Supervised Prediction of Beauty Rating Classes Using LIWC-Like Features

Vanshika Sharma
National College of Ireland
School of Computing
Dublin, Ireland
x23198389@student.ncirl.ie

***Abstract*—The global beauty industry is projected to grow significantly by 2030, with online platforms like Amazon playing a major influence in transforming shopping experience for the consumers. This study investigates consumer behavior by predicting product star rating (low, medium, high) using LIWC-like features which have been extracted from review tests. The data was cleaned, correlation analysis was performed to extract important features, and the dataset was split into training and testing sets. Three different algorithms were used to predict rating class, implemented in Python and WEKA. These algorithms produced varied results, influenced by various factors discussed in the report. This report aims to provide deeper analysis into consumer behavior offering valuable insights for various industries. The findings indicate that Random Forest performed the best among all models.**

I. Introduction

Consumer behavior refers to the study of how individuals make decisions related to buying a product. A trend in it has been the increasing dependence on online reviews and personalized recommendation. This study aims on similar ideas by using beauty products as this industry is expected to grow rapidly in coming years. A report on Statista[1] shares that the as of 2025, the global beauty market is evaluated at a price of $677.19 billion, and is expected to grow at a rate of 3.37% from now to 2030.

The aim of this study is to analyze consumer behavior specifically within the beauty product domain by building a classification model that predicts star ratings into three categories, low, medium, and high, based on LIWC-like features which have been extracted from the review text of the product. The focus is to understand how these features can affect customer satisfaction. This goal is not only to build a model but also to study the insights for platforms like Amazon to improve product recommendation, understand customer sentiment and potentially reduce operational cost by identifying areas of customer dissatisfaction.

The analysis follows KDD data mining methodology, ensuring a structured approach from data preparation and feature engineering to model evaluation and result interpretation which will be reflected throughout this report. This research is inspired on a study by Zhong et el[2] and it lead to the foundation of the ideas.

II. Literature Review

The beauty industry is fascinating subject for analyzing consumer behavior through reviews and feature extraction techniques. Several studies have taken diverse approaches to understand this behavior, often using similar types of datasets.

An interesting finding by xyx[3] was that perceived risk significantly influences consumer behavior, prompting consumers to seek online reviews from various outlets. These reviews help consumers make decisions and help in the growth of these restaurants or food items. This study clearly reveals that online reviews significantly influence customer trust, reduce perceived risk, and influence purchasing decisions.

Another study by Aditya[4], using both quantitative and qualitative methods examined how digital strategies affect business performance in the beauty industry. The study highlights how consumers respond to personalized recommendations and explains the importance of authenticity and relevance in marketing. This is closely aligned with the objectives discussed in the introduction section, particularly in developing a model for effective product recommendations.

A survey-based study conducted by Nivetha[5] explored the factors influencing consumer buying behavior. It emphasizes the shift among Indian women from traditional to modern cosmetics, driven by lifestyle changes and trends on online platforms such as Amazon and Temu.

Another very interesting study was conducted by Jignesh[6], which examined the evolving online fashion market by comparing two popular platforms, Amazon and Flipkart The study found that the consumer behavior is influenced by trends on these individual platforms. Interestingly, consumer preferences can shift even when the products are similar, as popularity on one platform may differ from the other potentially due to branding or presentation.

Research by Prosperpio et al[7], analyzed influence of Amazon's Climate Pledge Friendly (CPF) program on consumer purchasing behavior in terms of consumer purchasing behavior. The study reveals that sustainable products are purchased more by consumers, which increases the sales for low-visible products.

III. Data

The dataset used for this research work is based on Amazon's official site, which is a leading e-commerce company known for its customer-centric approach, vast product selection, and efficient delivery network. The dataset used in this study was collected by Jianmo Ni[8] through public web data crawling of the Amazon's site and was made publicly available through UCSD's data repository. It contained records from 1996 to 2018.

While the full page contains multiple product categories, this study specifically focuses on beauty products. Therefore, only relevant category which is 'All Beauty' category was downloaded in JSON format. Two Json files were used one containing review data and other containing product metadata. The metadata file included star ratings under a column named 'Summary', while the review dataset included 'Asin' which contained product's unique code and 'Brand' column.

