Sentiment Analysis of Hotel Reviews using SVM and XGBClassifier

- Name: Shahir Imam, Student ID: 21077367, Email: ssmimam@connect.ust.hk
- Name: Batmanaman, Student ID: 20855968, Email: Batmanaman@connect.ust.hk

Table of Contents

1. Introduction

1.1 Background

Sentiment analysis uses Natural Language Processing (NLP) to analyze text, such as online reviews, to determine the sentiment expressed in it. This is done by training a model that learns from a large set of labeled data, where each review is associated with a sentiment label (e.g., positive, negative, or neutral). This project aims to implement two machine learning models, SVM and XGBClassifier, primarily focusing on the machine learning topic covered in the course of Support Vector Machines, to automatically identify the sentiment in hotel reviews from Trip Advisor.

To illustrate the problem we have an example of a hotel guest leaving a review of the hotel stating, "The room was beautifully decorated, and the staff was helpful and friendly!" Sentiment analysis aims to automatically classify such reviews into the categories of positive, negative, or neutral. This process helps potential customers make valuable decisions based on the experiences of previous guests. Moreover, the hotel management can use this analysis to address areas that need improvement and highlight their strengths.

Once the model is trained, it can be used to analyze new hotel reviews and predict their sentiment automatically. For instance, if you input the review we mentioned earlier, the sentiment analysis model would correctly classify it as a positive sentiment. Eliminating the requirement for humans to read every review by themselves.

1.2 Problem and Purpose of Project

The objective of this project is to classify the sentiment in hotel reviews into positive, neutral, or negative categories. Using machine learning models such as Support Vector Machines (SVM) and XGBClassifier, this project aims to not only automate sentiment analysis for efficiency but also to analyze how other factors like seasonal changes can influence customer experience and feedback. The insights gained can assist the hotel management in strategic planning and operational adjustments based on customer sentiment trends found in different seasons.

The problem can be categorized as a text classification problem. The model takes the textual content of a hotel review as input. Before feeding the text into the model, preprocessing steps such as tokenization, lowercasing, and removing stop words are applied to convert the text into a suitable format. During the training process, the model will learn to recognize patterns and linguistic cues in the text data that are indicative of different sentiment categories. The model then outputs the predicted sentiment which can be represented as a class label, such as "positive," "negative," or "neutral." The model's output can also include probability scores indicating the confidence of the model's prediction for each sentiment category.

Support Vector Machines (SVM) is chosen as the primary model for this project due to its effectiveness in high-dimensional spaces, typical of text data. SVM works by finding the hyperplane that best separates different classes with the maximum margin, hence making it robust against overfitting in high-dimensional spaces. This is important for text classification tasks where the feature space can become exceedingly large.

XGBClassifier which is known for its exceptional performance and is often considered one of the most powerful algorithms for structured/tabular data. It implements advanced optimization techniques and parallel processing, making it efficient especially on large datasets. XGBoost has a reputation for delivering high accuracy in classification tasks, which provides a good comparison for the SVM.

2. Data Exploration and Preprocessing

2.1 Dataset

The project uses a free to use dataset from Kaggle, "Hotel Reviews from TripAdvisor"[1]. The dataset is from a hotel in Mussoorie, a hill station located in the north of India which accommodates guests throughout the year. It has beautiful mountains, greenery, cold breeze, and scenic beauty. The temperatures and climates change drastically along the year. The review dataset is a CSV file which has 15196 entries. The three features of the dataset include 'Review', a text of the review, 'Date of stay', the month and year of the review and 'Rating', a numerical rating of the reviewer.
Example:

Table 1: Example from the dataset.

Review	Date of Stay	Rating
Most beautiful stay at Mussoorie...such a gorgeous property. I really left my heart back there. One...	Date of stay: August 2021	
It was a memorable three nights that we spent at The Savoy, Mussoorie not only because the property ...	Date of stay: September 2021	

From the initial dataset we had to do some basic preprocessing and extract the relevant information such as the rating, which is found as the number right after 'bubble_' in the rating category.

[1] https://www.kaggle.com/datasets/ruchibhadauria/hotel-reviews-from-tripadvisor

2.2 Analyzing the Data

After removing all the duplicates and null values, there are 12796 entries remaining. We further analyze the dataset and its features.

