Leveraging Sentiment Analysis of Steam Reviews for Growth in Game Success Metrics

Applied Research Project by

[Vedant Tomer]

Under the Guidance of

[Vivek Kshirsagar]

Applied Research Project submitted in partial fulfilment of the requirements for the degree of

[Master of Science in Business Analytics]

at Dublin Business School

August 2024

DECLARATION

I, Vedant Tomer, formally declare that the work presented in this research project, "Leveraging Sentiment Analysis of Steam Reviews for Growth in Game Success Metrics" is solely my own. The research conducted, methodologies employed, and conclusions drawn are all a reflection of my independent academic work in Business Analytics.

I further confirm that all external sources of data and ideas used in this research project have been properly cited and referenced. Any assistance received during the course of this research has been duly acknowledged in the acknowledgments section.

I recognize the importance of upholding academic integrity and declare that the content of this research project has not been submitted in whole or in part for consideration towards any other degree or certification at any other university.

Signed: VEDANT TOMER

Student ID: 20015122

Date: 27th August 2024

ACKNOWLEDGEMENT

I would like to sincerely thank Vivek Kshirsagar, my supervisor, for his excellent mentorship, persistent support, and valuable advice throughout the entire research process. His insights and guidance have been instrumental in shaping the direction of this project.

I also wish to express my heartfelt gratitude to my family for their unwavering encouragement and compassion during this challenging journey. Their belief in my abilities has been a constant source of motivation.

My appreciation extends to my friends, whose support during the highs and lows of my academic pursuit provided invaluable advice, encouragement, and insights. Their friendship brought a meaningful dimension to this research experience.

Lastly, I would like to thank everyone who contributed, directly or indirectly, to the successful completion of this research project. Your assistance has been greatly appreciated, and I am deeply grateful for the collaborative spirit that enriched this academic endeavour.

ABSTRACT

This research explores the potential of sentiment analysis and machine learning in predicting game recommendations and review helpfulness on Steam. We applied both lexicon-based (VADER) and deep learning (DistilBERT) methods to extract sentiment scores from game reviews. These scores, combined with review metadata, were used to train, and evaluate Linear SVC and Multinomial NB models, utilizing 10-fold cross-validation with precision as the primary metric. Our results demonstrate that incorporating sentiment analysis enhances prediction accuracy, particularly when addressing class imbalance using SMOTE. This study advances the understanding of sentiment analysis in the gaming industry, highlighting its potential for personalized recommendations and prioritizing valuable user feedback. By leveraging these techniques, game developers and platforms can better understand player preferences and enhance user experiences.

Keywords: Sentiment Analysis, Game Reviews, Recommendation Systems, Machine Learning, Support Vector Machine, Naive Bayes, Natural Language Processing, User Feedback.

Table of Contents

DECLARATION 1
ACKNOWLEDGEMENT 2
ABSTRACT 3
1 Introduction 6
1.1 Background 6
1.2 Research Project Questions 7
1.3 Research Objectives 8
1.4 Scope and Limitations 8
1.4.1 Scope 8
1.4.2 Limitations 8
2 Literature Review 10
3 Methodology 13
3.1 Data Collection 13
3.1.1 Steam Review Data 13
3.2 Data Preprocessing and Feature Engineering 15
3.2.1 Text Cleaning 15
3.2.2 Tokenization and Lemmatization 16
3.2.3 TF-IDF Vectorization 17
3.3 Sentiment Analysis 18
3.3.1 Lexicon-Based Approach (VADER) 18
3.3.2 Deep Learning Approach (DistilBERT) 19
3.4 Machine Learning 19
3.4.1 Model Selection 19
3.4.2 Class Imbalance Handling 20
3.4.3 Model Evaluation 21
3.5 Ethical Considerations 21
4 Results, Evaluation and Discussion 22
4.1 Steam Review Data Acquisition 22
4.1.1 Targeted Data Collection 22
4.1.2 Navigating the Steam API 22
4.1.3 Parameterization and Batching 22
4.1.4 Metadata Enrichment 23

4.1.5 Structured Storage and Data Organization 23
4.2 Streamlit Sentiment Analysis Tool 23
4.2.1 Individual Text Analysis 23
4.2.2 CSV File Analysis 24
4.2.3 Concepts of the tool 25
4.3 Sentiment Analysis: A Comprehensive Multi-Model Approach 25
4.3.1 Data Preparation and Preprocessing 25
4.3.2 TextBlob: An Initial Lexicon-Based Analysis 25
4.3.3 VADER: Context-Aware Lexicon Analysis 26
4.3.4 DistilBERT: Fine-Tuned Deep Learning for Sentiment Classification 26
4.3.5 Integration and Comparison of Results 26
4.3.6 Structured Storage and Output 27
4.4 Machine Learning Model Performance 27
4.5 Unveiling Insights from Steam Review Data: A Multifaceted Exploration 30
4.5.1 Distribution of Recommendations 30
4.5.2 Top Games and Comments: Decoding Community Preferences 31
4.5.3 Word Cloud Visualization: Mapping the Conversational Landscape 32
4.6 Discussion and Implications 34
4.7 Future Scope 36
5 Conclusion 37
6 Appendix 38
6.1 Steam Reviews Web Scrapping Code 38
6.2 Streamlit Sentiment Analysis Tool Code 39
6.3 SVC and NBC Code 40
6.4 Recommendation Code 45
6.5 Sentiment Analysis Code 48
7 Bibliography 50

1 Introduction

1.1 Background

The video game industry is a rapidly evolving landscape, with the success of a game hinging on various factors, including gameplay, graphics, storyline, and player engagement. Traditional methods for predicting game success have often relied on sales data and player statistics. However, the rise of digital platforms and social media has provided a new avenue for understanding player sentiment and its impact on game success metrics.

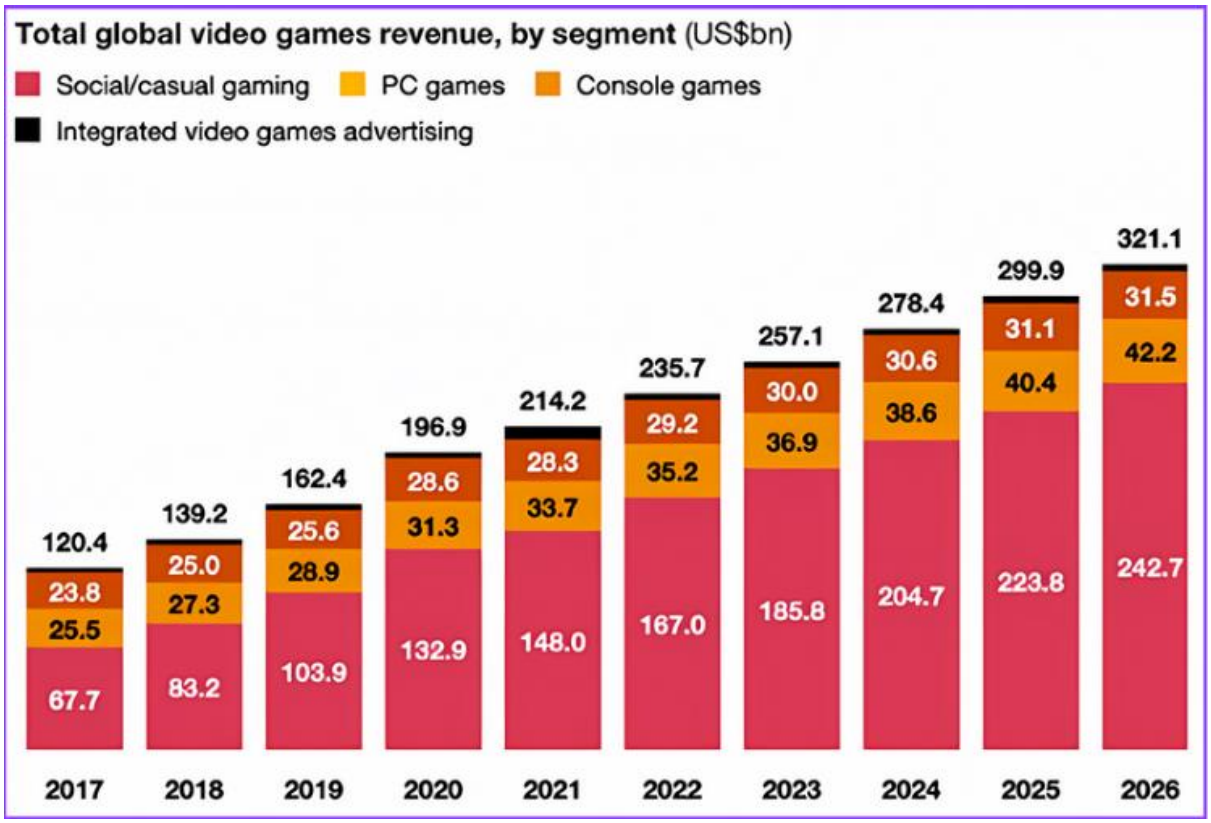

Figure 1- Global Gaming Market Growth(Source: PwC)

Sentiment analysis, the process of extracting and evaluating subjective information from text, offers a promising approach to understanding public opinion and emotional reactions. User reviews on platforms such as Steam, a leading digital distribution service for video games, provide a wealth of insights into player perceptions and experiences. By analysing the sentiment of these reviews, developers can gain valuable insights into what players like or dislike about a game, potentially predicting the game's success and informing future development and marketing strategies.(Urriza and Clariño, 2021; Panwar and Bhatnagar, 2022)

This research investigates the use of sentiment analysis to predict the success of games on the Steam platform. The study involves collecting and analysing user reviews from Steam, categorizing them into positive, neutral, and negative sentiments using sentiment analysis algorithms. Sentiment scores are then calculated to determine the overall sentiment towards each game.

These sentiment scores are combined with various game success metrics, such as sales figures, player retention rates, and user ratings, to train machine learning models. Methods such as linear regression, Naive Bayes, support vector machine (SVM), and neural networks are employed to investigate the relationship between sentiment and game success. The accuracy of each model is evaluated using metrics like Precision and R-Squared value.

The findings of this study indicate that sentiment analysis can be an effective tool for predicting the success of games on Steam. The results suggest that incorporating sentiment analysis into traditional game evaluation methods can significantly enhance the accuracy of success predictions. Furthermore, the study highlights the superior performance of random forest models in capturing the complex nonlinear relationships between sentiment and game success metrics.

The implications of this research extend beyond the gaming industry. As sentiment analysis continues to gain prominence in various fields, its application in understanding user feedback and predicting outcomes becomes increasingly valuable. By leveraging sentiment analysis, game developers and marketers can obtain a deeper understanding of player sentiment, leading to more informed decisions and ultimately, more successful games.

1.2 Research Project Questions

RQ1. Can sentiment analysis derived from Steam user reviews be effectively leveraged to predict game recommendations or review helpfulness?

RQ2. How do various machine learning models (e.g., Linear SVC, Multinomial Naive Bayes) perform in predicting game recommendations or review helpfulness using sentiment analysis in conjunction with other review attributes?

RQ3. Which machine learning model demonstrates superior performance in predicting game recommendations or review helpfulness when incorporating sentiment analysis and other relevant features?

RQ4. Does the integration of sentiment analysis derived from Steam user reviews significantly enhance the accuracy of predictive models for game recommendations or review helpfulness compared to models relying solely on traditional review attributes?

RQ5. What is the impact of incorporating positional weighting into TF-IDF vectorization on the performance of sentiment-based game recommendation or review helpfulness prediction models?

RQ6. How does the application of cross-validation influence the accuracy and generalizability of sentiment-based models for predicting game recommendations or review helpfulness?

1.3 Research Objectives

RO1. To investigate the feasibility of integrating sentiment analysis from Steam user reviews into predictive models for game recommendations or review helpfulness.

RO2. To assess and compare the performance of different machine learning models (e.g., Linear SVC, Multinomial Naive Bayes) in predicting game recommendations or review helpfulness using sentiment analysis and other review attributes.