This study was strictly adhering to the requirements of the CA therefore the most recent years data was filtered and

merged from both review data and metadata files and named into 'cleaned_amazon_reviews_2017_2018.csv'. Alternative datasets from more recent years, such as 2023 and 2024, were considered but ultimately not selected. The 2024 datasets were either synthetically generated or had very common categories for the products (like, household, or kitchenware). Moreover, in case 2023 datasets, they were available in JSONL format, which was difficult to use in preprocessing. To avoid overcomplicating the 2017-2018 dataset was chosen for its relevance, structure and usability.

The merged file was used to extract LIWC-like features using Natural Language Processing (NLP). These features, inspired by the Linguistic Inquiry and Word Count tool, were chosen for sentiment and behavioural analysis by using the reviews column and were later used to predict the star rating of the product based on these features. Some example of these psychological meaningful categories are emotions (e.g., anger, joy), cognitive features(e.g., thinking),etc and these are further discussed in methodology section. The properties of the all the datasets involved in this study have been summarised in a table below:

Original datasets		
	All_Beauty.json	meta_All_Beauty.json
Size	167,034 KB	64,853 KB
Format	JSON	JSON
Complexity	Semi-structured	Semi-structured
No. of Records	371345	32892
No. of Attributes	12	19
After combining into a single dataset		
	cleaned_amazon_reviews_2017_2018.csv	
Size	19,845 KB	
format	CSV	
Complexity	Structured	
No. of Records	57449	
No. of Attributes	6	
Final dataset for modelling with LIWC-like features		
	ml_model_dataset.csv	
Size	3544	
format	Csv	
Complexity	Structured	
No. of Records	15,657	
No. of Attributes	15	

Table 1. Summary of Datasets

IV. METHODOLOGY

The methodology used for the analysis was KDD which stands for knowledge Discovery in Databases. It is a data mining methodology which is useful to extract knowledge from large volumes of data. It involves several key steps which help to convert raw data into meaningful information.

A. Selection

To adhere to the objective as discussed in the Introduction, the goal of the selection stage was to get a dataset related to beauty products containing customer reviews and star ratings to build a machine learning model which could predict star rating.

This involved thorough research across many different platforms to get the dataset, including, Google dataset, Kaggle, UCI ML repository, etc. Many datasets were gathered and evaluated for the requirement alignment with goal of the study. This involved answering several key questions during the selection process, for e.g., whether the dataset was suitable for a research problem? Is the format easy to use or manipulate in Python, or compatible with GUI-based tools like WEKA? Is the dataset recent?

Finally, a dataset was found through a webpage created by Jianmo Ni, who had scraped and compiled reviews in JSON format. This dataset seemed the most suited data, as it was not synthetically generated and met all the criteria expect being a recent dataset. Later, the idea was to create LIWC-like features, which suited the best for this current dataset. These features are more insightful and interpretable than traditional feature extracting methods.

There was a similar study available on Google Scholar on consumer behaviour based on beauty products by Mingo[9], which used NLP to extract TF-IDF (Term Frequency-Inverse Document Frequency) based features and applied machine learning algorithms like logistic regression to classify the reviews into positive, negative or neutral sentiments. However, this paper differentiates itself by focusing on LIWC-like features instead of TF-IDF, offering a deeper, psychologically study on consumer behavioural analysis.

B. Preprocesing

In this stage, the task was to prepare the dataset ready for analysis, therefore, both the review data and the metadata of products were downloaded from the UCSD's data repository, all the relevant columns were combined into a single dataset. It was saved into a csv file using Spyder and named as 'cleaned_amazon_reviews_2017_2018.csv' which has information of all the beauty products from year 2017 to 2018.

To transform raw text into meaningful numerical representation, feature engineering was performed. The above-mentioned file was later used to extract LIWC-like features, which refers to the linguistic metrics inspired by the Linguistic Inquiry and Word Count (LIWC) tool, where text was converted into psychological, emotional, and cognitive states to capture deep insights for the analysis. The final working dataset was named as 'ml_model_dataset.csv' and this was coded using Jupyter Notebook.

The final dataset was loaded using pandas and converted into a DataFrame for manipulation. All the missing values were checked, and any row containing empty cell was dropped. All the duplicate values were dropped as well. This was confirmed using the shape function, it was observed that the size of the DataFrame changed from originally being (15657,15) to (10896, 15) after data cleaning. Later, the classes for the 'rating_class'(target variable) were checked and found to be imbalanced. Therefore, various strategies were considered to address the issue, as many machine learning algorithms are sensitive to class imbalance. Moreover, balancing the classes helps to build in building a less biased model and reduces the risk of overfitting to the majority class.

