X_test*

Confusion matrix: The model predicted correctly:

- Bad: 651 of 694 observations
- Excellent: 134 of 194 observations
- Good: 42 of 112 observations

```
# confusion matrix
confusion_matrix_b = confusion_matrix(y_test,y_pred_smote)
plt.figure(figsize=(5, 3))
sns.heatmap(confusion_matrix_b, annot=True, fmt='d', cmap='Blues',cbar=False,
            xticklabels=['Bad', 'Excellent', 'Good'],
            yticklabels=['Bad', 'Excellent', 'Good'])
plt.xlabel('Predicted')
plt.ylabel('Actual')
plt.title('Confusion Matrix for reviews classification')
```

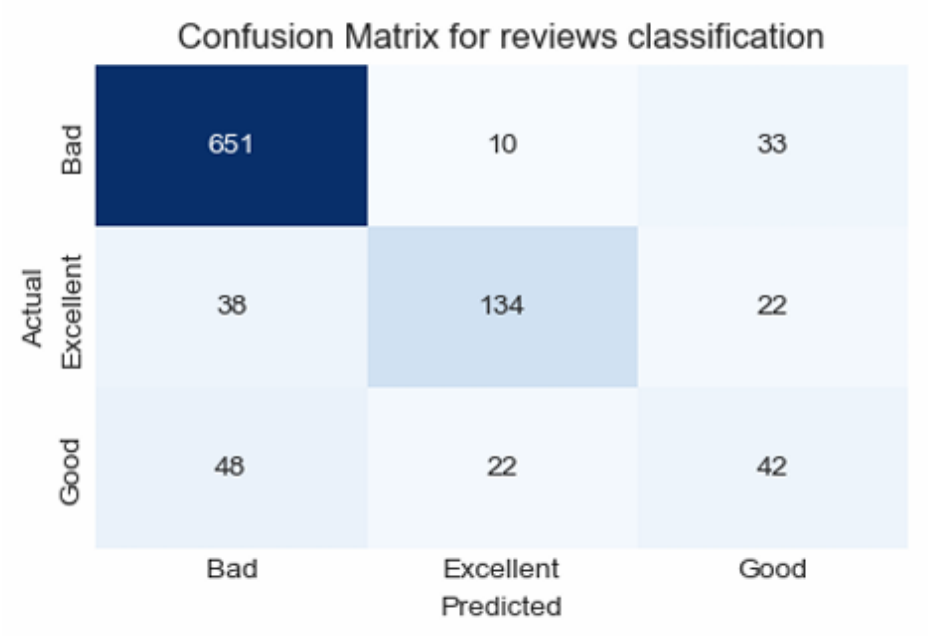

*Figure **Error! No text of specified style in document.**.28 Confusion matrix showing the model predictions for each class in the balanced data*

1.4.2 Sentiment Analysis – Sentiment Intensity Analyzer

Sentiment Intensity Analyzer: A lexicon-based sentiment analysis tool ideal for analysing sentiments in customer reviews because it accounts for context, intensity and negations (Hutto & Gilbert, 2015).

1.4.2.1 Process

Instantiating the SentimentIntensityAnalyzer: A pre-trained Valence-Aware Dictionary for Sentiment Reasoning (VADER) to estimate polarity and intensity.

```
# Instantiating the analyser
sentiment = SentimentIntensityAnalyzer()
```

*Figure **Error! No text of specified style in document.**.29 Instantiating the analyser*

Scoring Sentiments: For each review, the following scores are calculated using polarity_scores:

- Compound Score: A score ranging from -1 (most negative) to +1 (most positive)
- Positive, Neutral and Negative Scores: proportion of texts with positive, neutral and negative tone.

```
# sentiment analysis
df2['compound'] = [sentiment.polarity_scores(review)['compound'] for review in df2['Review']]
df2['neg'] = [sentiment.polarity_scores(review)['neg'] for review in df2['Review']]
df2['neu'] = [sentiment.polarity_scores(review)['neu'] for review in df2['Review']]
df2['pos'] = [sentiment.polarity_scores(review)['pos'] for review in df2['Review']]
```