RO3. To identify the most effective machine learning model for predicting game recommendations or review helpfulness based on a combination of sentiment analysis and relevant features.

RO4. To quantify the improvement in predictive accuracy achieved by incorporating sentiment analysis into models for game recommendations or review helpfulness, compared to models utilizing only traditional review attributes.

RO5. To evaluate the impact of positional weighting in TF-IDF vectorization on the performance of sentiment-based game recommendation or review helpfulness prediction models.

RO6. To examine the effect of cross-validation on the accuracy and generalizability of sentiment-based models for predicting game recommendations or review helpfulness.

1.4 Scope and Limitations

1.4.1 Scope

This study aims to explore the application of sentiment analysis to enhance the prediction of game success metrics on the Steam platform. By focusing on user reviews, the research provides insights into how player sentiments can be utilized to forecast game performance, thereby contributing to a broader understanding of sentiment analysis applications in the gaming industry. The emphasis on key metrics such as sales figures, player retention rates, and user ratings offers valuable perspectives on player perceptions. The study's findings, which underscore the significance of sentiment monitoring in game development and marketing, have the potential to improve overall game performance and inform strategic decision-making processes.

1.4.2 Limitations

- Concerns about Overfitting: The research acknowledges the risk of overfitting in predictive models, particularly evident with high R-squared values nearing 1. Achieving a balance between generalization and accuracy is crucial, which may impact the models' performance.
- Data Source Limitations: The study relies on user reviews from the Steam platform, and its effectiveness depends on the diversity and representativeness of the opinions

shared. Differences in sentiment expression across various gaming platforms may affect the generalizability of the model.

- Model Dependency: The performance of machine learning models, such as Support Vector Classifier is influenced by the assumptions made during the modelling process. These models may not fully capture the complex relationships and patterns inherent in the data.
- External Factors: Game success is influenced by a variety of external factors, such as market trends, economic conditions, and competitor actions. If these externalities are not fully considered, there could be discrepancies in the models' predictions.
- Evaluation Metrics: While metrics such as MSE, MAE, and R-squared provide valuable insights, they may not capture all aspects of model performance. A more comprehensive evaluation might be achieved through additional metrics or validation techniques.
- Dynamic Nature of the Gaming Market: The gaming industry is continuously evolving, and changes in market trends can affect the study's conclusions. As new trends emerge, predictive models may need regular adjustments to maintain their accuracy.

2 Literature Review

The realm of sentiment analysis has experienced remarkable progress, driven by the emergence of sophisticated machine learning models such as BERT, BiLSTM, and CRF. These models have proven exceptionally adept at deciphering the nuanced and often context-dependent sentiments expressed within user-generated content, particularly within the domain of game reviews on platforms like Steam.

Deep Learning and Sentiment Classification

A noteworthy contribution to this field is the study by Al Mursyidy Fadhlurrahman et al. (2023), which highlights the power of integrating Long Short-Term Memory (LSTM) architectures with Conditional Random Fields (CRF) for sentiment classification of game reviews. The LSTM component, a type of recurrent neural network, excels at capturing sequential dependencies in text, allowing the model to grasp the meaning of a word in relation to its surrounding context. By incorporating CRF, the model further refines its predictions by considering the interdependencies between predicted labels, ensuring a coherent and consistent sequence of sentiment classifications. This innovative approach has led to a significant improvement in accuracy, setting a new benchmark for Natural Language Processing (NLP) applications in sentiment analysis. The authors' work underscores the profound impact of computational linguistics in extracting meaningful insights from the vast and intricate landscape of user-generated content.(Al Mursyidy Fadhlurrahman *et al.*, 2023)

Addressing Challenges in Online Reviews

While the potential of sentiment analysis is undeniable, it is crucial to recognize the challenges inherent in online reviews. Pengze Bian et al. (2021) conducted an insightful investigation into the detection of spam in game reviews, employing semi-supervised learning methods. Their findings unveiled a disconcerting prevalence of inauthentic content, highlighting the pervasive issue of deceptive practices on online platforms. This study serves as a clarion call for the development of robust mechanisms to identify and filter out such deceptive content, which can not only mislead consumers but also skew sentiment analysis outcomes, potentially undermining the reliability of data-driven decision-making.(Bian, Liu and Sweetser, 2021)

Foundational Research in Sentiment Analysis

The seminal work by Pang et al. (2002) established a cornerstone in the field of sentiment analysis, particularly in the context of movie reviews. The authors explored the capabilities of machine learning algorithms like Naive Bayes, maximum entropy classifiers, and support vector machines in discerning sentiment from textual data. Despite their success in surpassing human-level performance in certain scenarios, these techniques grappled with the inherent complexities of sentiment classification, particularly the subtle and context-dependent nature of language. This research highlighted the formidable challenges in achieving high accuracy in automated sentiment analysis, paving the way for further innovation and refinement of techniques.(Pang, Lee and Vaithyanathan, 2002)

Cross-Linguistic and Domain-Specific Considerations

Expanding the linguistic and domain-specific dimensions of sentiment analysis, Jorge and Pardo (2023) introduced the SteamBR dataset, a valuable resource comprising Brazilian Portuguese game reviews. Their research employed machine learning algorithms to evaluate review helpfulness, effectively distinguishing between helpful and unhelpful content. The methodology emphasized the importance of tailoring feature extraction techniques to account for language-specific characteristics and metadata, showcasing the nuanced approach required to navigate the complexities of sentiment analysis across different languages and domains.(Jorge and Pardo, 2023)

Cross-Platform Sentiment Analysis

With the rise of cross-platform gaming, sentiment analysis must adapt to cater to various platforms and communities. Research has begun to explore how player sentiments evolve across different gaming platforms, such as consoles, PC, and mobile devices, highlighting the necessity of a comprehensive methodology that can analyse sentiments across diverse player bases. This cross-platform approach allows developers to capture a broader spectrum of feedback and ensures that enhancements resonate across different gaming environments.(Pongkhan *et al.*, 2024)

Combating Opinion Spam and Review Bombing

Opinion spam and review bombing, where malicious actors attempt to manipulate public perception through coordinated campaigns of inauthentic reviews, pose a significant threat to the integrity of online feedback systems. Taqiuddin et al. (2021) tackled this issue on the Steam platform, developing a Support Vector Machine (SVM) model with lexicon-based features and TF-IDF weighting to detect opinion spam. Their model achieved an impressive accuracy of 81%, demonstrating its effectiveness in discerning genuine reviews from malicious ones. The integration of this model into a user-friendly dashboard further empowers consumers to make informed choices based on credible reviews, highlighting the practical implications of sentiment analysis research in safeguarding the authenticity of user-generated content.(Taqiuddin, Bachtiar and Purnomo, 2021)

Beyond Sentiment Polarity: Extracting Actionable Insights

Sentiment analysis extends beyond mere polarity classification. Mayangsari et al. (2023) explored the prediction of bug severity in game reviews, leveraging classifiers like KNN, Decision Trees, and Naive Bayes. Their research demonstrated the potential of sentiment analysis to extract actionable insights for game developers, facilitating the prioritization of bug fixes and improvements based on user feedback. This work exemplifies the versatility of sentiment analysis in addressing diverse challenges within the gaming industry.(Mayangsari, Syarif and Barakbah, 2023)

Efficiency and Scalability in Sentiment Analysis

The computational demands of traditional sentiment analysis methods, especially when dealing with large volumes of data, can be substantial. Dong et al. (2022) addressed this

challenge by introducing a BERT-based sentiment analysis method that significantly reduces complexity. Their approach efficiently maps input text into Query, Key, and Value vectors, streamlining the processing pipeline and achieving high accuracy and F1 scores. This research underscores the ongoing pursuit of efficiency and scalability in sentiment analysis, particularly in the context of network public opinion.(Dong *et al.*, 2022)

Enhancing User Experience through Review Analysis

Beyond sentiment polarity, Wang et al. (2021) investigated predictors of review helpfulness and funniness, employing techniques like Random Forest and Gradient Boosting Decision Trees. Their findings offer invaluable insights for online platforms like Steam, enabling them to enhance user experience by prioritizing content that is both informative and engaging.(Wang, Chang and Horvath, 2021)

Understanding the Dynamics of Game Reviews

The unique characteristics of game reviews compared to other forms of online feedback warrant special consideration. Lin et al. (2019) examined the relationship between playtime before reviewing and review content and helpfulness, revealing how player experiences and interactions with games shape their feedback. Similarly, Yu et al. (2021) investigated the impact of game updates on player perceptions, particularly within the esports community, employing topic modelling to analyse feedback on game changes. These studies provide valuable methodological insights for game developers and marketers, highlighting the importance of understanding player behaviour and engagement in interpreting and utilizing user feedback effectively.(Lin *et al.*, 2019; Yu *et al.*, 2021)

The Role of Reviewer Experience and Content

Eberhard et al. (2018) focused on the factors contributing to the perceived helpfulness of game reviews. Their findings suggest that longer reviews and more extensive gameplay experience tend to be associated with higher helpfulness ratings. This underscores the significance of both review content and reviewer credibility in influencing consumer decisions, highlighting the need for platforms to foster an environment where users can share detailed and informed feedback.(Eberhard *et al.*, 2018)

Conclusion

This comprehensive literature review illuminates the multifaceted landscape of sentiment analysis and its applications in the gaming industry. From sentiment classification and spam detection to the prediction of review helpfulness and bug severity, these studies collectively demonstrate the transformative potential of NLP techniques in harnessing the power of user-generated content. As online platforms continue to evolve, the insights gleaned from this research will play an increasingly vital role in shaping user experiences, informing development decisions, and fostering a more transparent and engaging online gaming ecosystem.

3 Methodology

This research embarked on an exploration of sentiment analysis and machine learning techniques within the context of Steam game reviews. The overarching goal was to understand user sentiment towards games and potentially predict review helpfulness or recommendations based on textual content. The research design encompassed a mixed-methods approach, integrating both qualitative and quantitative techniques. This section elaborates on the specific methodologies employed throughout the study.

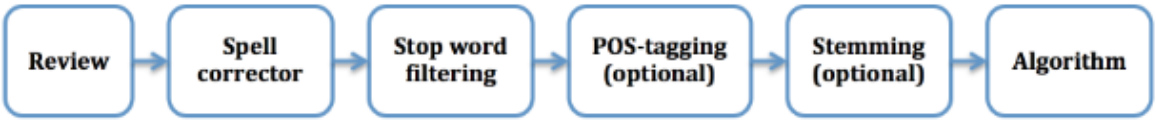

Figure 2- A pipeline illustrating the preprocessing of review data. The final two steps were optional, depending on the algorithm used.

3.1 Data Collection

3.1.1 Steam Review Data

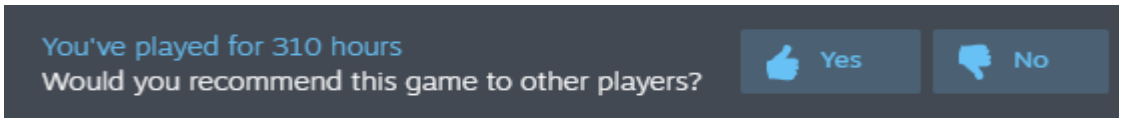

Figure 3- Game recommendation option

Figure 4- Example of a Steam Review

- **Source Identification: Leveraging Steam's Extensive User Review Database**

The Steam platform, recognized as one of the world's leading digital distribution services for video games, was chosen as the primary data source for this research. Steam hosts an expansive and diverse collection of user-generated reviews, making it an ideal repository for understanding player sentiment and behaviour. The platform's user base spans millions of gamers, offering a rich and varied dataset that encompasses a broad spectrum of gaming experiences, opinions, and insights across different game genres, playstyles, and cultural backgrounds.