In Python, some columns were used to generate new columns, which contained new features like the target variable. Lastly, NLP was used to extract LIWC-like features, and these were going to be used as the predictors in the machine learning models. However, in WEKA, it was done using filters in Preprocess tab.

C. Transformation

In this step, correlation analysis was conducted to assess the relationships between features. The goal was to identify and remove highly correlated features to avoid multicollinearity. However, most of the values were within the range of -0.07 to 0.24, therefore, no features were dropped. The features were all converted into a new DataFrame and named X, which is a traditional variable name for predictors, and the target variable was converted into a DataFrame as well and named y, for the same reason.

The target variable was encoded using LabelEncoder from sklearn.preprocessing as it was a categorical variable, since there were three categories, the values were converted into numbers, 0, 1 and 2. Later, the numerical attributes were scaled using StandardScaler from sklearn, which ensures that the features contribute equally to the model training and not one feature dominates the results. This made the dataset ready for training the models. In WEKA, transformation was done easily using the built-in tabs and it was easy to standardize these values and label encoding them.

D. Data Mining

As per the CA's requirement, a supervised classification model was to be implemented. Since the problem was predicting a star rating, which is a multiclass classification problem. To prepare for model training, the dataset was split into training and testing sets using the train_test_split function from scikit-learn, with an 80-20 split, which is a common standard academic practice. The stratify parameter was applied to ensure that the class distribution sets remain the same as the original dataset. Since the rating_class distribution was imbalanced, it was important to balance the classes in the datasets for predictors and target variables. It was done by oversampling the data, using SMOTE(Synthetic Minority Over-sampling Technique) from imbearn library.

With the prepared data, three classification models were trained and evaluated, the first model used was a multiclass Logistic Regression, implemented using LogisticRegression from sklearn.linear_model. Then, the next model was a KNN (K-Nearest Neighbors) model which is a non-parametric algorithm and classifies a data point based on the majority class of its nearest neighbors. The model was implemented using KNeighborsClassifier from sklearn.neighbors and was tested with 5 neighbors. Lastly, a Random Forest, was implemented using RandomForestClassifier from sklearn.ensemble in Python. All these three models were evaluated in WEKA as well, but it was used as a secondary tool, more details about it are discussed in the implementation and results section of the report.

E. Evaluation and Interpretation

To ensure the robustness and generalizing capability of the trained models, multiple evaluation strategies were applied. Each model was tested in different versions, both balanced and imbalanced, with and without applying cross-validation and hyperparameter tuning.

Logistic Regression showed better performance after applying SMOTE, with cross-validation and hyperparameter tuning enhancing its performance. K-nearest Neighbors performed best on balanced data but was sensitive to class distribution. Random Forest consistently delivered the highest F1-scores across all the scenarios, while being more resistant to overfitting. All these observations are discussed in detail in results section, for both Python and WEKA.

V. IMPLEMENTATION

All the machine learning models, and creation of LIWC-like features were created using Python across two different environments: Spyder and Jupyter Notebook. Primarily, Spyder was used to combine the two JSON files consisting of the review data and the metadata into a single CSV file. This was necessary because one dataset contained the reviews and the other data contained 'summary' column, which included the star ratings.

The dataset created was then modified and prepared for training the model. Since the dataset didn't have star ratings in numeric form, the first step was to create a new column named 'star_review'. A function was designed to parse through the 'summary' column and extract the star ratings, for example for 'two stars' was converted into 2, and 'four stars', was converted into 4, etc. If a cell in the 'summary' column contained no word 'star' in its cell, then the function returned None, and those rows were later dropped. This new column was converted into int type and then mapped into categorical classes, namely, low, medium, or high based on the numeric values in the 'star_review' column. For example, the numbers '1' and '2' were mapped to the low class, '3' was mapped to medium, and '4' and '5' to high, indicating good reviews. There was a total of five possible star values. These new categorical values were stored in a new column called 'rating_class'.

The next task was to create LIWC-like features, it was done using Natural Language Processing (NLP). Important libraries such as pandas (for data loading, manipulation, and analysis using DataFrames), NumPy (for efficient numerical operations and array handling), skikit-learn(for building, training, and evaluation machine learning models), nltk (for NLP tasks like tokenization and sentiment analysis) and tqdm (for displaying progress bars) were installed and used in Jupyter Notebook. All the rows with missing values and duplicates were dropped from the initial dataset named 'cleaned_amazon_reviews_2017_2018.csv'.