*Figure **Error! No text of specified style in document.**.30 Applying the analyser to get the polarity scores*

The summary statistics of the compound polarity show a median of -0.05 implying more negative sentiments.

```
# brief summary statistics
df2[['compound','neg','neu','pos']].describe()
```

	compound	neg	neu	pos
count	4999.000000	4999.000000	4999.000000	4999.000000
mean	-0.026603	0.079984	0.820881	0.099135
std	0.702214	0.071007	0.096623	0.102116
min	-0.999200	0.000000	0.011000	0.000000
25%	-0.726900	0.017000	0.768000	0.027000
50%	-0.051600	0.072000	0.830000	0.070000
75%	0.711950	0.121500	0.886000	0.136500
max	1.000000	0.540000	1.000000	0.989000

*Figure **Error! No text of specified style in document.**.31 Summary statistics of the polarity scores*

Visualizing the polarity scores: By plotting histograms, it will become apparent how the polarity is distributed across the dataset.

```
# distribution of sentiment
sns.histplot(df2['compound'], color='lightblue')
plt.title('Distribution of Compound Polarity')
plt.show()
```

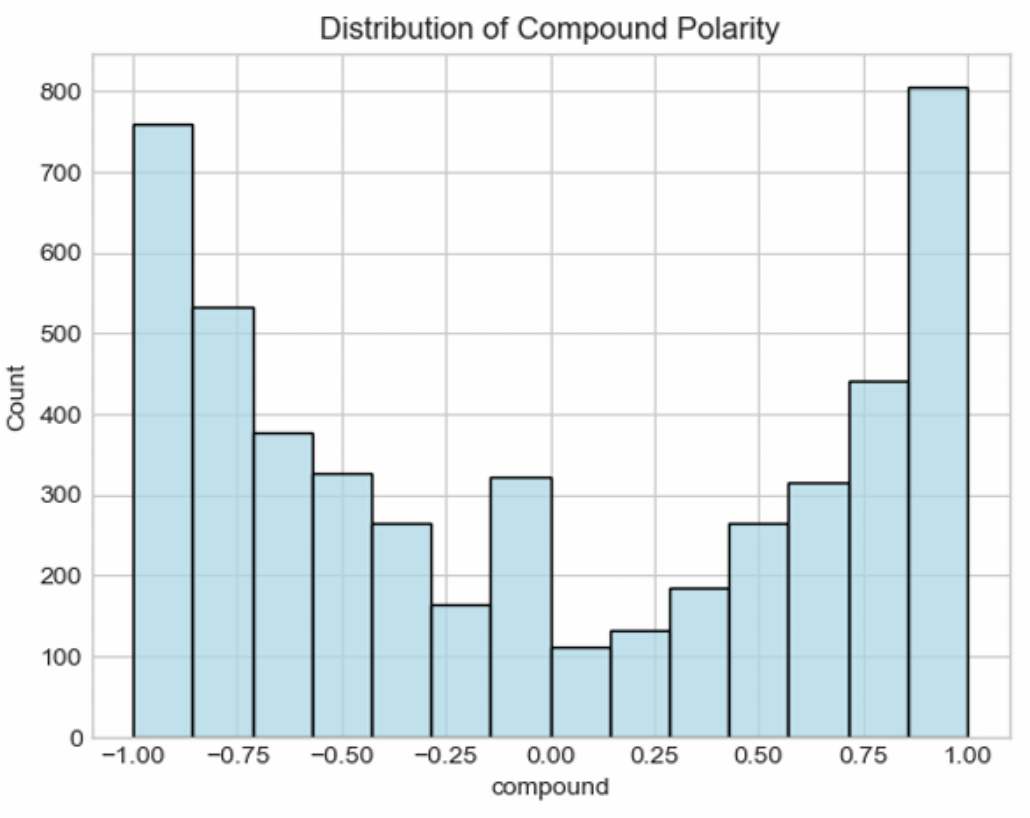

*Figure **Error! No text of specified style in document.**..32 Histogram showing the distribution of the compound polarity score*