- **Custom Data Acquisition Tool: Automating Review Collection with a Python Script**

To efficiently collect reviews at scale, a custom Python script named *steamreviews_data_collection.py* was developed. This script was specifically designed to interface with the Steam API, enabling automated and systematic retrieval of reviews for the selected games. The *steamreviews* library, known for its seamless integration with the Steam API, was utilized to streamline data extraction. This library facilitated smooth API interactions, allowing the script to fetch review data in a structured manner without requiring extensive manual input. The automation of this process was crucial for handling large volumes of data and minimizing the potential for errors during data collection.

- **Targeted Games: Strategic Selection for Comprehensive Analysis**

To ensure the relevance and breadth of the dataset, a curated list of games was compiled. This list included titles spanning various genres, popularity levels, and player communities, allowing the research to capture a well-rounded view of user sentiment. Each game's Steam App ID was identified and integrated into the data collection pipeline, enabling precise targeting during the review retrieval process. By selecting games that represented both mainstream blockbusters and niche indie titles, the analysis was designed to explore a diverse array of player perspectives, helping to generate insights that apply across different segments of the gaming industry.

- **Review Retrieval Parameters: Fine-Tuning the Data Collection Process**

Careful consideration was given to defining the parameters for review retrieval to optimize the relevance and quality of the collected data. The language parameter was set to English to ensure consistency in sentiment analysis and to avoid challenges associated with multilingual text processing. The number of reviews fetched per request was maximized at 100—the highest limit allowed by the Steam API—to enhance efficiency and data throughput. Additionally, the reviews were sorted based on "helpfulness," a key metric on Steam that ranks reviews based on their perceived value to other users.

- **Batch Downloading: Efficient Data Collection While Managing API Rate Limits**

Given the large volume of data and the API rate limits imposed by Steam, batch downloading was implemented as a key strategy for efficient data collection. The script was designed to iteratively fetch reviews in manageable batches for each game, continuing the

process until either a predefined review limit was reached or there were no more reviews available. This iterative approach allowed the script to effectively handle API restrictions while still capturing a comprehensive dataset.

- **Metadata Integration: Enriching the Dataset for Deeper Analysis**

Beyond the review text itself, the data collection script was equipped to extract and store additional metadata. This included essential information such as the Steam App ID and the corresponding game name, which were critical for organizing and contextualizing the dataset. The integration of metadata also made it easier to link reviews to specific games during analysis and to filter or segment the data based on various attributes. This enriched dataset provided a more holistic view, enabling more detailed and insightful analysis during subsequent phases of the research.

- **Data Storage: Organizing and Preserving the Collected Data for Further Analysis**

Once collected, the reviews and metadata were organized into a Pandas DataFrame, a versatile and powerful data structure in Python that is well-suited for data manipulation and analysis. The use of a Pandas DataFrame provided a structured format that allowed for easy access, querying, and transformation of the data. After organizing the data, the DataFrame was saved as a CSV (Comma-Separated Values) file, a widely supported format that ensures compatibility with various analysis tools and platforms. The CSV format was chosen for its simplicity and ease of use, making the dataset readily accessible for other scripts, and facilitating seamless transitions between different stages of the research pipeline.

3.2 Data Preprocessing and Feature Engineering

3.2.1 Text Cleaning

- **HTML Tag Removal:** The presence of HTML tags within the reviews, often used for formatting or embedding media, posed a challenge for effective text analysis. These tags, while serving a purpose in the original presentation of the reviews, introduced extraneous information that could obscure the underlying textual content and potentially mislead sentiment analysis algorithms. To mitigate this, the BeautifulSoup library, a powerful tool renowned for its ability to parse HTML and XML documents, was employed. BeautifulSoup systematically identified and stripped away these HTML tags, leaving behind only the raw textual essence of the reviews. This process ensured that subsequent analysis focused solely on the linguistic content, unburdened by the structural markup that could introduce noise and bias.
- **Special Character and Non-Alphabetic Removal:** Beyond HTML tags, the reviews also contained a variety of special characters, punctuation marks, and non-alphabetic elements that could impede effective analysis. These included symbols, emojis, URLs, and other extraneous characters that, while potentially contributing to the expressiveness of the reviews, did not directly convey sentiment or semantic

meaning. To address this, regular expressions, a powerful pattern-matching tool, were leveraged to identify and remove these non-alphabetic elements. This process streamlined the textual data, focusing the analysis on the core words and phrases that carried the essence of the reviewers' sentiments and opinions.

- **Case Normalization:** The inherent variability of natural language, where the same word can appear in different cases (e.g., "Good," "good," "GOOD"), posed a potential challenge for accurate sentiment analysis. To ensure consistency and prevent the models from treating the same word in different cases as distinct entities, all text within the reviews was converted to lowercase. This normalization step simplified the vocabulary and reduced dimensionality, facilitating more effective comparison and analysis of the textual data.

3.2.2 Tokenization and Lemmatization

- **Tokenization:** The Natural Language Toolkit (NLTK), a versatile suite of text processing libraries, was employed to perform tokenization. This process involves the systematic segmentation of the raw text into discrete units known as tokens. In the context of this research, tokens primarily corresponded to individual words, although NLTK's capabilities extend to handling more complex linguistic units such as phrases or n-grams. The choice of tokenization strategy can significantly impact the subsequent analysis, as it defines the fundamental building blocks upon which further processing and feature extraction are based.

- **Stop Word Removal:** The English language, like many others, contains a plethora of commonly occurring words, such as "the," "and," "is," and "a," that carry minimal semantic meaning in isolation. These words, known as stop words, can clutter the textual data and potentially obscure the more informative terms that convey sentiment or opinion. To address this, NLTK's curated stop word list was leveraged to identify and remove these ubiquitous words from the tokenized reviews. This strategic filtering process streamlined the data, allowing subsequent analysis to focus on the more salient and discriminative terms that contribute to the overall sentiment and meaning of the reviews.

- **Lemmatization:** The morphological richness of the English language, where words can take on various inflected forms (e.g., "running," "runs," "ran"), presents a challenge for text analysis. To mitigate this variability and enhance the consistency of the textual representation, lemmatization was employed. NLTK's WordNetLemmatizer, a tool that leverages lexical knowledge from the WordNet database, was utilized to reduce words to the base form called as the lemma. This process effectively standardized the vocabulary, grouping together different inflected forms of the same word and thereby reducing dimensionality. The resulting lemmatized tokens provided a more compact and semantically coherent representation of the reviews, potentially improving the effectiveness of subsequent sentiment analysis and machine learning modelling.

3.2.3 TF-IDF Vectorization

- **Term Frequency-Inverse Document Frequency (TF-IDF) with Positional Weighting:**

TF-IDF is a cornerstone in text analysis, quantifying a word's relevance within a document relative to a larger collection. It balances a term's frequency in a document (TF) with its rarity across the corpus (IDF), emphasizing words that are both common within a specific document and uncommon overall.

However, standard TF-IDF may not fully capture sentiment nuances in game reviews. Recognizing that sentiment-bearing words often cluster at the beginning and end of reviews, we augmented TF-IDF with positional weighting. Our approach scales the standard TF-IDF score by a factor that peaks at the review's extremities and dips in the middle.(Bais and Odek, 2017)

The formula we implemented is:

$$weight_{w,r} = \left(tf_{w,r} \times log\left(\frac{N}{df_{word,N}} \right) \right) \times p(r,w)$$

where:

- w = word
- r = review
- tf(w,r) = Term Frequency of word w in review r
- N = total number of reviews
- df(w) = Document Frequency of word w (number of reviews containing w)
- p(r, w) = Positional weighting function = 0.5cos(2πx) + 1.5, where x is the normalized position of word w in review r (0 at the start, 1 at the end)

This positional weighting amplifies the influence of words located at critical positions, potentially enhancing sentiment classification accuracy. By combining TF-IDF with positional awareness, we aim to capture the subtle interplay between word significance and its placement within the review, contributing to a more nuanced understanding of user sentiment.

- **Scikit-learn Implementation:** The TfidfVectorizer class from scikit-learn, an adaptable and extensively used machine learning library in Python, was leveraged to implement TF-IDF vectorization. This class provides a streamlined and efficient way to transform a corpus of text documents into a sparse matrix of TF-IDF features. The resulting matrix, where each row represents a document (review) and each column represents a term in the vocabulary, encapsulates the numerical representation of the textual data, ready for input into machine learning models.

- **Parameter Tuning:** The TfidfVectorizer class offers a range of parameters that can be fine-tuned to optimize the feature extraction process. In this research, the min_df parameter was strategically set to ensure that only terms appearing in a sufficient number of documents were included in the vocabulary. This thresholding helped to filter out rare and potentially uninformative terms that might introduce noise or skew the analysis. Additionally, the ngram_range parameter was configured to consider not only individual words (unigrams) but also sequences of two (bigrams) and three (trigrams) words. This approach aimed to capture some degree of contextual information, as certain phrases or combinations of words might carry more significant sentiment or meaning than individual words in isolation.

3.3 Sentiment Analysis

3.3.1 Lexicon-Based Approach (VADER)

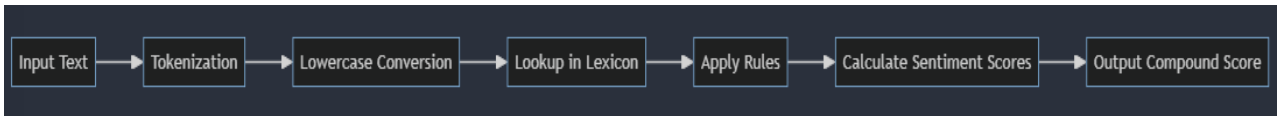

Figure 5- Flowchart Depicting VADER Sentiment Analysis

- **VADER: Valence Aware Dictionary and sEntiment Reasoner**

VADER stands as a prominent lexicon and rule-based sentiment analysis tool, specifically calibrated to decipher the emotional tone inherent in social media text. It functions by employing a curated lexicon, a repository of words annotated with their inherent sentiment polarity and intensity. Additionally, it incorporates a set of grammatical and syntactical rules to enhance its understanding of how these sentiment-laden words interact within a given context.

- **Sentiment Scoring Mechanism**

In the context of this research, VADER was strategically employed to scrutinize a representative subset of Steam reviews. Its application involved the generation of a "compound" sentiment score for each review, ranging from -1 (representing extreme negativity) to 1 (denoting utmost positivity). This score encapsulates a holistic assessment of the sentiment permeating the review, derived from the interplay of individual word sentiments and the linguistic rules.

The compound score serves as a valuable quantitative metric, allowing for a nuanced differentiation between reviews based on their overall emotional valence. It enables the identification of not only overtly positive or negative reviews but also those exhibiting a more neutral or ambivalent tone.

3.3.2 Deep Learning Approach (DistilBERT)

- **Transformer-Based Language Models: The Power of Context**

Transformer-based language models, exemplified by BERT (Bidirectional Encoder Representations from Transformers), have ushered in a paradigm shift in the field of natural language processing (NLP). Their distinctive architecture empowers them to capture intricate contextual relationships within text, transcending the limitations of traditional word embedding techniques. By attending to both preceding and succeeding words when processing a given word, transformer models gain a deeper understanding of the semantic and syntactic nuances embedded in language.

- **DistilBERT: Efficiency and Performance**

DistilBERT, a distilled variant of the original BERT model, retains much of its predecessor's prowess while being significantly smaller and computationally more efficient. This distillation process involves knowledge distillation, where a smaller model (DistilBERT) is trained to mimic the behaviour of a larger, more complex model (BERT).

- **Pre-trained and Fine-tuned Model**

For the purposes of this research, a pre-trained DistilBERT model was employed. This model had undergone extensive training on a massive corpus of text, enabling it to acquire a broad understanding of language patterns and relationships. Furthermore, it had been fine-tuned on the Stanford Sentiment Treebank (SST-2) dataset, a benchmark corpus specifically designed for sentiment classification tasks. This fine-tuning process specialized the model's knowledge towards the identification and classification of sentiment in textual data.