Since the LIWC-like features had to be obtained from 'reviewText' column, which contained product reviews, all reviews were first converted to lowercase so that both 'A' and 'a' were treated equally. A dictionary was then created with seven features acting as keys, namely, 'i_words_list', 'positive_words', 'negatve_words', 'social_words', 'cognitive_words', 'allure_words', and 'moralization_words'. Each key held a list of words representing the linguistic feature, for example, 'cognitive_words' included words like 'think', 'know', 'reason', etc. Using nltk, the text in 'reviewText' column was tokenized, and a function was created to iterate over the tokens and count the occurrences. To visualize the progress of this operation, tqdm was used. The new dataset was saved and later named as 'ml_model_dataset'.

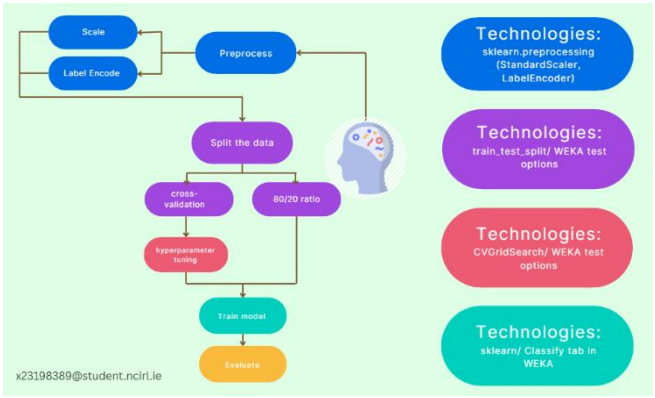

Fig 1. Visual Chart of Workflow

Three different machine learning models, namely, Logistic Regression, K-Nearest Neighbors, and Random Forest, were built using Python in Jupyter Notebook and GUI-based tool WEKA. To prepare the data for training these models, a common structure was followed across all implementations. This included installing the required libraries using pip, loading and exploring the data using

Feature	Machine Learning Models		
	Logistic Regression[11]	***KNN[12]***	***Random Forests[13]***
Algotihtm Type	Linear Classifier	Instance-based	Ensemble (Bagging, Decision Trees)
Main Library	Sklearn.linear_model.LogisticRegression/ Functions.Logistic	Sklearn.neighbors.KNeighborsClassifier/ Lazy.IBk	Sklearn.ensemble.RandomForestClassifier/ trees.RandomForest
Preprocessing	Feature Scaling (StandardScaler/Normalize)	StandardScaler, LabelEncoder	same
Handling Imbalance	SMOTE from imblearn	SMOTE from imblearn	SMOTE from imblearn
Train-Test Split	train_test_split	Same	Same
Cross Validation	cross_val_score/ Classify tab in Weka	StratifiedkFold/ Classify tab in Weka	Cross_val_score/ WEKA classify tab
Hyperparameter	CVGridSearch (for C values) / Ridge (R) & Max Iteration (M)	GridSearchCV for k/ k-nearest neighbor, WEKA test options	GridSearchCV for n_estimators, max_depth (Python)/ WEKA test options
Visualization	matplotlib, learning_curve from sklearn.model_selection	Matplotlib, confusion matrices	Matplotlib
Automation	Manual Scripting (in Python)/ GUI-based in WEKA	Semi-automated in both	Semi-automated in both
Strength	Interpretable coefficients	Simple, Intuitive	Robust to overfitting
Weakness	Assumes lineairty	Sensitive to irrelevant features	Slow on large datasets

Table 2. Comparison of techniques for different ML models

pandas, and creating visualizations using matplotlib. Data cleaning was performed, followed by feature selection based on correlation analysis. Feature and target variables were then selected, and models were built using different versions. The detailed results of these models will be discussed in the results section. There is a summary table which has been included, listing in detail all the packages, tools, and technologies used in building the models..

To investigate consumer behavior based on reviews, the seven LIWC-like features were used and explored into WEKA Explorer, the dataset was converted into .arff format using Python, which is compatible in the WEKA environment. Although CSV files are also supported in WEKA, due to complications in the dataset, WEKA was unable to upload it properly. Therefore, the safest approach was to convert it into its most compatible format. To do this, a code snippet uploaded on GitHub by anaavila[10] was used. All unnecessary columns were dropped using the Preprocess tab in WEKA, and using the Classify tab , different classifiers were selected to build machine learning model with rating_class variable of nominal type.