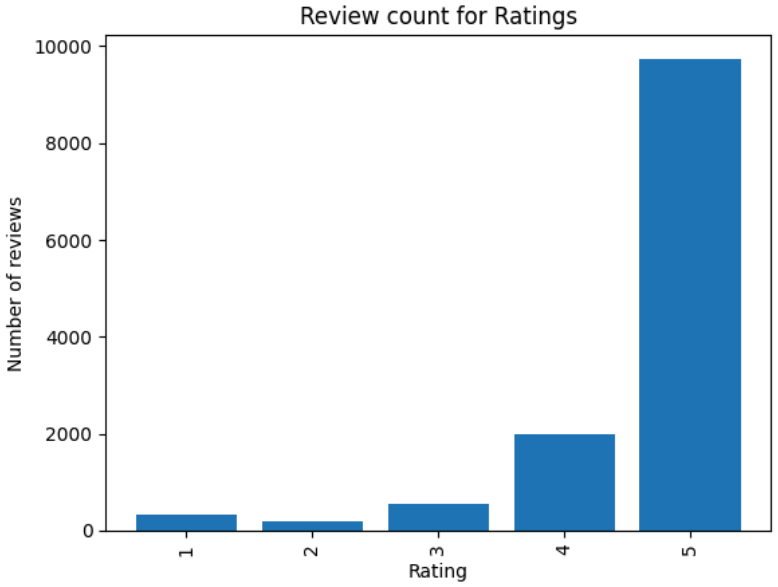

Figure 1: Proportions of ratings in the dataset.

The dataset is heavily biased towards the positive reviews as 76% of the reviews are 5 stars and 15% of the reviews are 4 stars while 3 to 1 star reviews only take up less than 10% of the reviews.

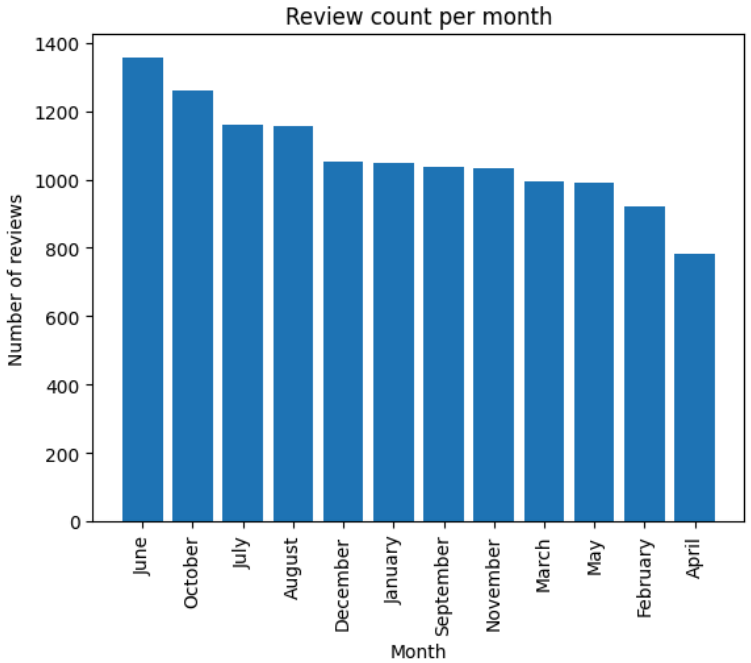

Figure 2: Distribution of reviews by month in the dataset.