The polarity appears to be almost evenly distributed between the positive and negative scores.

```
# distribution of positive scores
sns.histplot(df2['pos'],color='#66C2A5')
plt.title('Distribution of Positive polarity')
plt.show()
```

*Figure **Error! No text of specified style in document.**..33 code to plot the histogram of the positive polarity scores*

```
# distribution of positive scores
sns.histplot(df2['pos'],color='#FC8D62')
plt.title('Distribution of Negative polarity')
plt.show()
```

*Figure **Error! No text of specified style in document.**..34 code to plot the histogram of the negative polarity scores*

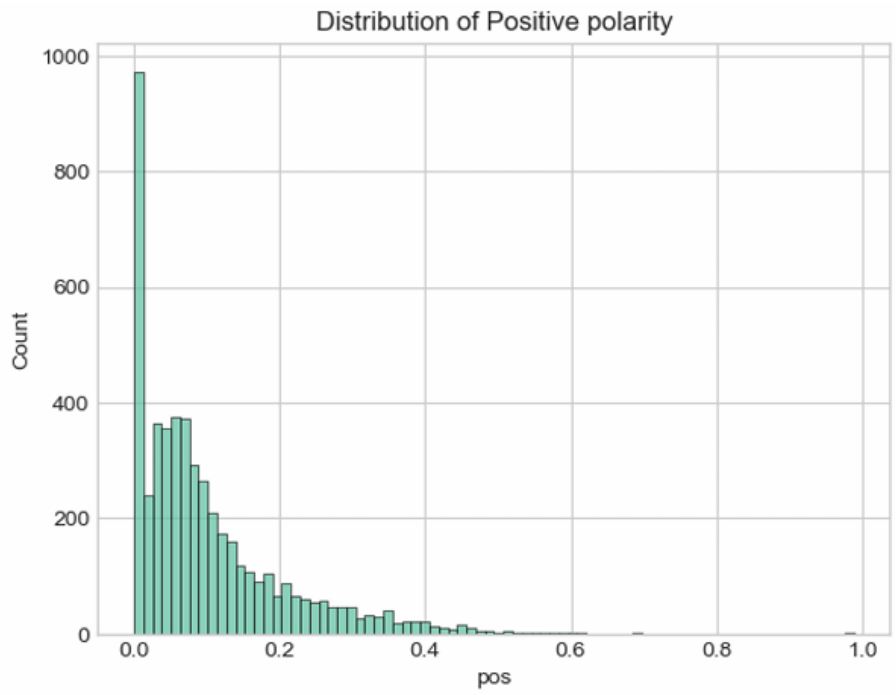

*Figure **Error! No text of specified style in document.**.35 Histogram of the positive polarity scores*