All the machine learning models in Python followed a common structure, which began with using pandas for data loading, exploring, and cleaning the data. To reduce the risk of overfitting, feature selection was performed, as it is an important step in building efficient models. A Python function was written to remove all attributes that were highly correlated, as high correlation increases the risk of multicollinearity and leads to unpredictable results. This was verified using heatmap chart, which clearly indicated that no two attributes had a high correlation. The table below shows the correlation between each LIWC feature.

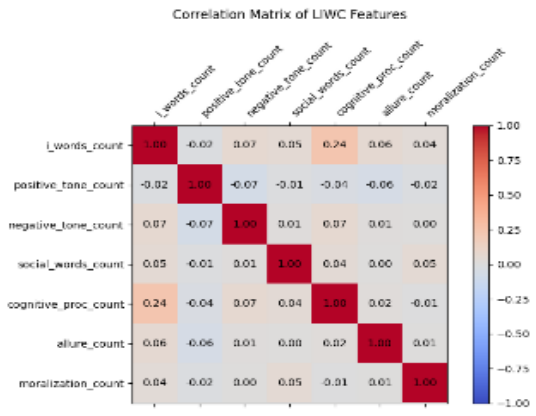

Fig 2. Correlation Matrix of LIWC Features

This dataset was split into two parts and later divided into an 80/20 ratio using train_test_split, where 80% of the data was used for training and 20% for testing and evaluation. The modelling aspect is discussed in the following paragraphs for each model. These models were then trained for balanced data by oversampling the classes to ensure equal representation. Finally, cross-validation and hyperparameter tuning were applied to evaluate which model performs the best. Several plots were generated using matplotlib, and the findings will be discussed in the Result section.

The first model built was Logistic Regression, which was implemented in Python using pandas for exploration and data cleaning. As discussed in the previous paragraph, in Python, the dataset was split into an 80/20 ratio. Similarly, for the same version of the data that is the imbalanced data, WEKA was used, the split was done using the Test Options section under the Classify tab. For modeling, the scikit-learn library was used to scale and split the data, perform label encoding, and implement the model. The model was built using LogisticRegression module from sklearn.linear_model in Python and LogiticRegression Classifier in WEKA. To enhance performance, cross-validation was applied, in Python, this was done using scikit-learn's cross_val_score, while in WEKA, it was done directly in the Classify tab.

Later, hyperparameter tuning was applied in both environments. In WEKA, this was achieved using WEKA's CVParameterSelection meta-classifier, which evaluates parameter values using 10-fold stratified cross-validation. The ridge parameter was tuned across the range of 1.0E-4 to 0.01 in five steps. However, in Python, it was performed using GridSearchCV, which evaluates all possible parameter combinations to find the best one.

Since Logistic Regression is sensitive to class imbalance, therefore, a Python code inspired through a page in GitHub[14] was used to apply SMOTE from imblearn library to oversample the minority class and balance the dataset. All the same steps, preprocessing, splitting, modeling, cross-validation, and hyperparameter tuning were repeated on the balanced data, and model was evaluated accordingly.

The second model was KNN (K-Nearest Neighbors), a simple, non-parametric yet effective model. It was built in python using scikit-learn, following a structure very similar to the previous model. The data was loaded and cleaned using pandas. All the common steps were followed, then, for modelling, KNeighborsClassifier from sklearn.neighbors in Python with n to be five and in WEKA, it was done using Lazy.IBk, later to get a better model, cross-validation and

hyperparameter tuning was applied and it turned out that the model performed the best with best number of neighbors to be 11. This was observed in both environments. In WEKA, the application for applying hyperparameter tuning was easy to implement as it provides direct option in Test Options in Classify tab.

The last model was built using Random Forest because it supports complex, non-linear interactions and is resistant to overfitting and handles noisy data well. The workflow aimed to support flexibility and reproducibility while working with unbalanced datasets. All the same steps were followed for loading and preprocessing, but the model was built using RandomForestClassifier in Python and in WEKA it was built using RandomForest algorithm available under 'trees' classifiers. The table shows the different technologies and libraries used.

VI. Results

The result of these models has helped to partially address the objectives discussed in the introduction section. The key evaluation metrics for imbalanced data, both with and without cross-validation, in Python as well as in WEKA, are presented. As discussed in the implementation section, three different models were built, and their results have been summarized in the table below.