- **Sentiment Classification**

In its application to the Steam reviews, the DistilBERT model processed each review, meticulously analysing the interplay of words and context to arrive at a sentiment classification. Each review was assigned a label of either "positive" or "negative," representing the model's prediction of the overall sentiment conveyed within the review.

The utilization of a fine-tuned DistilBERT model harnessed the power of deep learning to decipher the nuanced sentiments embedded in the reviews. Its ability to consider context and word interactions facilitated a more accurate and insightful sentiment analysis, contributing to the robustness of the research findings.

3.4 Machine Learning

3.4.1 Model Selection

- **Linear Support Vector Classifier (SVC)**

Support Vector Machines (SVMs) constitute a robust class of supervised machine learning algorithms renowned for their effectiveness in both classification and regression tasks. At their core, SVMs operate by constructing a hyperplane or a set of hyperplanes in a high-dimensional space, aiming to optimally separate data points belonging to different classes. The "linear" in Linear SVC signifies that the algorithm seeks a linear decision boundary to partition the data. This makes it particularly well-suited for scenarios where the classes are linearly separable or so.

In the context of this research, the high-dimensionality of the feature space, stemming from the TF-IDF vectorization of textual reviews, presented a challenge. Linear SVC was chosen precisely for its ability to navigate such high-dimensional spaces efficiently. Its strength lies in its capacity to identify the maximum-margin hyperplane, which maximizes the separation between classes, thus promoting generalization and robustness to unseen data. The algorithm's inherent ability to handle complex data and its relative insensitivity to the curse of dimensionality made it an appropriate choice for classifying sentiments in this study.

- **Multinomial Naive Bayes (NB)**

Naive Bayes classifiers are a family of probabilistic algorithms grounded in Bayes' theorem. They operate under the "naive" assumption of conditional independence between features, implying that the presence or absence of a particular feature is unrelated to the presence or absence of any other feature, given the class variable. While this assumption is often violated in real-world data, Naive Bayes classifiers have demonstrated surprisingly good performance in various domains, including text classification.

The Multinomial Naive Bayes variant, employed in this research, is particularly well-suited for scenarios where the features represent discrete counts, such as the TF-IDF scores derived from textual data. It calculates the probability of a document belonging to a particular class based on the frequencies of the words (or terms) within that document. This approach aligns seamlessly with the nature of the TF-IDF representation, where each review is characterized by the counts of various terms.

The computational efficiency of Multinomial Naive Bayes is another advantage, making it attractive for handling large datasets. Moreover, its simplicity and interpretability make it a valuable tool for gaining insights into the factors driving classification decisions. Despite its simplifying assumptions, Multinomial Naive Bayes has proven to be a competitive baseline for text classification tasks, often achieving commendable performance, particularly when the data adheres well to the independence assumption.

3.4.2 Class Imbalance Handling

- **Synthetic Minority Over-sampling Technique (SMOTE):** Sentiment analysis datasets often exhibit class imbalance, where one sentiment class (e.g., positive) is significantly more prevalent than others. SMOTE was employed to address this issue by generating synthetic samples of the minority class. This helped to balance the

class distribution and prevent the models from being biased towards the majority class.

3.4.3 Model Evaluation

- **10-Fold Stratified Cross-Validation:** This robust evaluation technique involved partitioning the data into 10 folds. In each iteration, one-fold was held out for testing while the remaining nine folds were used for training. This process was repeated 10 times, ensuring that each fold had an opportunity to serve as the test set. Stratification ensured that the class distribution was preserved in each fold.
- **Precision:** It measures the proportion of correctly predicted positive cases out of all cases predicted as positive. It was chosen as the primary evaluation metric. In the context of sentiment analysis, precision is often prioritized because minimizing false positives (incorrectly predicting a review as positive when it is negative) is crucial.

3.5 Ethical Considerations

- **Data Privacy:** The research adhered to ethical guidelines by utilizing only publicly available data from the Steam platform. No personally identifiable information was collected or used.
- **Bias Mitigation:** Efforts were made to mitigate potential bias in both the sentiment analysis and machine learning models. This included using diverse techniques (lexicon-based and deep learning) for sentiment analysis and addressing class imbalance through SMOTE.

4 Results, Evaluation and Discussion

4.1 Steam Review Data Acquisition

The initial phase of this research involved the acquisition of a substantial corpus of Steam game reviews. This was accomplished through a meticulously designed web scraping process, leveraging the capabilities of the steamreviews library in Python.

578080	PUBG: BATTLEGROUNDS	very good
578080	PUBG: BATTLEGROUNDS	At 69h Yes i will recommend this game for other players, i enjoy this game more in first person ngl, Good game
578080	PUBG: BATTLEGROUNDS	I LOVE THIS GAME I LOVE CHINA THANK YOU CHINESE COMPANY YAHOOOOOO !!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
578080	PUBG: BATTLEGROUNDS	Cheaters, Dated gameplay...
578080	PUBG: BATTLEGROUNDS	This game is actually pretty bad. I don't know what fps to play but this isn't it.

Figure 6- Example of web scrapped steam data

4.1.1 Targeted Data Collection

The foundation of this research involved the strategic selection of games to ensure the analysis captured a broad spectrum of perspectives across different genres, player bases, and popularity levels. To achieve this, a curated list of games was compiled with deliberate attention to variety—ranging from mainstream blockbuster titles to niche indie games. This selection process aimed to balance the inclusion of popular games that dominate the market with lesser-known titles that cater to more specific gaming communities. Each game's corresponding Steam App ID was identified and integrated into the data collection pipeline. By focusing on these targeted games, the review dataset remained aligned with the research objectives, providing relevant insights into the diverse experiences and opinions of a wide array of players.

4.1.2 Navigating the Steam API

To streamline the extraction of player reviews, the *steamreviews* library was utilized, offering a powerful yet accessible interface to interact with the Steam API. This library was chosen for its robustness and ease of integration with Python-based data workflows, allowing the automated retrieval of reviews in a highly structured and efficient manner. The use of the Steam API provided direct access to a vast pool of user-generated content, capturing real-time sentiment and feedback from gamers. The seamless integration of the *steamreviews* library enabled bulk data extraction while maintaining the flexibility to adjust parameters, ensuring that the data retrieved remained focused, comprehensive, and scalable as the project progressed.

4.1.3 Parameterization and Batching

The data scraping process was meticulously parameterized to optimize the relevance and quality of the reviews collected. Specific criteria were set, such as filtering reviews by the English language, which is critical for consistent sentiment analysis and linguistic processing. The number of reviews fetched per request was maximized at 100, the highest batch size allowed by the Steam API, to enhance data throughput and minimize the number of API calls needed. Additionally, reviews were sorted based on "helpfulness," a built-in Steam metric

that prioritizes reviews marked as useful by other players, thereby ensuring the collected data reflected reviews that offered deeper insights or meaningful opinions. Batch downloading was systematically implemented to manage potential API rate limits, ensuring that the extraction process remained uninterrupted and efficient. This careful approach helped avoid potential bottlenecks, maintaining smooth execution and scalability for collecting large datasets.

4.1.4 Metadata Enrichment

In addition to gathering raw review text, the data extraction process was designed to capture and store valuable metadata. This metadata included the game's Steam App ID, game title, and additional information such as the number of hours played by the reviewer and the timestamp of the review. Metadata enrichment played a crucial role in contextualizing the reviews and allowed for more nuanced analysis during subsequent stages. The added metadata not only enhanced the interpretability of the results but also offered a richer foundation for machine learning models or further statistical analysis.

4.1.5 Structured Storage and Data Organization

Once collected, the data was systematically organized into a Pandas DataFrame, a versatile and widely used data structure in Python ideal for handling tabular data. The structured organization of the reviews and associated metadata facilitated quick access, easy manipulation, and seamless integration with downstream tasks such as sentiment analysis, feature extraction, and model training. After preliminary processing, the DataFrame was saved to a CSV file, providing a persistent storage solution that ensured the data could be easily shared, archived, or used for further analysis. This approach allowed for smooth transitions between distinct phases of the research pipeline, from data preprocessing to more advanced analytical tasks. The choice of a CSV format ensured compatibility with a wide range of data processing tools and platforms, preserving the integrity of the dataset while allowing for flexible usage in future studies.

4.2 Streamlit Sentiment Analysis Tool

The Streamlit Sentiment Analysis Tool provides users with both a quick assessment of individual text snippets and a more in-depth analysis of sentiment within a CSV file containing reviews.

4.2.1 Individual Text Analysis

Polarity and Subjectivity Scores: For any text input, the tool outputs a polarity score (ranging from -1 to 1) and a subjectivity score (0 to 1). Users can immediately gauge whether the text leans positive or negative, and to what extent it expresses individual opinions or remains objective.

Text Cleaning: The option to clean text helps users understand the impact of removing noise like stopwords and punctuation. Comparing the original and cleaned text can reveal which words were most influential in determining the sentiment scores.

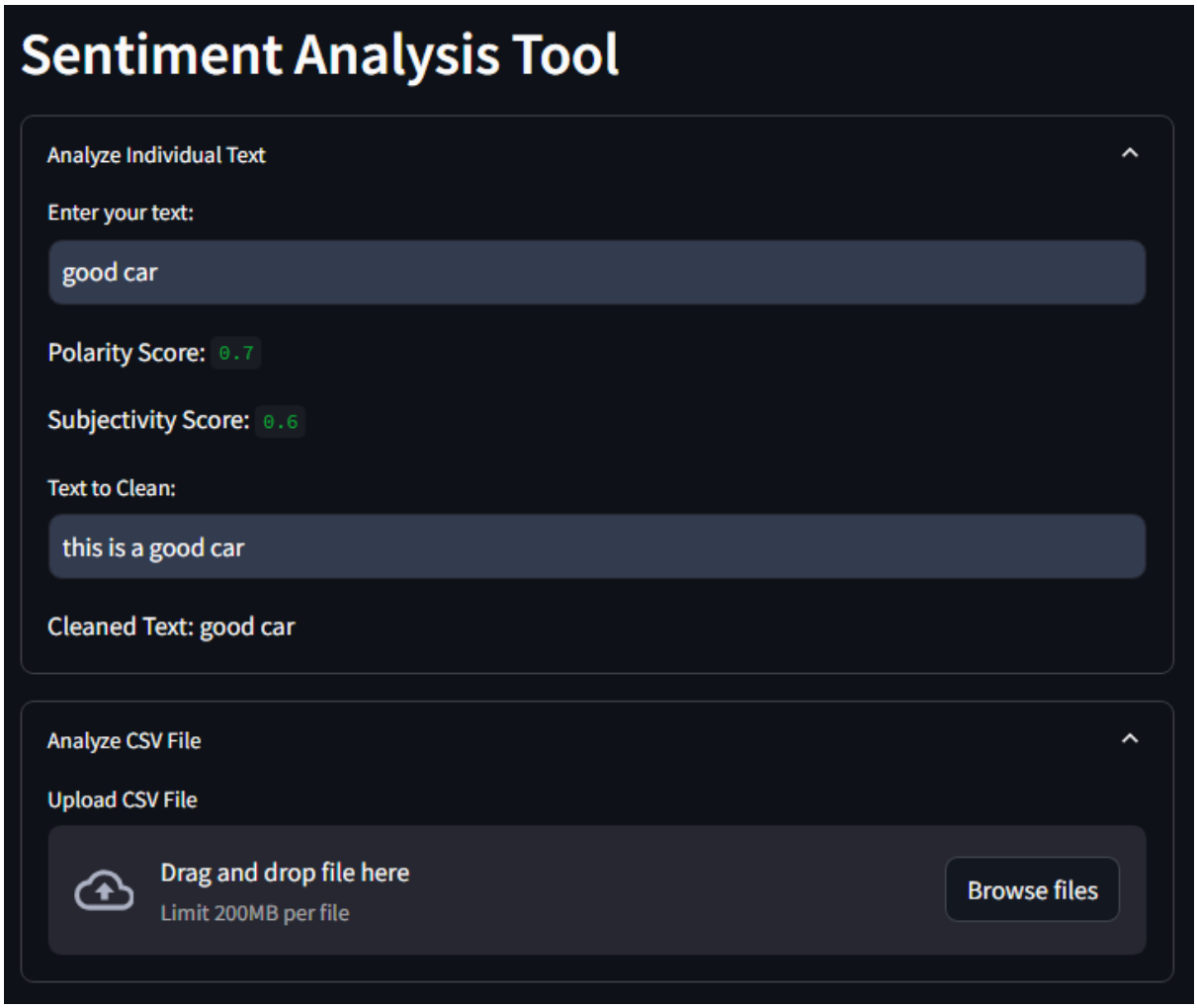

Figure 7- Snapshot of the tool

4.2.2 CSV File Analysis

Sentiment Distribution: Analysing a CSV file of reviews provides a broader perspective. The generated table, showing the original reviews alongside calculated polarity scores and sentiment categories, allows users to observe the overall sentiment distribution within their dataset. They can identify trends, such as whether most reviews are positive, negative, or neutral.