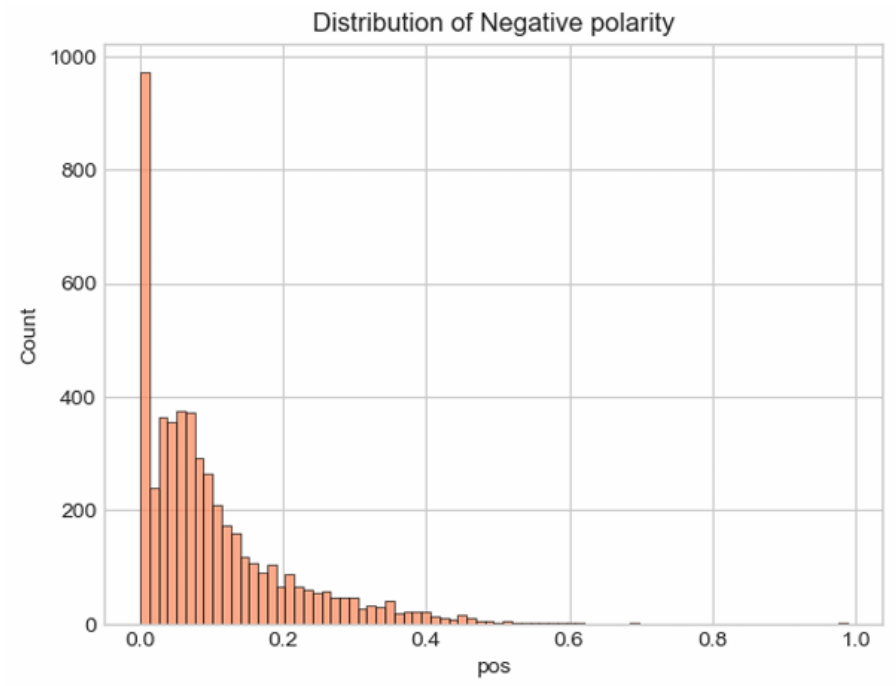

*Figure **Error! No text of specified style in document.**.36 Histogram of the negative polarity scores*

Classifying Sentiments: Sentiments were classified based on the compound score:

- Positive Sentiment: Compound score > 0.
- Negative Sentiment: Compound score ≤ 0.

```
# Classifying sentiment into positive and negative
def classify_sentiment(compound):
    if compound > 0:
        return 'Positive'
    else:
        return 'Negative'
df2['sentiment'] = df2['compound'].apply(classify_sentiment)
df2['Rating_category'] = df['Rating_category']
df2.head()
```

	Company	Date	Review	Rating	compound	neg	neu	pos	sentiment	Rating_category
0	Sainsburys	17 hours ago	First rate buffet food A friend suggested we o...	5	0.9804	0.000	0.744	0.256	Positive	Excellent
1	Sainsburys	2 days agoInvited	MY SAINSBURYS SHOPPING EXPERIENCESI've always ...	5	0.9804	0.000	0.803	0.197	Positive	Excellent
2	Sainsburys	2 days agoInvited	Shopping online is Josettevery convenient and.....	5	0.9196	0.000	0.840	0.160	Positive	Excellent
3	Sainsburys	3 days agoInvited	great quality food at great prices with...great ...	5	0.9229	0.036	0.741	0.222	Positive	Excellent
4	Sainsburys	2 days ago	Great way to shop ! Easy I manage to get every...	5	0.8852	0.114	0.676	0.210	Positive	Excellent

*Figure **Error! No text of specified style in document.**.37 Output showing the sentiments of each review in a data frame*

Visualizing the sentiment distribution: The distribution is plotted to understand the overall tone of the customers in all the supermarkets. From the distribution, the sentiment is leaning more toward the negative side of 55% negative sentiments.

```
# Sentiment distribution
plt.figure(figsize=(6, 5))
df2['sentiment'].value_counts(normalize=True).plot.pie(
    autopct='%1.0f%%',
    startangle=90,
    colors=['#FC8D62', '#66C2A5'],
    labels=['Negative', 'Positive'],
    explode=[0.01, 0.01])
plt.title('Distribution of Sentiment')
plt.ylabel('')
plt.show()
```

*Figure **Error! No text of specified style in document.**.38 Code to plot a pie chart of the sentiment distribution*