In Python						
	W/o cross-validation & hyperparameter tuning			With cross-validation & hyperparameter tuning		
	Logistic Regression	*K-Nearest Neighbors*	*Random Forest*	*Logistic Regression*	*K-Nearest Neighbors*	*Random Forest*
Accuracy	*0.77*	*0.76*	*0.50*	*0.76*	*0.76*	*0.76*
Precision	0.52	*0.52*	*0.45*	*0.59*	*0.44*	*0.70*
Recall	*0.35*	*0.35*	*0.47*	*0.33*	*0.35*	*0.76*
F1-score	*0.33*	*0.33*	*0.39*	*0.29*	*0.32*	*0.33*
In WEKA						
	W/o cross-validation & hyperparameter tuning			With cross-validation & hyperparameter tuning		
	Logistic Regression	*K-Nearest Neighbors*	*Random Forest*	*Logistic Regression*	*K-Nearest Neighbors*	*Random Forest*
Accuracy	*0.77*	*0.77*	*0.77*	*0.78*	*0.76*	*0.76*
Precision	*0.73*	*0.73*	*0.73*	*0.77*	*0.76*	*0.70*
Recall	*0.77*	*0.77*	*0.78*	*0.77*	*0.77*	*0.76*
F1-score	*N/A*	*N/A*	*N/A*	*N/A*	*N/A*	*0.68*

Table 4. Evaluation of Imbalanced Data

In WEKA, several other metrics were available for evaluation, such as Kappa statistics, mean absolute error, and root relative squared error, but they were not effective for this task as they led to biased models even though some models achieved high accuracy and low error rates (like RMSE over 90%). Therefore, the evaluation was primarily focused on the F1-score(macro average), as it treats all classes equally by averaging the F1-score for each class, ignoring the class imbalance. Additionally, recall was used as well as it measures how many true positive cases were correctly identified, which is particularly important for evaluating the performance on the minority class. As expected, models trained on the imbalanced data without cross-validation showed inflated accuracy but very low F1-scores, clearly highlighting their biased nature towards the majority class. This shows a limitation of using accuracy as a standalone metric, particularly for imbalanced data. While a model may appear to achieve a high accuracy rate, it may still be underperforming or completely ignore the minority class. For example, for Logistic Regression, the accuracy was 76%, but the F1-score was only 0.29, indicating that while it

Both Python and WEKA showed a little disparity in the results due to differences in how the data was split. In Python, the split was performed using a package, yielding 2180 instances, while in the case of WEKA, it was done internally, resulting in 2179 instances. Even though the difference is very minute yet the results varied significantly, as seen in the case of the Random Forest model. It was surprising to observe a significant improvement in the F1-score, increasing from 0.33 in Python to 0.68 in WEKA, providing a much stable model.

An important objective of this project was to assess how class balancing and evaluation strategies like cross-validation affect model reliability. The performance of the KNN model was disappointing due to its poor results, it was unexpected as it was supposed to perform the best among all the models. This was expected based on a study conducted by students of the Institute of Technology in India[15] which analyzed consumer behavior through sentiment analysis using various machine learning models. In the study, KNN was the highest F1-score(0.91) achieving model, followed by Random Forest. However, the findings from this study showed that KNN performed equally poorly as Logistic Regression, from which it was expected due to its sensitivity towards class imbalance, and its nature as a linear classifier, whereas LIWC features show non-linear patterns.

After applying cross-validation and balancing the dataset, several key observations were made for all three models. Validation curves for Logistic Regression showed the best performance when the regularization parameter C was 0.1, while KNN performed optimally at n=11, which was also supported by its validation plot here the F1 macro score peaked. Even though in the result table, the F1 is 0.32 for it but that is because maybe the model was overfitted and doesn't generalize the actual test set or maybe due to difference in data split. For Random Forest, the best results in Python were achieved with n_estimators = 100. However, in WEKA, despite using only 10 estimators, Random Forest showed strong performance on imbalanced data, achieving an F1-score of 0.678, which was the highest among all the models.

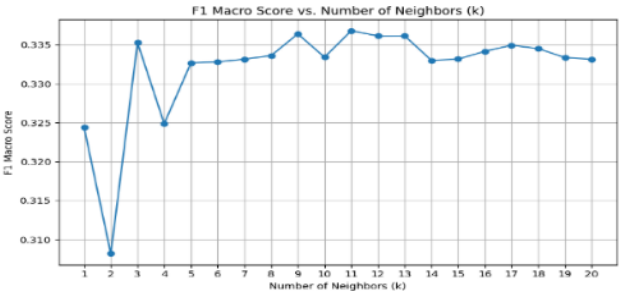

Fig 3. Validation Curve for KNN

To visualize the performance of the three models together, ROC-AUC curves, and it showed that Random Forest achieved the highest AUC (0.68), indicating it has the highest ability to distinguish between classes. Also, it can be interpreted that KNN and Logistic Regression underperformed, which may be due to their sensitivity to class imbalance and linear assumptions as in case for Logistic