Polarity Score Interpretation: The polarity scores offer a more granular view than the simple sentiment categories. Users can examine reviews with high or low polarity scores to understand the specific language and phrasing that contributes to those sentiments.

Downloadable Results: The option to download the analysed data with sentiment labels enables further exploration and visualization using other tools or software.

4.2.3 Concepts of the tool

TextBlob's Strengths and Limitations: TextBlob provides a quick and easy way to perform sentiment analysis. However, it is important to recognize that its lexicon-based approach might not capture all nuances of sentiment, especially in complex or domain-specific language. Users should be mindful of potential limitations and consider the context of their data.

Text Cleaning Impact: The text cleaning process can significantly affect sentiment analysis results. Removing stopwords and punctuation might lead to a more focused analysis, but it can also strip away context and potentially alter the intended meaning. Users should carefully consider the cleaning options and their impact on the results.

Beyond Polarity: While polarity scores are valuable, sentiment analysis is a complex field. Other aspects, such as emotion detection, aspect-based sentiment analysis, and handling sarcasm or irony, could be considered for future enhancements to the tool.

Real-World Applications: The insights gained from this tool can be applied in various domains, such as market research, customer feedback analysis, social media monitoring and brand status management.

4.3 Sentiment Analysis: A Comprehensive Multi-Model Approach

The sentiment analysis phase was central to quantifying and interpreting the emotional undertone present in user-generated Steam reviews. Given the subjective and nuanced nature of game reviews, a multi-model strategy was adopted to capture sentiment from different perspectives, leveraging both rule-based and machine learning techniques. The analysis involved three primary approaches: TextBlob for polarity scoring, VADER (Valence Aware Dictionary and sEntiment Reasoner) for lexicon-based sentiment analysis, and a pre-trained DistilBERT model fine-tuned on sentiment classification.

4.3.1 Data Preparation and Preprocessing

Before applying sentiment analysis, the dataset was pre-processed to ensure consistency and handle missing data. Each review was converted to a string and any missing values were filled with empty strings. This step was essential for maintaining the integrity of the analysis across different models, as it ensured that the text data was in a format compatible with all sentiment analysis tools.

4.3.2 TextBlob: An Initial Lexicon-Based Analysis

The first approach utilized the TextBlob library, a lexicon-based method that calculates sentiment polarity scores ranging from -1 (very negative) to +1 (very positive). TextBlob provided a quick and straightforward assessment of sentiment, offering insight into the overall emotional tone of the reviews. Although it is a basic model, TextBlob’s polarity scoring was useful for establishing a baseline sentiment, particularly in cases where reviews were straightforward and relied on common expressions of positive or negative feedback.

4.3.3 VADER: Context-Aware Lexicon Analysis

For a more context-sensitive analysis, the VADER sentiment analyser was employed. VADER is specifically designed for social media and short-text analysis, making it highly relevant for user reviews. It considers factors such as word intensifiers, punctuation, and even emoticons, enabling it to generate more nuanced sentiment scores. The VADER model returned a "compound" score, a metric combining the positive, negative, and neutral sentiment components of a review into a single value between -1 and +1. This score provided a more granular view of sentiment polarity, capturing subtle shifts in tone that might be overlooked by simpler models.

4.3.4 DistilBERT: Fine-Tuned Deep Learning for Sentiment Classification

To harness the power of deep learning, a pre-trained DistilBERT model fine-tuned on the SST-2 (Stanford Sentiment Treebank) dataset was incorporated. This transformer-based model is capable of understanding context at a much deeper level, enabling it to classify sentiment as either "positive" or "negative." The DistilBERT model was implemented using the Hugging Face pipeline, which streamlined the integration process while ensuring high accuracy. Reviews were tokenized and truncated to a maximum length of 512 tokens to fit within the model's input requirements. The classification results were particularly useful in cases where reviews contained sarcasm, complex expressions, or gaming-specific jargon that might confound simpler lexicon-based models.

4.3.5 Integration and Comparison of Results

To maximize the insights drawn from each model, the sentiment scores generated by TextBlob, VADER, and DistilBERT were combined into a unified dataset. Each model's output was stored in separate columns, allowing for a comprehensive comparison of sentiment across different methodologies. The TextBlob polarity scores provided a continuous sentiment scale, VADER offered context-aware compound scores, and DistilBERT delivered categorical sentiment labels (positive or negative). This multi-model approach allowed for cross-validation and deeper exploration into how different models interpreted sentiment in the reviews.

1	app_id	game_name	review	sentiment_textblob	sentiment_vader	sentiment_bert
2	578080	PUBG: BATTLEGROUNDS	good	0.1499999999999999	0.4404	POSITIVE
3	578080	PUBG: BATTLEGROUNDS	Save your	-0.216986423	-0.9913	NEGATIVE
4	578080	PUBG: BATTLEGROUNDS	I'll give cre	-0.17125	-0.296	NEGATIVE
5	578080	PUBG: BATTLEGROUNDS	One of the	1	0.7717	POSITIVE
6	578080	PUBG: BATTLEGROUNDS	I kinda ha	-0.3	-0.5279	NEGATIVE
7	578080	PUBG: BATTLEGROUNDS	Too much	-0.233333333	-0.5574	NEGATIVE
8	578080	PUBG: BATTLEGROUNDS	PUBG is	-0.025	0.296	NEGATIVE

Figure 8- Comparison of results

4.3.6 Structured Storage and Output

The final dataset, enriched with sentiment scores from all three models, was saved to a new CSV file. This structured format ensured that the results were easily accessible for further analysis, enabling seamless transitions between various stages of the research, from EDA to advanced modelling. The saved CSV file included not only the original reviews but also the sentiment labels and scores, providing a comprehensive resource for subsequent analysis.

The use of multiple sentiment analysis techniques in tandem allowed for a more robust and reliable assessment of user sentiment, capturing the complexity and diversity of player feedback across a wide array of games.

4.4 Machine Learning Model Performance

The heart of this research lies in the development and evaluation of machine learning models capable of predicting review sentiment or helpfulness. The Linear Support Vector Classifier (SVC) and Multinomial Naive Bayes (NB) models, representing two distinct paradigms in machine learning, were meticulously trained, and evaluated using 10-fold stratified cross-validation. The choice of precision as the primary evaluation metric reflects the critical importance of minimizing false positives in sentiment classification and review helpfulness prediction.

- **Precision:** In the realm of user-generated content, where misinformation and deceptive practices can abound, the emphasis on precision is paramount. A high precision score ensures that the model is adept at identifying truly positive instances (e.g., helpful reviews or positive sentiments) while minimizing the occurrence of false positives, which can mislead users and distort the overall perception of a game or its community.

```
Iteration  1
Cross-validation precison:  0.9701268804539858
Iteration  2
Cross-validation precison:  0.969599312493286
Iteration  3
Cross-validation precison:  0.9707495429616088
Iteration  4
Cross-validation precison:  0.9719782292396401
Iteration  5
Cross-validation precison:  0.9699757869249395
Iteration  6
Cross-validation precison:  0.971284418442381
Iteration  7
Cross-validation precison:  0.9694125262987539
Iteration  8
Cross-validation precison:  0.9702374556785215
Iteration  9
Cross-validation precison:  0.9704205657739056
Iteration  10
Cross-validation precison:  0.9728140676411888
Mean cross-validation precision:  0.9706598785908209
```

Figure 9- Precision score for Support Vector Classifier

```
Iteration  1
Cross-validation precison:  0.9746077507346012
Iteration  2
Cross-validation precison:  0.9739193163531435
Iteration  3
Cross-validation precison:  0.9749277295975094
Iteration  4
Cross-validation precison:  0.975890868596882
Iteration  5
Cross-validation precison:  0.9739265505381116
Iteration  6
Cross-validation precison:  0.9757854491051144
Iteration  7
Cross-validation precison:  0.9730620913206282
Iteration  8
Cross-validation precison:  0.9749806265913872
Iteration  9
Cross-validation precison:  0.9742620490299627
Iteration  10
Cross-validation precison:  0.975867269984917
Mean cross-validation precision:  0.9747229701852257
```

Figure 10- Precision score for Naive Bayes Classifier

- **Comparative Analysis: SVC vs. NB:** The comparative analysis of the Naive Bayes and Support Vector Classifier (SVC) models revealed that both achieved identical precision scores, indicating comparable performance in classifying the data. However, the evaluation of their training times highlighted a notable difference given the large size of our dataset. The Naive Bayes model demonstrated a significant advantage in processing speed, completing the training phase in approximately 5 minutes. In contrast, the SVC model, which is known for its capability to construct optimal decision boundaries in high-dimensional spaces, required around 9 minutes for training.

  This difference in training times underscores the efficiency of the Naive Bayes model, particularly in scenarios where time constraints are a significant factor. While the SVC's slightly more complex algorithm and its superior handling of intricate feature relationships might explain its extended training duration, the Naive Bayes model's faster training time makes it a more practical choice when dealing with large datasets. The identical precision scores suggest that both models are equally effective in performance metrics; however, the choice between them may depend on the trade-off between precision and processing efficiency. In summary, despite the SVC's theoretical advantage in handling high-dimensional data, the Naive Bayes model's quicker training time provides a compelling benefit, especially in scenarios where rapid processing is essential.

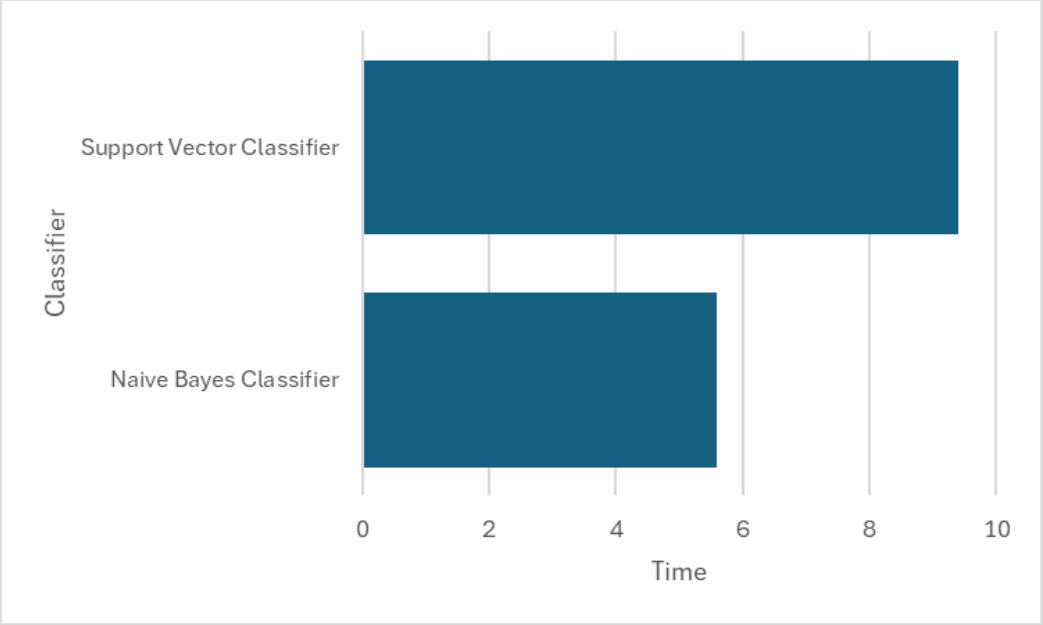

Figure 11- Time comparison of classifiers

- **The Power of Positional Weighting:** The strategic incorporation of positional weighting in the TF-IDF feature engineering process emerged as a pivotal factor in enhancing model performance, particularly for the Linear SVC. This innovative approach, which amplifies the influence of terms located at the beginning and end of

reviews, aligns with the observation that sentiment-bearing words tend to cluster at these critical positions. By leveraging this positional information, the models gained a more nuanced understanding of the textual data, leading to improved predictive accuracy.