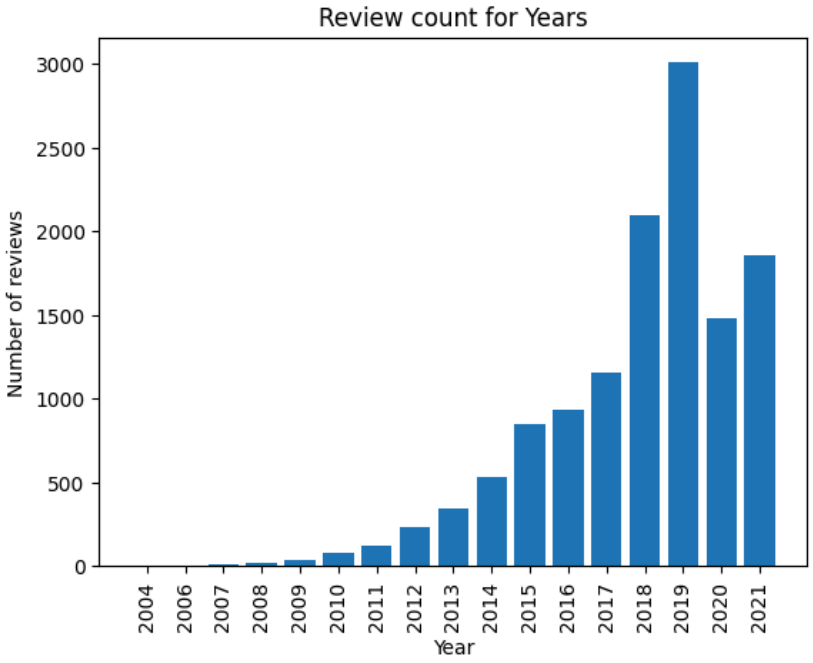

Figure 3: Distribution of reviews by year in the dataset.

There has been a steady increase in the number of reviews for the hotel from 2004 to 2017 and rapid increase in the years 2018 and 2019. However a sudden drop in the number of reviews in the year 2020, possibly due to COVID-19 affecting the tourism and decreasing number of visitors to the hotel.

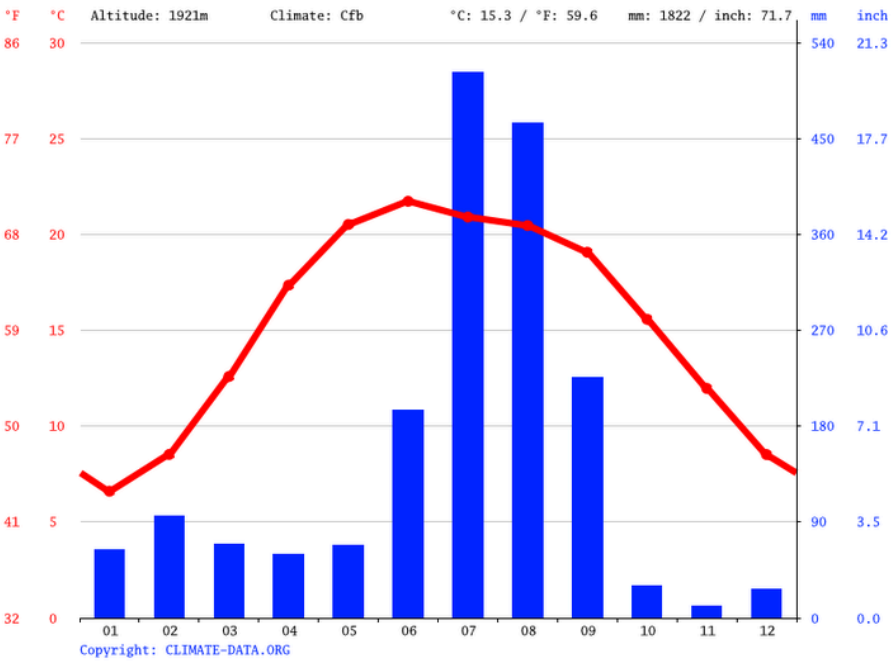

Figure 4: Monthly temperature data in Musoorie, India from climate-data.org

With data from climate-data.org, we notice that there is a big difference in both climate and temperature for different months. Using this information, the dataset is split into two separate datasets,

Hotter months: April, May, June, July, August and September;
Colder months: January, February, March, October, November and December.

The aim is to train the two models separately to see if these models can predict with higher accuracy knowing which season it is. With different climate and temperature it is assumed to have a difference in experience of the stay and therefore a difference in the reviews, with the expectation of there being different things in focus in the reviews.

2.3 Preprocessing the data

The preprocessing included several steps to prepare the text for analysis:

- Cleaning: Duplicates and null values were removed to ensure data quality, while extracting relevant data.
- Text Normalization: Techniques such as tokenization, stop word removal, and lemmatization were employed to standardize the text.
- TF-IDF Vectorization: Text transformed into a numerical format, optimizing the representation for machine learning models.

We implement a review preprocess function that tokenizes the reviews by separating each word in a sentence and applying part of speech tags. Then it removes stop words such as ‘a’ or ‘the’ which add no value to the sentiment and also removes non-alphabetic words, symbols and numbers. Then it performs lemmatization on each token to its original state such as ‘running’ to ‘run’ to avoid creating new tokens for words with the same meaning. Lastly it joins the words which are in list form back to text for vectorization.