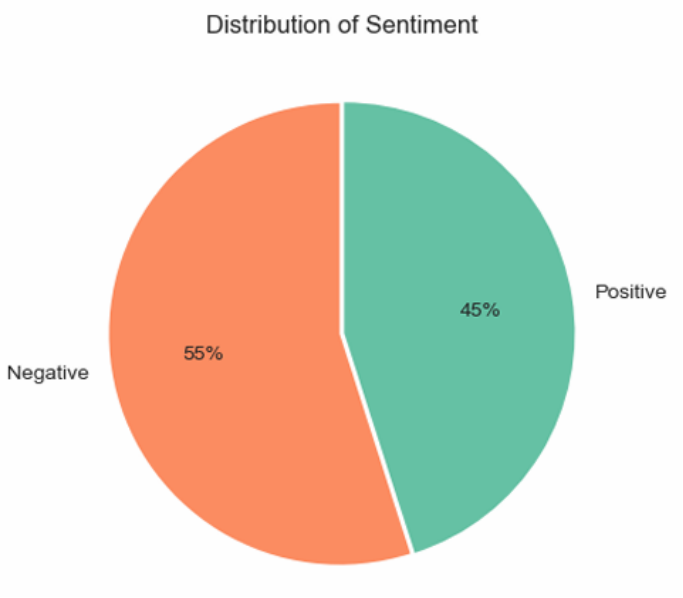

*Figure **Error! No text of specified style in document.**.39 Pie chart of the sentiment distribution*

Sentiment classification results were also broken down for each supermarket to see the variations in customer sentiment across each brand.

```
# sentiment per company
plt.figure(figsize=(3,2))
sns.catplot(data=df2, x='Company',hue='sentiment',kind='count',palette='Set2')
plt.title('Sentiment per Company')
```

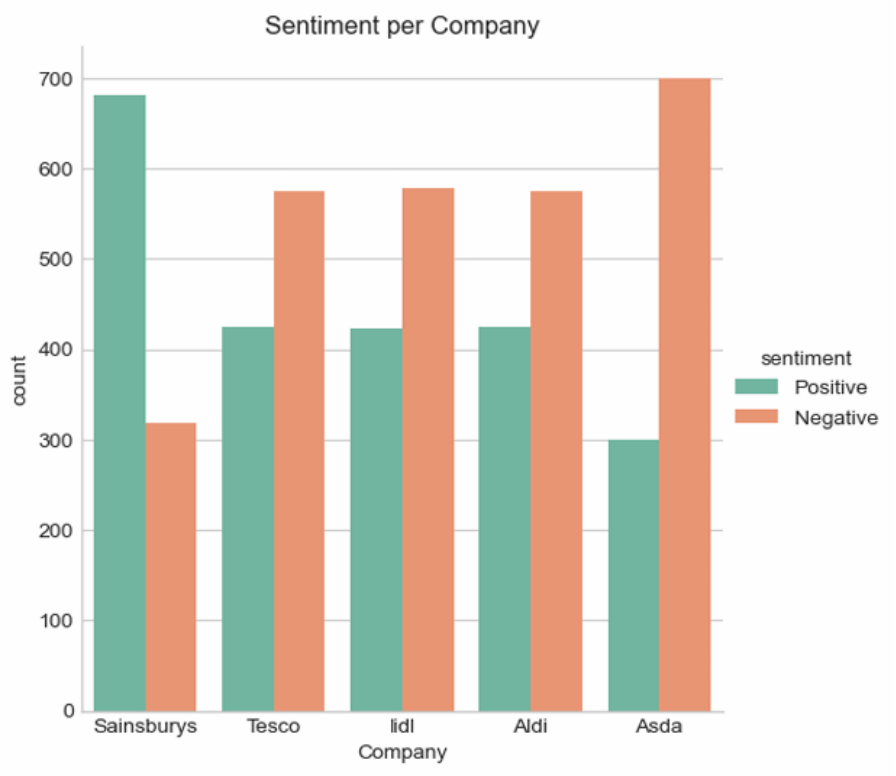

*Figure **Error! No text of specified style in document.**.40 Clustered column chart showing sentiment per company*

Figure **Error! No text of specified style in document.**.40 illustrates that Sainsbury's has the most positive sentiments and is the only supermarket with more positive sentiments than negative.

1.5 Result Analysis and Discussion

1.5.1 Text Classification Model Evaluation

The trained data predicts categories based on the unseen data (X_test) generating a list of predicted classes. This prediction is now evaluated based on accuracy, precision, recall and F1 score metrics. Due to the imbalanced nature of the dataset, recall and F1-score are better evaluation metrics for their ability to identify all actual positive observations for a class.

Classification Report: This shows how well the model predicted each class based on 3 metrics.