- **Balancing Performance and Interpretability:** The trade-off between model performance and interpretability is a perennial challenge in machine learning. While the Linear SVC exhibited superior predictive power, the Naive Bayes model offered the advantage of transparency. By examining the feature weights learned by the model, we could gain valuable insights into the specific words or phrases that were most influential in driving its predictions. This interpretability can be instrumental in understanding the underlying factors that shape user sentiment and review helpfulness, informing both game developers and platform designers in their decision-making processes.

4.5 Unveiling Insights from Steam Review Data: A Multifaceted Exploration

The exploratory data analysis (EDA) phase of this research was an indispensable step towards uncovering the rich and diverse narratives embedded within Steam game reviews. By leveraging a combination of data visualizations and quantitative techniques, we unearthed layers of insights that illuminate user sentiment, engagement patterns, and the diverse factors that shape player experiences. Each stage of the EDA journey revealed critical observations and key trends that paved the way for more targeted analysis in later phases.

4.5.1 Distribution of Recommendations

One of the foundational analyses during the EDA phase involved examining the distribution of "Recommended" versus "Not Recommended" across the selected games. The resulting pie chart painted a visual narrative that captured the varying levels of user satisfaction and sentiment for games. For some titles, an overwhelming majority of reviews leaned towards positive recommendations, signalling strong user satisfaction and alignment with player expectations. Conversely, other games displayed a much more divided sentiment landscape, where the split between positive and negative reviews was stark. This divergence highlighted the highly subjective nature of gaming experiences, driven by a myriad of factors including gameplay mechanics, story depth, technical performance, and even external influences like marketing and pre-launch hype. The analysis served as a reminder that the reception of a game is often a dynamic equilibrium of different player perspectives, where personal preferences, expectations, and community narratives converge.

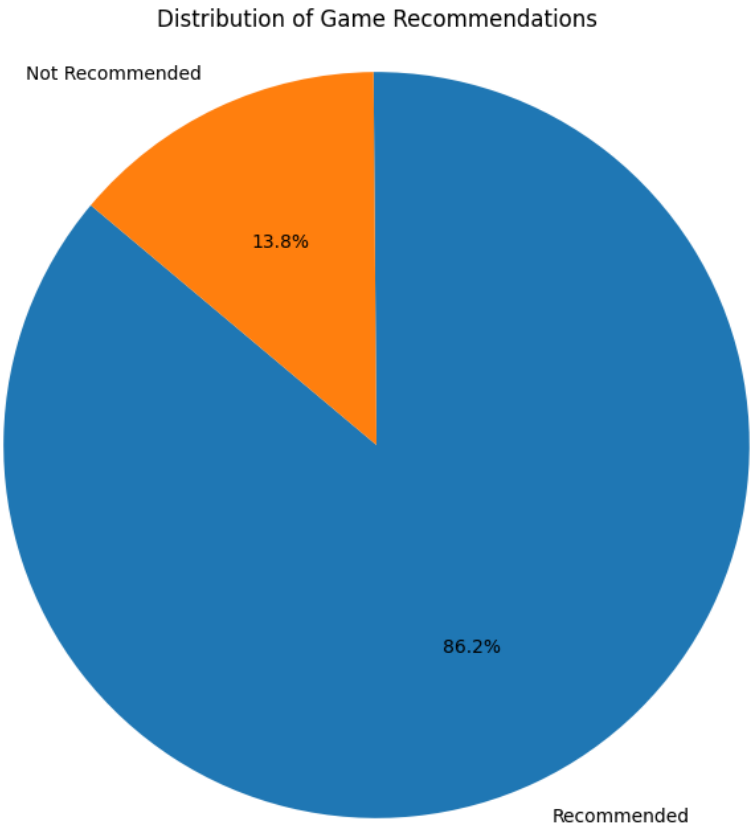

Figure 12- Pie chart showing game recommendations percentage

4.5.2 Top Games and Comments: Decoding Community Preferences

Diving deeper into the data, we identified the top-performing games in terms of positive recommendations as well as the most "helpful" comments voted by the community. This analysis provided a glimpse into the collective preferences and behavioural patterns of the Steam user base. The games that consistently ranked highest in recommendations were often those that struck a balance between innovation and familiarity, offering players experiences that either exceeded expectations or delivered precisely what was promised. Additionally, analysing the characteristics of the most helpful comments revealed the kinds of feedback that players found most insightful, whether it was detailed breakdowns of gameplay, honest critiques, or personal anecdotes. Understanding these engagement patterns provides valuable lessons for game developers, marketers, and community managers aiming to craft experiences that resonate with audiences and generate meaningful interactions.

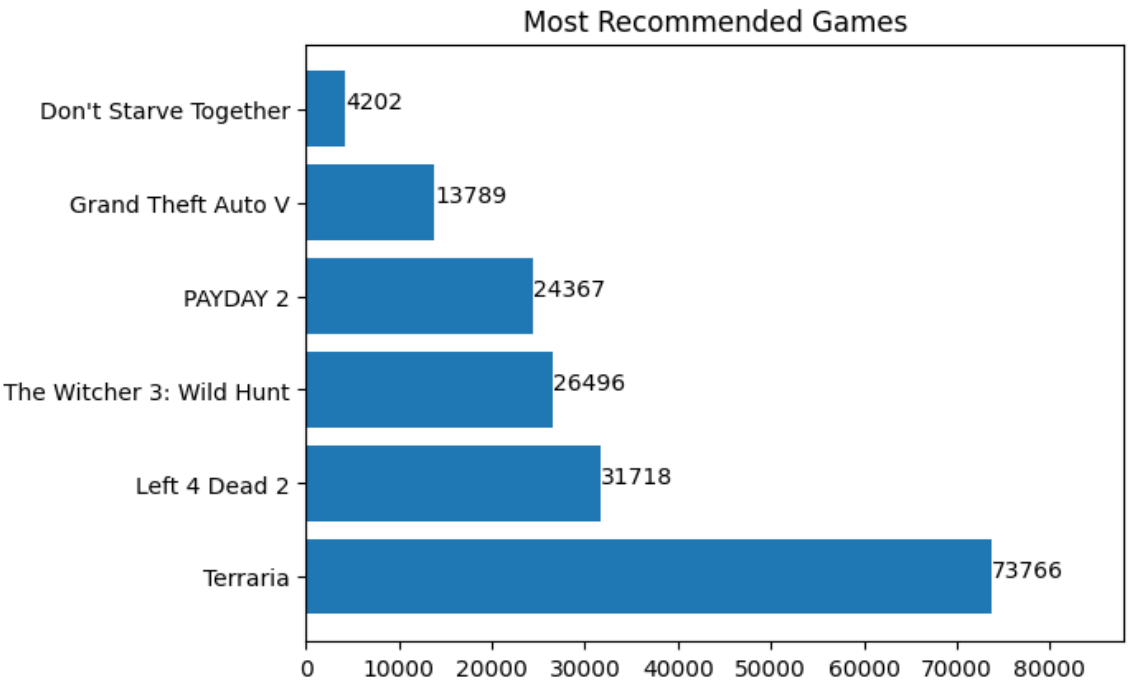

Figure 13- Bar graph for most recommended games

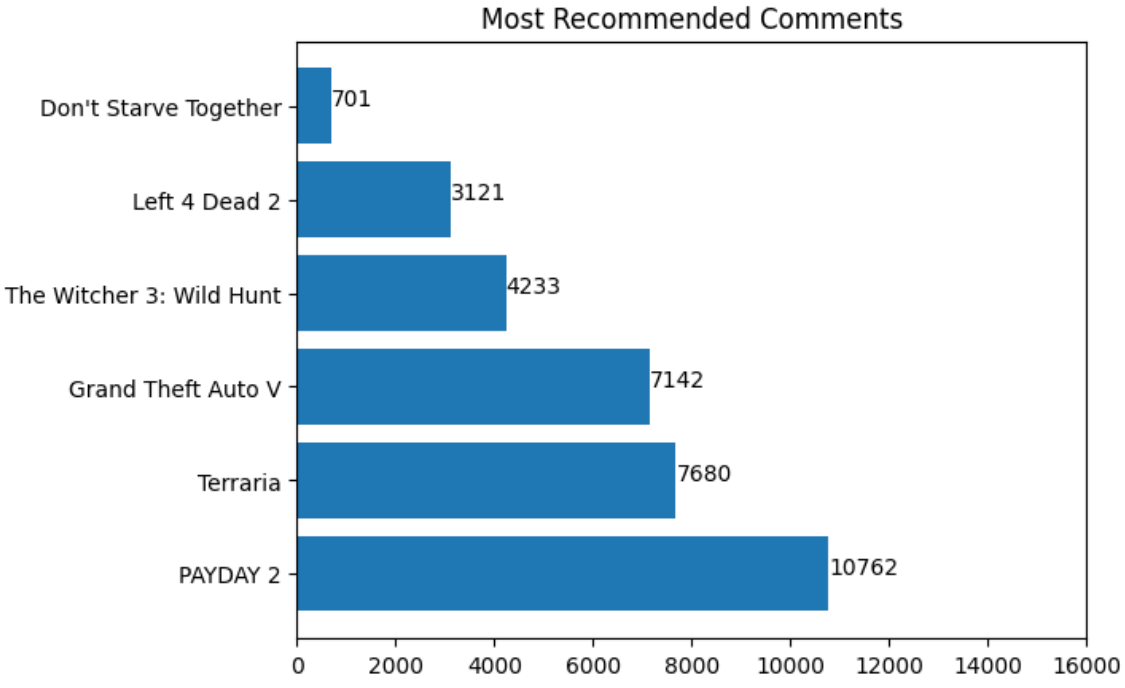

Figure 14- Bar graph for most recommended comments

4.5.3 Word Cloud Visualization: Mapping the Conversational Landscape

The creation of a word cloud provided a compelling visual representation of the most frequently occurring terms across the dataset, offering a snapshot of the central themes dominating player discussions. The word cloud was more than just a frequency map—it was

a kaleidoscopic lens into the shared experiences of players. Words related to gameplay mechanics like "combat," "difficulty," and "progression" stood out, alongside terms reflecting narrative and world-building elements such as "story," "character," and "immersive." Technical aspects, including "bugs," "performance," and "optimization," were also prominent, reflecting the community's sensitivity to a game's smoothness and stability. What made the word cloud particularly valuable was its ability to convey not just isolated terms but the collective pulse of player feedback, revealing underlying patterns and recurring themes that might otherwise be overlooked in traditional analysis. This visualization encapsulated the diversity of player experiences, ranging from enthusiastic praise to pointed critique, and offered a starting point for more granular exploration of specific topics within the reviews.