Sentiments are also mapped to the ratings given by each reviewer, where ratings 5 and 4 are mapped to positive, 3 as neutral and 2 & 1 as negative. We analyze the dataset further after mapping the sentiments.

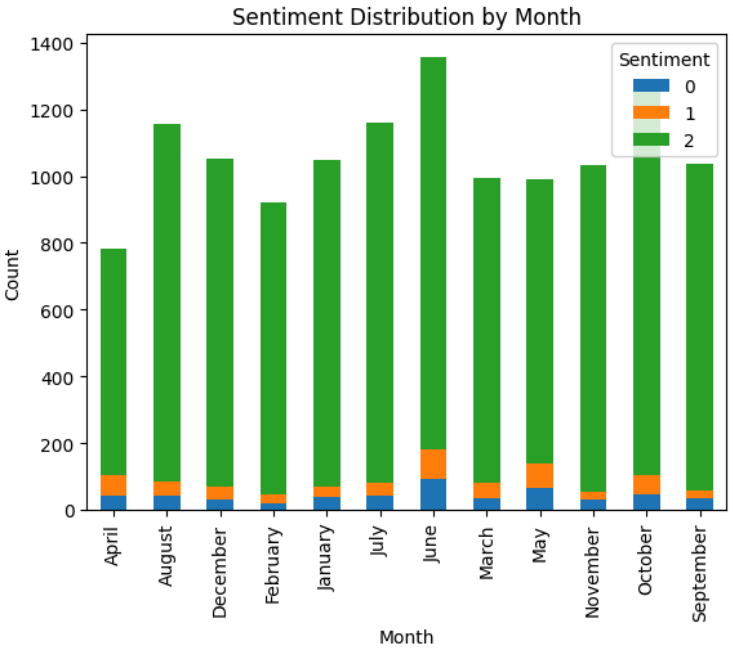

Figure 5: Distribution of sentiment in review by month.

June, May, April and October seem to have a higher proportion of negative and neutral reviews than other months. While April, May and June are holiday seasons (summer) in India. The difference in season could have an impact in the reviews as well and will be further explored in the experiment and results section.

After preprocessing the reviews we can analyze the separated words. We can visualize the frequency of the words appearing in the reviews using word clouds as seen below.

Figure 6: Word Cloud of the positive reviews visualizing the most common words found.

The most common words among the positive reviews which are 5 star and 4 star reviews include mall, road, good, great, resort , hospitality, restaurant, and beautiful.

Figure 7: Word Cloud of the negative reviews visualizing the most common words found.

The most common words among the negative reviews which are 2 star and 1 star reviews include stay, good, staff, service, property, book, food, bad, night.

Figure 8: Word Cloud of the neutral reviews visualizing the most common words found.

The most common words among the negative reviews which are 2 star and 1 star reviews include stay, food, good, mall, road, property. It is quite evident that the words appearing in neutral reviews overlap with most of the positive words, which can add further confusion in the sentiment prediction and removing such reviews may improve model's performance and will be further explored in the project.

3. Machine learning tasks

3.1 Task

The first step is to split the dataset into training and test data before training the model. We chose an 80-20 split for our data, with 80% used for training and 20% for testing. Shuffling was enabled to randomize the data, and stratification was also set to the processed sentiments. This helps keep a consistent proportion of different classes in both the training and testing sets, which is especially useful when dealing with an imbalanced dataset.

The next step is vectorization which is performed after the data split to avoid data leakage and prevent the vectorizer from calculating statistics (e.g., document frequencies) based on the entire corpus of documents instead of just the training documents. TfidfVectorizer, a built -in library from sklearn, is used for converting a collection of raw individual units of text into a matrix of TF-IDF (Term Frequency-Inverse Document Frequency) features. Term Frequency (TF) measures the frequency of a word within each review, the more the word appears the higher the TF value. Inverse Document Frequency (IDF) measures the rarity of a word in the whole collection of reviews, words that appear often in the reviews have lower IDF value. Upon previous exploration and experimentation with the TfidfVectorizer, We have added some parameters to improve the process which includes:

- min_df (10) - This parameter specifies the minimum number of documents , when set to 10, a word must appear in at least 10 documents for it to be considered in the vocabulary.
- max_df (0.8) - sets the threshold frequency (or fraction) above which a term will be ignored. it is set to 0.8, indicating that a term appearing in more than 80% of the documents will be ignored.
- max_features(None) - This parameter determines the maximum number of features (terms) to be included in the vocabulary, setting it to None, results in all features to be considered.
- stop_words('english') - This parameter specifies the stop word list to be used. so the built-in stop word list for English will be used.
- analyzer(word) - This parameter determines whether the TfidfVectorizer should analyze text at the word level or character level.
- ngram_range(1,3) - This parameter specifies the range of n-grams to be considered. An n-gram is a sequence of words of length n. when set to (1, 3) it means that unigrams, bigrams, and trigrams will be considered in the feature extraction process.
- sublinear_tf (1) - This parameter applies a sublinear scaling to the term frequency. When set to 1, it uses a logarithmic form to scale the term frequency, which can help in reducing the impact of very frequent words.

3.2 Methods

Two main methods were employed:

Support Vector Machine (SVM): This model was used for initial training and baseline performance measurements. It efficiently separates different classes by maximizing the margin between the closest data points of each class, known as support vectors, and the hyperplane. This technique ensures high accuracy and robustness, significantly reducing the risk of misclassification. SVM is also versatile, capable of processing various data types through its kernel functions, such as linear, polynomial, and radial basis function (RBF). These kernels help SVM perform in both linear and non-linear scenarios. For text classification, the linear kernel is often preferred because it is effective and computationally efficient when dealing with high-dimensional data, where the relationships between features (words) are primarily linear. Depending on the data's unique attributes, particularly in text classification, the kernel settings can be adjusted to capture complex patterns in large datasets.

XGBClassifier: A class in the XGBoost library specifically designed for classification tasks, stands for eXtreme Gradient Boosting. XGBoost is a popular and powerful machine learning library that utilizes gradient boosting techniques to build predictive models. Gradient boosting is an ensemble method that combines multiple weak prediction models (typically decision trees) into a strong predictive model. It uses an optimized implementation of gradient boosting that efficiently handles large datasets and provides high predictive accuracy.

Initially the XGBClassifier creates a simple model, often called the "base model" or "first learner," which is usually a decision tree with a single node. then it creates a loss for the multiclass classification task. After building the base model, the XGBClassifier starts creating additional weak prediction models called "boosters". And it uses a technique called "gradient boosting" to determine the direction and magnitude of the corrections for each observation in the training data. This gradient indicates the direction in which the predictions need to be adjusted to reduce the loss. The XGBClassifier then builds additional weak learners like decision trees to predict the gradients calculated in the last step. The XGBClassifier repeats the process of creating weak predictors, computing gradients, building weak learners, and combining predictions until a stopping criterion is met. This classifier also incorporates regularization techniques like L1 and L2 regularization.

3.3 Hyperparameter Tuning

Due to the heavy imbalance in the reviews, we adjusted the class weights inversely proportional to class frequencies by initializing the **class weight** parameter in both methods to assign higher weights for neutral and negative reviews than positive reviews. This helps the model give more importance to the minority class. This helps in penalizing

misclassifications of the minority class more than the majority class. The class weights assigned for this project is 1, 7.5 and 7.5 for positive, neutral and negative respectively.

Excluding class weights, the following hyperparameters were tuned manually and the following were made to improve model performance:

For SVM,

- Regularization Parameter - C: This parameter helps balance the trade-off between simplicity and accuracy in the model. Lower values of C create a simpler model that tolerates more errors, which was more effective in our tests.

- Choice of Kernel: Although RBF and polynomial kernels were considered due to their ability to manage non-linear patterns, the linear kernel was found to be the most effective for our text data, aligning with its strengths in handling high-dimensional spaces efficiently.

For XGBClassifier,

- Higher n_estimators which determine the number of trees in the XGBoost ensemble. Increasing the number of trees can potentially improve the model's performance by capturing more complex relationships in the data.

- Higher min_child_weight which controls the minimum sum of instance weight needed in a child, a higher value can help reduce overfitting by adding more regularization to the model.

- Lower gamma which controls the minimum loss reduction reduction required, lowering it allows more splits and helps to capture more complex patterns.

- Higher colsample_bytree which determines the fraction of features to be randomly sampled for each tree. Increasing it introduces more randomness in the trees and may improve the model's performance.