1.5.1.1 Imbalanced Model

```
# classification report
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

         Bad       0.84      0.97      0.90       694
   Excellent       0.79      0.70      0.74       194
        Good       0.48      0.12      0.20       112

    accuracy                           0.82      1000
   macro avg       0.71      0.60      0.61      1000
weighted avg       0.79      0.82      0.79      1000
```

*Figure **Error! No text of specified style in document.**.41 Classification report of the imbalanced MNB model*

Findings: Although the imbalanced model has an accuracy of 82%, it fails to identify each class precisely. Here are some class-specific results:

- Bad and Excellent: High precision, recall and F1-score due to dominance in the dataset.
- Good: Low Recall of only 12%, which shows the model struggles to identify this minority class.

1.5.1.2 Balanced Model with SMOTE

```
# Model Evaluation
print(classification_report(y_test,y_pred_smote))
```

```
         Bad       0.88      0.94      0.91       694
   Excellent       0.81      0.69      0.74       194
        Good       0.43      0.38      0.40       112

    accuracy                           0.83      1000
   macro avg       0.71      0.67      0.69      1000
weighted avg       0.82      0.83      0.82      1000
```

*Figure **Error! No text of specified style in document.**.42 Classification report of the balanced MNB model*

Findings: The balanced model increased slightly in accuracy; it shows significant improvement in the F1-score of the "Good" class, which increases the weighted f1-score to 82%.

Model Comparison: The SMOTE-balanced model outperforms the imbalanced model in all evaluation metrics, demonstrating the need to address class imbalance.

1.5.2 Sentiment Analysis Results

Word clouds and frequency distributions can be used to find and understand the key drivers of positive and negative sentiments for each company. Two functions were created to generate a word cloud plot and frequency distribution of the most common words in both positive and negative reviews.

Word Cloud function: This function takes in three parameters; the company name, sentiment type and colour.

```
# function to create wordcloud
def gen_wordcloud(company, sentiment, color):
    company_sentiment = df2.loc[(df2['Company']==company) & (df2['sentiment']==sentiment)]
    token = ' '.join(company_sentiment['Review'])
    wordcloud = WordCloud(background_color='white',colormap=color,
                          stopwords=new_stop_words,random_state=random).generate_from_text(token)
    #plotting
    plt.figure(figsize=(10,6))
    plt.imshow(wordcloud, interpolation='bilinear')
    plt.axis('off')
    plt.show()
```

*Figure **Error! No text of specified style in document.**.43 Function to plot word cloud*

Frequency distribution: also takes in the same three parameters.

```
# generate frequecy distribution
def gen_freq_dist(company, sentiment,color):
    company_sentiment = df2.loc[(df2['Company']==company) & (df2['sentiment']==sentiment)]
    token = ' '.join(company_sentiment['Review'])
    freq_dist = nltk.probability.FreqDist(token.split())
    freq_dist.plot(20, color=color)
    plt.title('Top 20 Word Counts')
    plt.xlabel('Word')
    plt.ylabel('Count')
    plt.show()
```

*Figure **Error! No text of specified style in document.**.44 Function to plot frequency distribution*

Analysis of Sainsbury's Reviews

- **Positive reviews:** "quality product", "good staff", "price", "helpful", "friendly", "nectar"
- **Negative reviews:** "delivery", "time", "stock", "self-checkout"

```
# Positive review Wordcloud and freqdistribution
gen_wordcloud('Sainsburys','Positive','viridis')
gen_freq_dist('Sainsburys','Positive','lightgreen')
```

*Figure **Error! No text of specified style in document.**.45 code to plot word cloud and frequency distribution for Sainsburys positive sentiment*

Figure ***Error! No text of specified style in document.****..46 Word cloud of Sainsburys positive sentiment*