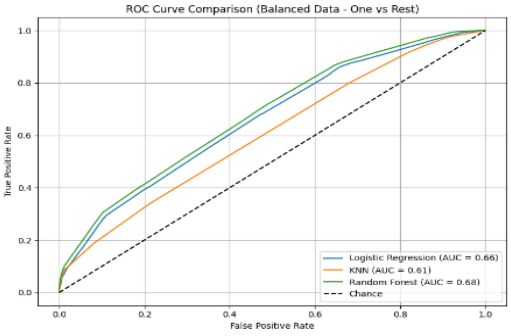

Fig 4. ROC-AUC Curve for three models

Regression. The table below shows the result with and without cross-validation for the balanced data in Python, due to time restrictions only Python was used for the balanced data.

	W/o cross-validation & hyperparameter tuning			With cross-validation & hyperparameter tuning		
	Logistic Regression	***K-Nearest Neighbors***	***Random Forest***	***Logistic Regression***	***K-Nearest Neighbors***	***Random Forest***
Accuracy	0.49	0.49	0.49	0.76	0.49	0.49
Precision	0.52	0.51	0.44	0.59	0.51	0.44
Recall	0.49	0.49	0.46	0.33	0.49	0.46
F1-score	0.43	0.43	0.39	0.29	0.41	0.39

Table 5. Evaluation of Balanced Data

It can be seen from the table that the overall performance of the model contradicted after balancing the data, Logistic Regression and K-Nearest Neighbors seem to perform better than Random Forest. Moreover, in case of logistic regression the performance depreciated from 0.43 to 0.29, this depicts that cross validation and hyperparameter do not promise to yield a better result because sometimes, even techniques like SMOTE can yield overfitting on synthetic data. Amongst these three models, KNN seems to be the most reliable one, even though the F1-score is low, but it doesn't change drastically before and after applying the cross-validation . This can be observed even by plotting the confusion matrix for the three models before cross-validation

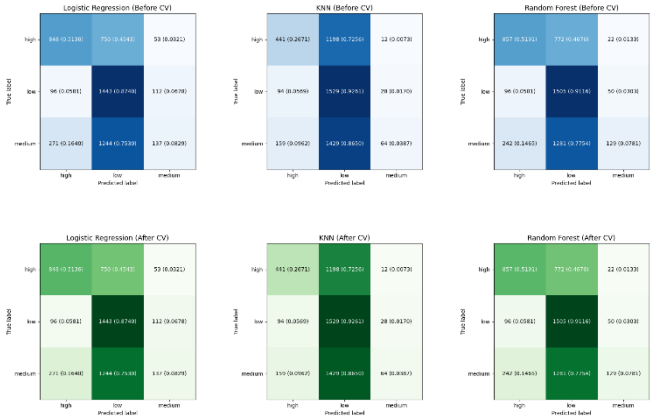

Fig 5. Confusion Matrix of KNN before and after cross-validation & hyperparameter tuning

From a consumer behavior prediction standpoint, the study highlights that linguistic feature, when properly engineered and normalized, can significantly help in predicting star ratings. The results show that models like K-Nearest Neighbors (KNN) and Logistic Regression show improved performance after applying class balancing techniques like SMOTE. However, Random Forest performed better on original imbalance dataset, suggesting that it is not sensitive to class imbalance and yields better results without any biased nature which could have been arose due to oversampling.

VII. Conclusions and Future Work

One of the major learnings from this project is that no matter how many advanced techniques are applied, the quality and fairness of the dataset remain the most critical factors in building a reliable machine learning model. A biased or imbalanced dataset can significantly affect model's performance, even after using techniques like cross-validation and hyperparameter tuning. These techniques are helpful but do not guarantee to yield desirable results if the foundational data is imbalanced as techniques like SMOTE can just make the data biased by overfitting the majority class. If more time had been available, additional machine learning models could have been explored, mainly those which are less robust to class imbalance. Moreover, alternative GUI-based tools would have been explored and applied as well. Another potential extension of the project would have been trying different products which haven't been explored much. If given the opportunity to redo this project, a new approach would be applied where the data would be gathered from scratch using web scraping techniques for recent years data across multiple platforms so that the results of the project are not biased towards the reviews on one site. This would allow for a more diverse and generalized model. Overall, the project offered valuable insights into the importance of data preprocessing, class balancing, and model evaluation, which also gave experience on dealing with advanced techniques like using NLP for feature extraction which increases study's relevance in producing a meaningful model for consumer behavior analysis.