Through GridSearchCV, we found optimal values for these parameters that performed the best for the average F1 score across all sentiments.

4. Experiments and Results

The total accuracy of the models was essentially ignored since the models tend to overfit to the positive reviews considering the heavy imbalance in amount of data. Instead, an average of F1 scores over all sentiments was considered to evaluate the models since the goal was to create a model that effectively could predict all sentiments.

The imbalance of the dataset proved to be the biggest challenge in the project. Initially, models were biased towards predicting the majority class (positive). Through hyperparameter tuning and class weighting, we improved the sensitivity towards minority classes.

4.1 Support Vector Machine (SVM)

Initially the model was trained without any parameters.

Classification Report:
Overall Accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.91	0.61	0.73
Neutral	0.40	0.02	0.03
Positive	0.94	1.00	0.97

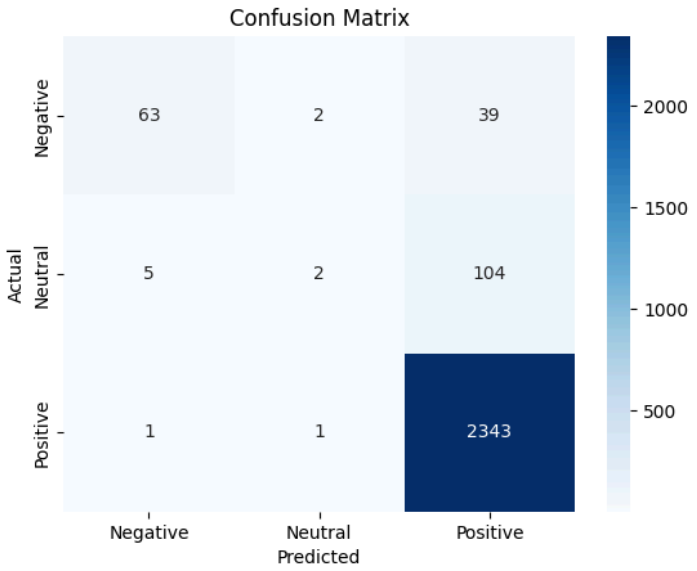

Figure 9: Confusion matrix of running the SVM model without any parameters.

Here the model clearly overfit to the positive reviews, having almost perfect accuracy, but misclassifying all neutral and some negative to positive. Considering the large number of positive reviews that are correctly predicted the model still has a high overall accuracy, showing the fault of using accuracy as the main evaluation score.

4.2 XGBClassifier

Initially the model was trained without any parameters.

Classification Report:
overall accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.76	0.50	0.60
Neutral	0.44	0.11	0.17
Positive	0.95	1.00	0.97

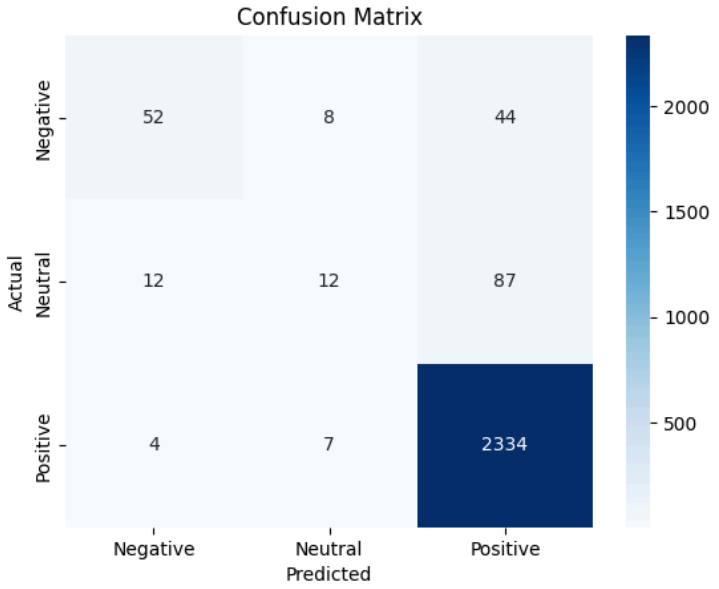

Figure 10: Confusion matrix of running the XGB model without any parameters.