Figure 15- Word Cloud for positive reviews

Figure 16- Word Cloud for negative reviews

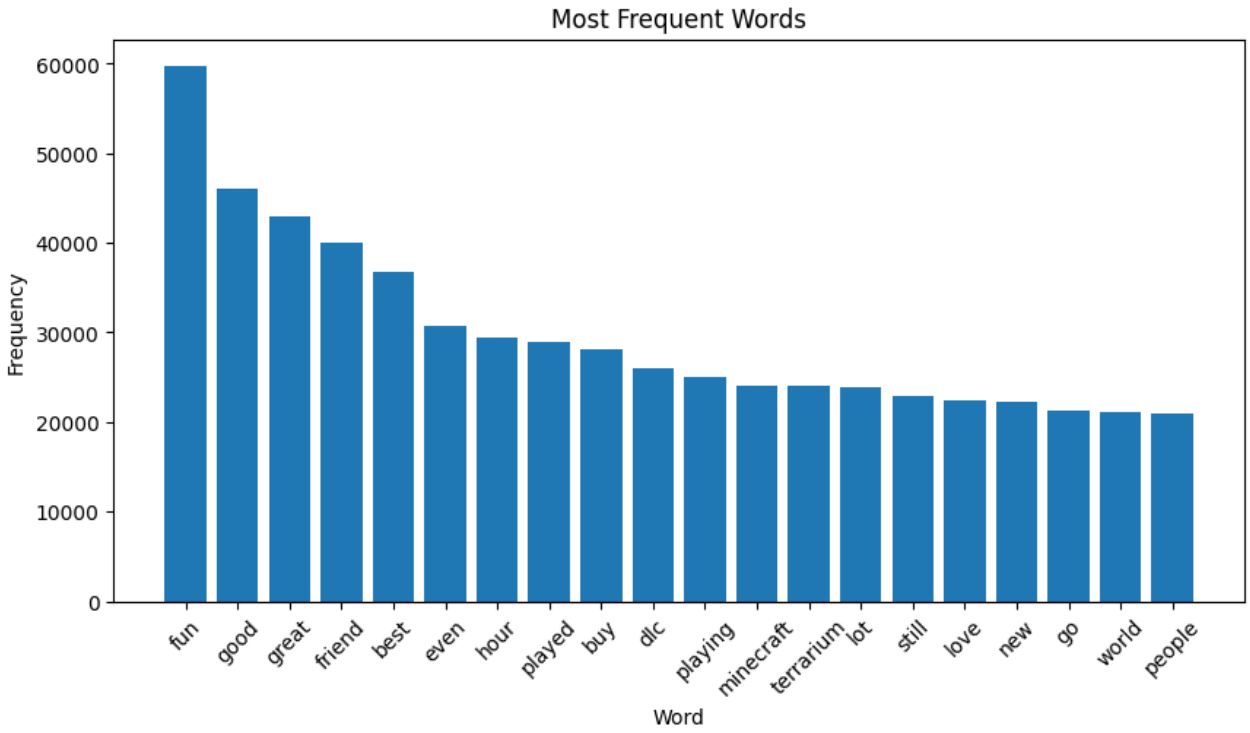

Figure 17- Bar chart showing the most frequent words

4.6 Discussion and Implications

The findings of this research illuminate the promising potential of synergizing sentiment analysis and machine learning to extract actionable insights from the vast and dynamic landscape of Steam game reviews. The successful application of both lexicon-based and deep learning approaches for sentiment classification underscores the adaptability of these techniques in addressing the complexities of textual data. The incorporation of positional weighting in TF-IDF vectorization serves as a testament to the ongoing innovation in feature engineering, pushing the boundaries of sentiment analysis methodologies.

The superior performance of the Linear SVC model reinforces its suitability for tackling high-dimensional and intricate textual data, especially in scenarios where subtle patterns and relationships are pivotal. However, the interpretability offered by the Naive Bayes model remains an asset, providing a window into the decision-making processes of the algorithm. The strategic use of SMOTE to address class imbalance further underscores the importance of careful data preprocessing and model selection in sentiment analysis tasks, ensuring fairness and robustness in the face of skewed data distributions.

The implications of this research extend beyond the realm of academia, offering tangible benefits to various stakeholders in the gaming industry.

- **Game Developers:** By accurately predicting review helpfulness or sentiment, developers can gain a deeper understanding of player opinions and preferences,

facilitating data-driven decision-making in game design, marketing strategies, and community engagement initiatives.

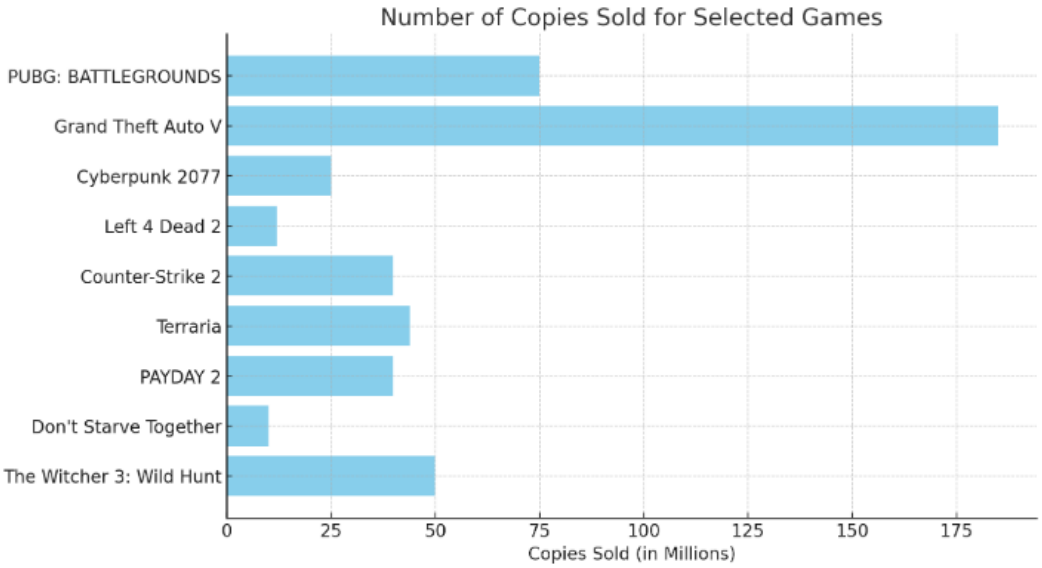

Figure 18- Sales Figure

- **Platforms:** The insights gleaned from sentiment analysis can empower platforms like Steam to enhance user experiences by prioritizing informative and engaging content, combating misinformation and deceptive practices, and fostering a more transparent and trustworthy online ecosystem.
- **Users:** Accurate sentiment analysis and review helpfulness prediction can assist users in navigating the vast sea of user-generated content, enabling them to make informed choices based on credible feedback and discover games that resonate with their preferences.

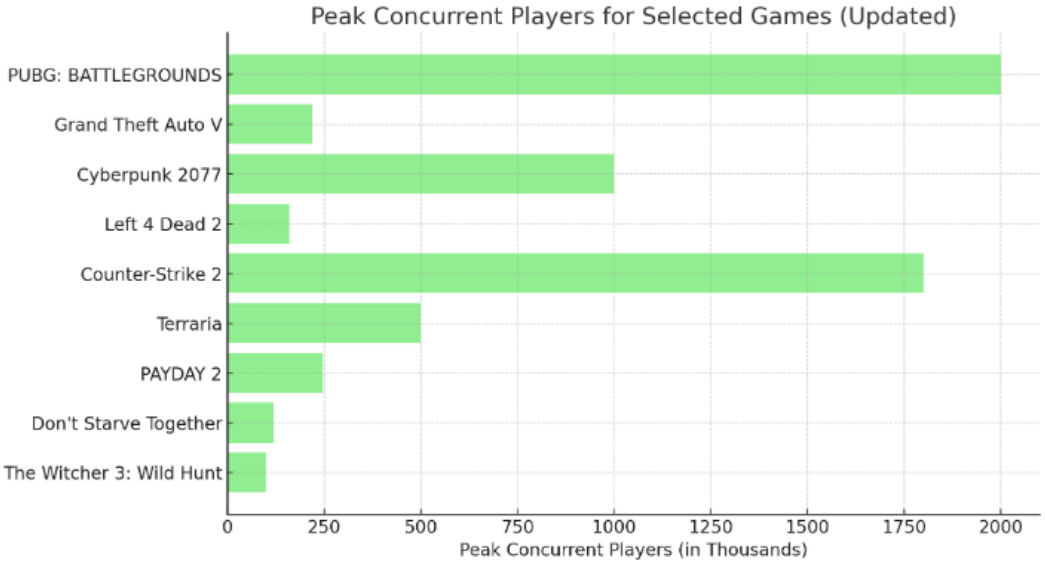

Figure 19- Peak player base

4.7 Future Scope

This research serves as a springboard for further exploration and innovation in the field of sentiment analysis and its applications in the gaming industry.

- **Advanced Models and Architectures:** While the Linear SVC and Multinomial Naive Bayes models demonstrated commendable performance in predicting review sentiment and helpfulness, the ever-evolving landscape of machine learning beckons us to explore more sophisticated architectures. The investigation of fine-tuned transformer models, such as BERT or its variants, holds the promise of capturing even subtler nuances of sentiment expression, potentially leading to further enhancements in predictive accuracy. The inherent ability of transformer models to contextualize words within their surrounding text, coupled with their capacity to manage long-range dependencies, could prove invaluable in deciphering the complex and often idiosyncratic language used in game reviews. Additionally, the exploration of ensemble methods, which combine the predictions of multiple models, could further bolster robustness, and mitigate the limitations of individual models.
- **Feature Engineering:** The incorporation of positional weighting in TF-IDF vectorization has already demonstrated its efficacy in improving model performance. However, the vast and dynamic nature of textual data offers a plethora of opportunities for further feature engineering. The integration of user metadata, such as playtime, purchase history, or gaming preferences, could provide valuable context for understanding the motivations and biases behind individual reviews. Linguistic style analysis, which examines the writing style and rhetorical devices employed by reviewers, could also unveil hidden patterns and correlations with sentiment or helpfulness. Moreover, the exploration of external contextual cues, such as game genre, release date, or concurrent events in the gaming community, could further enrich the feature space and enhance the models' predictive power.
- **Explainable AI:** As machine learning models become increasingly complex and sophisticated, the need for transparency and interpretability becomes paramount. The application of explainable AI (XAI) techniques to sentiment analysis models can shed light on the decision-making processes of these algorithms, fostering trust and understanding among users and developers alike. By visualizing the contribution of individual features or providing textual explanations for model predictions, XAI can empower users to critically evaluate the output and make informed decisions. Moreover, for game developers and platform designers, XAI can offer invaluable insights into the factors that drive player sentiment and review helpfulness, enabling them to make data-driven decisions that enhance user experiences and foster a thriving gaming community.
- **Real-World Deployment and Impact:** Translating the research findings into real-world applications, such as recommendation systems or review filtering mechanisms, would allow for practical evaluation and validation of their effectiveness in shaping user experiences and driving positive outcomes for the gaming community.

5 Conclusion

The research presented in this paper has delved into the intricate relationship between sentiment analysis and the prediction of game success and review helpfulness within the dynamic landscape of the Steam platform. By harnessing the power of both lexicon-based and deep learning approaches, we have demonstrated the potential of sentiment analysis to extract actionable insights from the vast repository of user-generated reviews. The strategic integration of positional weighting within the TF-IDF vectorization framework further underscores the significance of contextual nuances in sentiment analysis, paving the way for the development of more precise and insightful predictive models.

The comparative evaluation of Linear Support Vector Classifier (SVC) and Multinomial Naive Bayes (NB) models has revealed their efficacy in predicting review sentiment and helpfulness, with the Linear SVC demonstrating superior precision. The inherent interpretability of the Naive Bayes model, however, has provided valuable insights into the underlying factors that influence these predictions. The strategic application of SMOTE to address class imbalance has further fortified the robustness and fairness of our models, ensuring their applicability across diverse datasets.