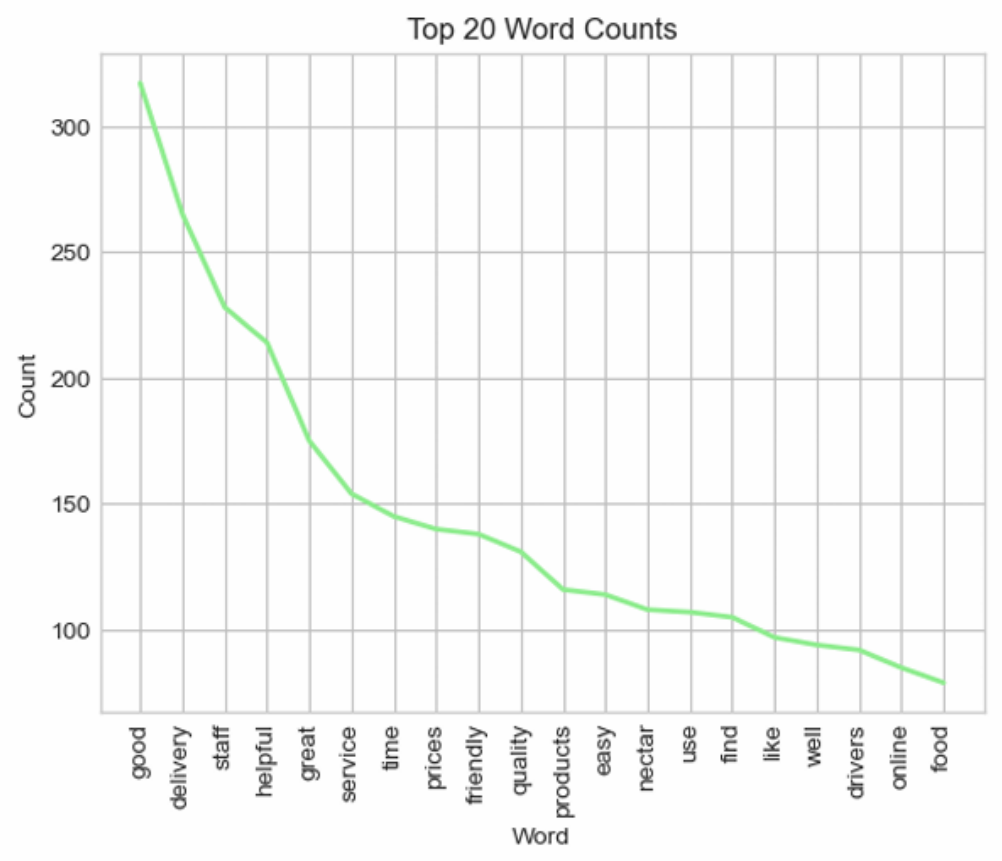

Figure ***Error! No text of specified style in document.****..47 Frequency distribution of Sainsbury's positive sentiment*

```
# Negative review Wordcloud and freqdistribution
gen_wordcloud('Sainsburys','Negative','icefire')
gen_freq_dist('Sainsburys','Negative','red')
```

Figure ***Error! No text of specified style in document.****..48 Word cloud of Sainsburys negative sentiment*