References

[1] "Beauty & Personal Care - Worldwide | Market Forecast," Statista. Accessed: Apr. 18, 2025. [Online]. Available: http://frontend.xmo.prod.aws.statista.com/outlook/cmo/beauty-personal-care/worldwide

[2] Q. Zhong, S. Liang, L. Cui, H. K. Chan, and Y. Qiu, "Using online reviews to explore consumer purchasing behaviour in different cultural settings," *Kybernetes*, vol. 48, no. 6, pp. 1242–1263, Oct. 2018, doi: 10.1108/K-03-2018-0117.

[3] T. Lam, J. Heales, and N. Hartley, "The role of positive online reviews in risk-based consumer behaviours: an information processing perspective," *Aslib J. Inf. Manag.*, vol. 77, no. 2, pp. 282–305, Oct. 2023, doi: 10.1108/AJIM-03-2023-0102.

[4] Hemachandra and A. Kusuma, "Unveiling the Impact Digital Marketing Strategies and Business Performance in the Beauty Industry," *J. Econ. Manag. Bus. Technol.*, vol. 2, no. 2, Art. no. 2, Mar. 2024, doi: 10.35335/jembut.v2i2.206.

[5] M. A. R. Nivetha, "A STUDY ON CONSUMER BUYING BEHAVIOUR IN COSMETIC PRODUCTS WITH RESPECT TO AMAZON," vol. 13, no. 18, 2024.

[6] J. Vidani, "The Evolution of Fashionable Products in online Retailing with the Focus on Amazon and Flipkart," Apr. 30, 2024, *Social Science Research Network, Rochester, NY*: 4849804. doi: 10.2139/ssrn.4849804.

[7] D. Proserpio, A. Goli, T. Mangini, K. Lau, and D. Yu, "The impact of sustainability programs on consumer purchase behavior: Evidence from Amazon," Dec. 05, 2024, *Social Science Research Network, Rochester, NY*: 5045830. doi: 10.2139/ssrn.5045830.

[8] jianmo Ni, "Amazon reviews dataset · Issue #30627 · ClickHouse/ClickHouse," GitHub. Accessed: Apr. 09, 2025. [Online]. Available: https://github.com/ClickHouse/ClickHouse/issues/30627

[9] H. Mingo, "Enhancing Sentiment Analysis with Logistic Regression and NLP: A Case Study of 2023 Amazon Software Reviews," in *2024 International Symposium on Networks, Computers and Communications (ISNCC)*, Oct. 2024, pp. 1–8. doi: 10.1109/ISNCC62547.2024.10759007.

[10] Ana, *anaavila/convert-csv-to-arff*. (Apr. 08, 2025). Python. Accessed: Apr. 21, 2025. [Online]. Available: https://github.com/anaavila/convert-csv-to-arff

[11] "Machine-Learning-with-Python/Logistic Regression in Python - Step by Step.ipynb at master · susanli2016/Machine-Learning-with-Python," GitHub. Accessed: Apr. 24, 2025. [Online]. Available: https://github.com/susanli2016/Machine-Learning-with-Python/blob/master/Logistic%20Regression%20in%20Python%20-%20Step%20by%20Step.ipynb

[12] "scikit-learn/sklearn/neighbors/_classification.py at main · scikit-learn/scikit-learn." Accessed: Apr. 24, 2025. [Online]. Available: https://github.com/scikit-learn/scikit-learn/blob/main/sklearn/neighbors/_classification.py

[13] 262588213843476, "Random Forest Classification with Python and Scikit-Learn," Gist. Accessed: Apr. 24, 2025. [Online]. Available: https://gist.github.com/pb111/88545fa33780928694388779af23bf58

[14] "kntb0107/hyperparameter-tuning-with-logistic-regression: Project made for Optimisation and Deep Learning course." Accessed: Apr. 20, 2025. [Online]. Available: https://github.com/kntb0107/hyperparameter-tuning-with-logistic-regression/tree/main

[15] V. Shrirame, J. Sabade, H. Soneta, and M. Vijayalakshmi, "Consumer Behavior Analytics using Machine Learning Algorithms," in *2020 IEEE International Conference on Electronics, Computing and Communication Technologies (CONECCT)*, Jul. 2020, pp. 1–6. doi: 10.1109/CONECCT50063.2020.9198562.