The overall accuracy of the model is 0.94. Although the precision is acceptable with 0.76, 0.58, 0.95 (negative, neutral, positive), the recall is very poor. The recall for positive is 1 and negative is 0.57, However it is only 0.1 for neutral comments with the overall F1 score being 0.65, 0.17, 0.97 (negative, neutral, positive).

4.3 After hyperparameter tuning:

Parameter tuning was focused on optimizing the balance between recall and precision across classes, where Grid Search was used to find the optimal hyperparameters that address both accuracy and class imbalance.

Support Vector Machine (SVM):
Classification Report:
overall accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.81	0.74	0.77
Neutral	0.39	0.50	0.44
Positive	0.98	0.97	0.97

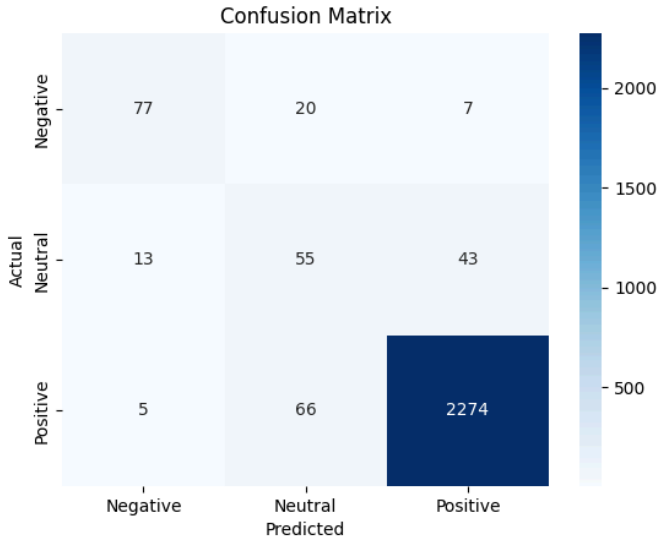

Figure 11: Confusion matrix of the SVM model with optimal parameters.

The recall of both negative and neutral comments improved considerably after hyperparameter tuning. The F1 score improved from 0.73, 0.03, 1.00 to 0.77, 0.44, 0.97 for negative, neutral and positive respectively. Although a slight loss in F1 score for positive reviews, the overall performance of the model improved. However, the model still struggled to predict neutral reviews with precision.

XGBClassifier:

Classification Report:

overall accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.75	0.59	0.66
Neutral	0.44	0.24	0.31
Positive	0.96	0.99	0.98

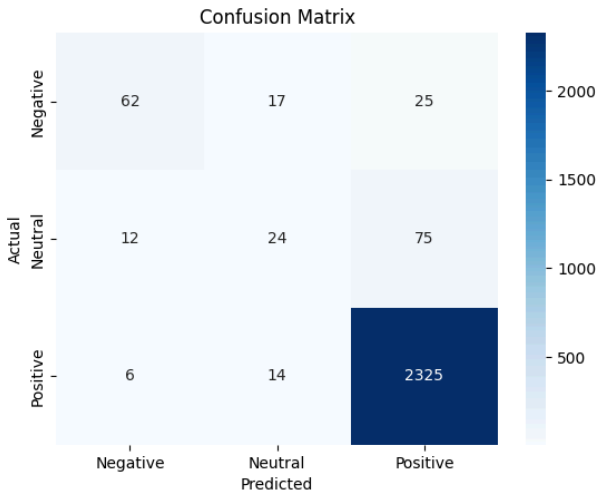

Figure 12: Confusion matrix of the XGB model with optimal parameters.

Although the accuracy is still at 0.94, the recall of both negative and neutral comments have improved quite a bit. The F1 score improved from 0.60, 0.17, 0.97 to 0.66, 0.31, 0.98 for negative, neutral and positive respectively.

4.4 Training model according to season:

Here we train the two datasets that were split according to season. Hot dataset includes: April, May, June, July, August, September. Cold dataset includes: January, February, March, October, November, December.

Support Vector Machine (SVM):
Hot Classification Report:
overall accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.84	0.71	0.77
Neutral	0.51	0.37	0.43
Positive	0.96	0.99	0.98

Cold Classification Report:
overall accuracy: 0.94

Class	Precision	Recall	F1-score
Negative	0.68	0.69	0.68
Neutral	0.41	0.24	0.30
Positive	0.97	0.98	0.98

The hot dataset performed a lot better than the cold dataset, where we see a significant drop in F1 score between the two. However, they both performed worse or similar to the overall dataset, considerably struggling with predicting neutral reviews.