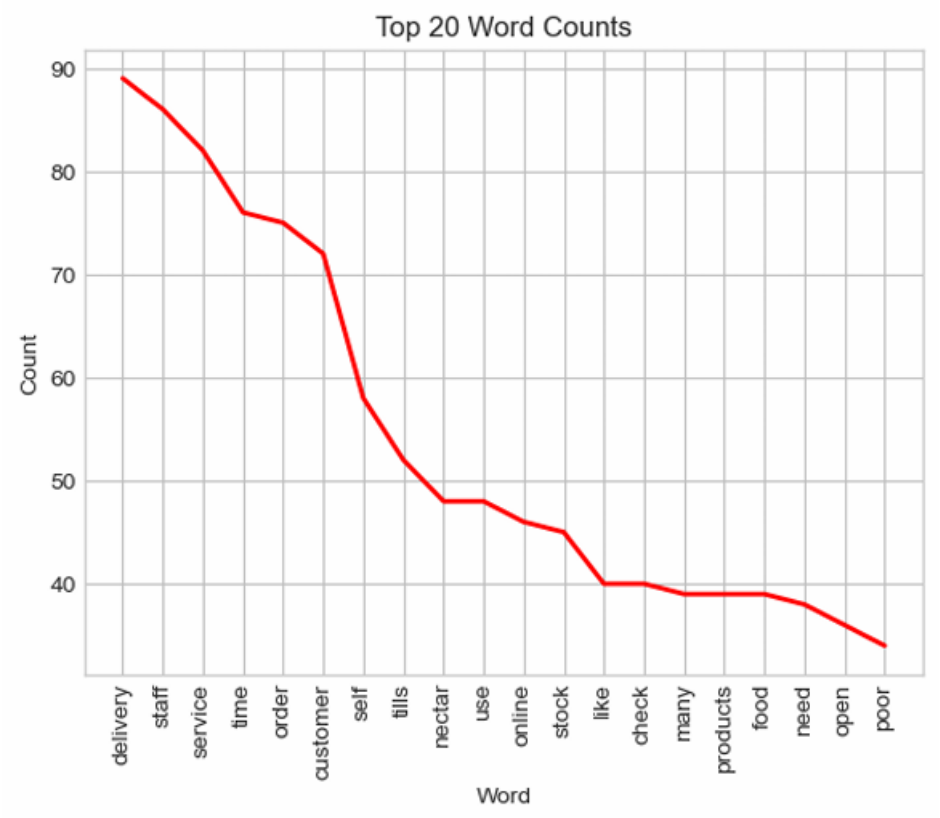

Figure ***Error! No text of specified style in document.****..49 Frequency distribution of Sainsburys negative sentiment*

Similar analyses were carried out for all the supermarkets and below are the findings.

1.5.2.1 Sentiment Distribution:

Overall Sentiment: Negative sentiments, reflecting dissatisfaction with customer service and delivery.

Company-specific Sentiment:

- **Sainsbury':** Customers praised the helpful staff, quality products, and competitive pricing. However, they have stock availability issues.
- **Tesco:** Clubcard provides good pricing, but customers reported dissatisfaction with the refund process.
- **Lidl:** Reviews often mentioned affordable pricing and good product quality, though issues with the checkout process were frequently noted.
- **Aldi:** Customers liked the quality of food and products. However, negative sentiments highlighted problems with bags, self-checkout systems, and rude staff behaviour.
- **Asda:** Awesome rewards but negative feedback focused on challenges with orders and refunds.

1.5.3 Business Implications

This study can lead to the following:

- Supermarkets can gain a better understanding of their customers through the sentiment analysis results.
- Insights gained can guide supermarkets in making data-driven decisions to improve on negative aspects and double down on the positive aspects.
- Addressing common complaints can lead to improved customer experience and foster loyalty.

1.5.4 Actionable Insights

- Address the causes of the drivers of negative sentiments such as customer service, and complicated check-out process.
- Capitalize on the positive sentiments to improve customer loyalty.
- Enhance channels for customer feedback, to identify areas needed to be improved.

1.6 Conclusion

The analysis of customer reviews from five major UK supermarkets provided valuable insights into customer sentiment and the key factors driving satisfaction or dissatisfaction. The results discovered the present state of UK supermarkets is dominated by negative sentiments about customer service and operation inefficiency.

1.6.1 Recommendation

- Supermarkets should address specific operational issues such as stock management and better check-out services.
- Customer service is the major driver of negative service; therefore, supermarkets should invest more in staff training.
- Loyalty and rewards programs (e.g., Tesco Clubcard and Asda rewards) are drivers of positive sentiment, so adopting these programs will lead to enhanced customer satisfaction
- Supermarkets should have their own feedback collection process to better identify and address customer preferences.

This project demonstrates the value of sentiment analysis in demystifying customer opinions and enabling data-driven decision-making. By addressing the drivers of negative sentiments, supermarkets can improve customer satisfaction and foster loyalty which eventually leads to business growth.