XGBClassifier:
Hot Classification Report:
overall accuracy: 0.93

Class	Precision	Recall	F1-score
Negative	0.73	0.55	0.63
Neutral	0.43	0.14	0.21
Positive	0.94	0.99	0.97

Cold Classification Report:
overall accuracy: 0.95

Class	Precision	Recall	F1-score
Negative	0.69	0.46	0.55
Neutral	0.65	0.24	0.35
Positive	0.96	0.99	0.98

For XGB on the two datasets the cold data performed better looking at the average F1 scores. The hot model performed worse on neutral reviews, while the cold model performed better at the cost of losing accuracy on the negative reviews.

4.5 Removing neutral comments:

Removing all instances where the sentiment = 1 (neutral reviews), then fitting the new data to the SVM and XGBmodel to see if the performance is better.
After the removal, there are 11724 positive reviews and 518 negative reviews

Support Vector Machine (SVM):
Classification Report:
overall accuracy: 0.99

Class	Precision	Recall	F1-score
Negative	0.92	0.88	0.90
Positive	0.99	1.00	1.00

XGBClassifier:
Classification Report:
overall accuracy: 0.98

Class	Precision	Recall	F1-score
Negative	0.90	0.63	0.75
Positive	0.98	1.00	0.99

Both models perform better with this update and the SVM model shows better results than XGB, both models have the same score for positive reviews however the F1 score of negative reviews from the SVM model is better than XGB model. Both models also run faster due to lower complexity from binary classification.

4.6 Removing neutral comments and Training model according to season

Support Vector Machine (SVM):
Hot Classification Report:
overall accuracy: 0.99

Class	Precision	Recall	F1-score
Negative	0.90	0.82	0.85
Positive	0.99	0.99	0.99

Cold Classification Report:
overall accuracy: 0.98

Class	Precision	Recall	F1-score
Negative	0.74	0.72	0.73
Positive	0.99	0.99	0.99

XGBClassifier:
Hot Classification Report:
overall accuracy: 0.99

Class	Precision	Recall	F1-score
Negative	0.88	0.45	0.59
Positive	0.97	1.00	0.98

Cold Classification Report:
overall accuracy: 0.98

Class	Precision	Recall	F1-score
Negative	0.67	0.31	0.42
Positive	0.98	0.99	0.99

The SVM improved significantly for both recall and precision for negative and positive comments from previous seasonal reviews without removing neutral comments. The XGB improved slightly for positive comments but tends to perform worse than before for recall of negative comments, however the precision of the negative comments have improved but the overall F1 for negative is worse with removing neutral.

5. Conclusion

Even with the problems of the dataset such as imbalance, the model still managed to perform with high accuracy, especially after removing neutral reviews. This imbalance however, made it hard to determine if there were any seasonal differences since the seasonal models were trained on datasets with especially few negative and neutral reviews, making the model overfit to positive reviews even with parameters meant to combat imbalance. Even if the models seem to perform with high accuracy, the F1 scores show struggles with predicting neutral reviews. With a model trained on datasets with more balance and generally more data there could easily be trends to be found using the season specific models instead.

While both SVM and XGBClassifier performed robustly, SVM showed a notable improvement in handling data imbalance especially in the seasonal model training possibly due to the smaller dataset. Both models show higher accuracy and faster model training with removing neutral comments due lower complexity of classification from multi-class classification to binary classification and the absence of overlapping words from neutral comments. For future work, a more complete dataset should be considered.

6. Division of Labor

The project tasks were evenly split 50/50 between the two team members. Shahir Imam mainly focused on SVM modeling, including hyperparameter tuning, model evaluation, data preprocessing and seasonal training, while Batmanaman focused on XGBClassifier implementation including hyperparameter tuning, model evaluation, data analysis/exploration and binary model training. Both members collaborated on the concept and writing of the report and the slides.

References

Kaggle dataset source:
https://www.kaggle.com/datasets/ruchibhadauria/hotel-reviews-from-tripadvisor

Chen, T., & Guestrin, C. (2016). XGBoost: A scalable tree boosting system. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (pp. 785-794). ACM.

Climate data in Mussoorie: Mussoorie climate: Weather Mussoorie & temperature by month (climate-data.org)