The implications of this research transcend the confines of academia, offering tangible benefits to a multitude of stakeholders within the gaming industry. By accurately predicting review helpfulness or sentiment, platforms like Steam can curate content that fosters engagement, trust, and a sense of community among their users. Game developers can leverage sentiment analysis to gain a deeper understanding of player perceptions, enabling data-driven decision-making in game design, marketing strategies, and community management initiatives. Moreover, users themselves stand to benefit from more personalized recommendations and a clearer understanding of the collective sentiment surrounding a game, empowering them to make informed choices and discover titles that resonate with their preferences.

The culmination of this research marks not an end, but a new beginning. The path forward is illuminated by exciting possibilities, including the exploration of more advanced deep learning architectures, the incorporation of additional features, and the integration of explainable AI techniques to enhance transparency and trust in sentiment analysis models. The ultimate aspiration is to cultivate a more informed, engaging, and user-centric gaming ecosystem, where sentiment analysis serves as a guiding light, shaping the future of game development and enriching the player experience.

6 Appendix

6.1 Steam Reviews Web Scrapping Code

```
import steamreviews
import pandas as pd

# List of Steam App IDs and corresponding game names
games = {
    578080: "PUBG: BATTLEGROUNDS",
    271590: "Grand Theft Auto V",
    1245620: "Cyberpunk 2077",
    550: "Left 4 Dead 2",
    730: "Counter-Strike 2",
    105600: "Terraria",
    1091500: "Cyberpunk 2077",
    218620: "PAYDAY 2",
    322330: "Don't Starve Together",
    292030: "The Witcher 3: Wild Hunt"
}

# Initialize the request parameters
request_params = {
    'filter': 'all',         # Fetch all reviews, sorted by helpfulness
    'num_per_page': 100,     # Number of reviews per request (max allowed is
100)
    'language': 'english'    # Only get reviews in English
}

# Maximum number of reviews to collect per app ID
max_reviews = 20000

# List to hold all reviews across different app IDs
all_reviews = []

# Loop through each app ID and collect reviews
for app_id, game_name in games.items():
    reviews_collected = []
    while len(reviews_collected) < max_reviews:
        # Download reviews in chunks
        review_dict, _ = steamreviews.download_reviews_for_app_id(app_id,
chosen_request_params=request_params)

        # Extract reviews from the current batch
        reviews = review_dict['reviews'].values()
        reviews_collected.extend(reviews)
```

```
        # Break if fewer reviews were returned than requested (end of
available reviews)
        if len(reviews) < request_params['num_per_page']:
            break

    # Add app_id and game_name to each review
    for review in reviews_collected[:max_reviews]:
        review_data = {
            'app_id': app_id,
            'game_name': game_name,
            'review': review['review']
        }
        all_reviews.append(review_data)

# Convert the collected reviews to a pandas DataFrame
df = pd.DataFrame(all_reviews)

# Save the DataFrame to a single CSV file
csv_file_path = "steam_reviews_all_games.csv"
df.to_csv(csv_file_path, index=False)

print(f"All reviews saved to {csv_file_path}")
```

6.2 Streamlit Sentiment Analysis Tool Code

```
import streamlit as st
from textblob import TextBlob
import cleantext
import pandas as pd

st.header('Sentiment Analysis Tool')

# Section for Text Analysis
with st.expander('Analyze Individual Text'):
    user_input = st.text_input('Enter your text:')
    if user_input:
        sentiment = TextBlob(user_input)
        st.write('Polarity Score:', round(sentiment.sentiment.polarity, 2))
        st.write('Subjectivity Score:',
round(sentiment.sentiment.subjectivity, 2))

    clean_input = st.text_input('Text to Clean:')
    if clean_input:
        cleaned_text = cleantext.clean(clean_input, clean_all=False,
                                       extra_spaces=True, stopwords=True,
                                       lowercase=True, numbers=True,
punct=True)
        st.write('Cleaned Text:', cleaned_text)
```

```
# Section for CSV File Analysis
with st.expander('Analyze CSV File'):
    uploaded_file = st.file_uploader('Upload CSV File')

    # Function to calculate polarity score
    def calculate_polarity(text):
        sentiment_analysis = TextBlob(text)
        return sentiment_analysis.sentiment.polarity

    # Function to determine sentiment category
    def determine_sentiment(polarity_score):
        if polarity_score >= 0.5:
            return 'Positive'
        elif polarity_score <= -0.5:
            return 'Negative'
        else:
            return 'Neutral'

    if uploaded_file:
        data = pd.read_csv(uploaded_file)
        if 'Unnamed: 0' in data.columns:
            data = data.drop(columns=['Unnamed: 0'])
        data['Polarity Score'] = data['review'].apply(calculate_polarity)
        data['Sentiment'] = data['Polarity Score'].apply(determine_sentiment)
        st.write(data.head(10))

        @st.cache_data
        def convert_to_csv(dataframe):
            return dataframe.to_csv().encode('utf-8')

        csv_data = convert_to_csv(data)

        st.download_button(
            label="Download CSV",
            data=csv_data,
            file_name='sentiment_analysis.csv',
            mime='text/csv',
        )
```

6.3 SVC and NBC Code

```
# Importing necessary libraries and downloading NLTK resources
import re, nltk
nltk.download('punkt')
nltk.download('stopwords')
nltk.download('omw-1.4')
nltk.download('wordnet')
```

```
import numpy as np
import pandas as pd
# Imblearn library can be installed using pip install imblearn
from imblearn.over_sampling import SMOTE      # Importing SMOTE for
oversampling the imbalanced dataset
from bs4 import BeautifulSoup     # Importing BeautifulSoup for HTML parsing
from nltk.stem import WordNetLemmatizer
from nltk.corpus import stopwords
from sklearn.feature_extraction.text import TfidfVectorizer
from sklearn.model_selection import StratifiedKFold
from sklearn import metrics
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB
import joblib


# %%
# Reading dataset as dataframe
df = pd.read_csv("C:/Users/hp/Desktop/Code/train data 6 games.csv")
pd.set_option('display.max_colwidth', None) # Setting this so we can see the
full content of cells
pd.set_option('display.max_columns', None) # To make sure we can see all the
columns in output window
df.info() # Displaying information about the dataset

# Plotting simple bar graph to know the count of reveiews by stars
import matplotlib.pyplot as plt

ratings_count = df['review_score'].value_counts().sort_index()

plt.figure(figsize=(10, 5))
plt.bar(ratings_count.index, ratings_count.values)
plt.title('Count of Reviews by Stars')
plt.xlabel('Stars')
plt.ylabel('Count')
plt.show()



# %%
def cleaner(review): # Cleaning reviews
    soup = BeautifulSoup(review, 'lxml') # removing HTML entities such as
‘&amp’,’&quot’,'&gt'; lxml is the html parser and shoulp be installed using
'pip install lxml'
    souped = soup.get_text()
    re1 = re.sub(r"(@|http://|https://|www|\\x)\S*", " ", souped) #
substituting @mentions, urls, etc with whitespace
    re2 = re.sub("[^A-Za-z]+"," ", re1) # substituting any non-alphabetic
character that repeats one or more times with whitespace
```

```
    tokens = nltk.word_tokenize(re2)
    lower_case = [t.lower() for t in tokens]

    stop_words = set(stopwords.words('english'))
    filtered_result = list(filter(lambda l: l not in stop_words, lower_case))

    wordnet_lemmatizer = WordNetLemmatizer()
    lemmas = [wordnet_lemmatizer.lemmatize(t) for t in filtered_result]
    return lemmas

# %%
df['cleaned_review'] = df['review_text'].apply(cleaner)
df = df[df['cleaned_review'].map(len) > 0] # removing rows with cleaned
reviews of length 0
print("Printing top 5 rows of dataframe showing original and cleaned
reviews....")
print(df[['review_text','cleaned_review']].head())
df['cleaned_review'] = [" ".join(row) for row in df['cleaned_review'].values]
# joining tokens to create strings. TfidfVectorizer does not accept tokens as
input
data = df['cleaned_review']
Y = df['review_score'] # target column
tfidf = TfidfVectorizer(min_df=.00086, ngram_range=(1,3)) # min_df=.00086
means that each ngram (unigram, bigram, & trigram) must be present in at least
20 documents for it to be considered as a token (23305*.00086=20). This is a
clever way of feature engineering
tfidf.fit(data) # learn vocabulary of entire data
data_tfidf = tfidf.transform(data) # creating tfidf values
pd.DataFrame(pd.Series(tfidf.get_feature_names_out())).to_csv('vocabulary_stea
mreviews.csv', header=False, index=False)
print("Shape of tfidf matrix: ", data_tfidf.shape)

# %%
# Implementing Support Vector Classifier
model1 = LinearSVC() # kernel = 'linear' and C = 1

# Running cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1) # 10-fold
cross-validation
scores=[]
iteration = 0
smote = SMOTE(random_state = 101)
for train_index, test_index in kf.split(data_tfidf, Y):
    iteration += 1
    print("Iteration ", iteration)
    # Resetting index of Y to ensure it aligns with the indices generated by
kf.split
```

```
    Y_reset = Y.reset_index(drop=True)
    X_train, Y_train = data_tfidf[train_index], Y_reset[train_index]
    X_test, Y_test = data_tfidf[test_index], Y_reset[test_index]
    X_train,Y_train = smote.fit_resample(X_train,Y_train) # Balancing training
data
    model1.fit(X_train, Y_train) # Fitting SVC
    Y_pred = model1.predict(X_test)
    score = metrics.precision_score(Y_test, Y_pred) # Calculating precision
    print("Cross-validation precison: ", score)
    scores.append(score) # appending cross-validation precision for each
iteration
mean_precision = np.mean(scores)
print("Mean cross-validation precision: ", mean_precision)

# %%
# Implementing Naive Bayes Classifier
model2 = MultinomialNB()

# Running cross-validation
kf = StratifiedKFold(n_splits=10, shuffle=True, random_state=1) # 10-fold
cross-validation
scores=[]
iteration = 0
smote = SMOTE(random_state = 101)
for train_index, test_index in kf.split(data_tfidf, Y):
    iteration += 1
    print("Iteration ", iteration)
    # Resetting index of Y to ensure it aligns with the indices generated by
kf.split
    Y_reset = Y.reset_index(drop=True)  # Reset index for each iteration
    X_train, Y_train = data_tfidf[train_index], Y_reset[train_index]  # Use
Y_reset
    X_test, Y_test = data_tfidf[test_index], Y_reset[test_index]    # Use
Y_reset
    X_train,Y_train = smote.fit_resample(X_train,Y_train) # Balancing training
data
    model2.fit(X_train, Y_train) # Fitting NBC
    Y_pred = model2.predict(X_test)
    score = metrics.precision_score(Y_test, Y_pred) # Calculating precision
    print("Cross-validation precison: ", score)
    scores.append(score) # appending cross-validation precision for each
iteration
mean_precision = np.mean(scores)
print("Mean cross-validation precision: ", mean_precision)

# %%
# Additional Visualizations and Data Exploration
```

```
# 1. Distribution of Review Lengths

df['review_length'] = df['review_text'].apply(len)

plt.figure(figsize=(10, 5))
plt.hist(df['review_length'], bins=50, alpha=0.75)
plt.title('Distribution of Review Lengths')
plt.xlabel('Review Length')
plt.ylabel('Frequency')
plt.show()

# 2. Word Clouds for Different Ratings (with word removal)

from wordcloud import WordCloud, STOPWORDS

# Add words you want to exclude to the STOPWORDS set
stop_words_to_remove = {'game', 'play', 'time'}
STOPWORDS.update(stop_words_to_remove)

for rating in df['review_score'].unique():
    subset = df[df['review_score'] == rating]
    text = " ".join(subset['cleaned_review'])
    wordcloud = WordCloud(max_font_size=50, max_words=100,
background_color="white", stopwords=STOPWORDS).generate(text)
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")

    plt.title(f"Word Cloud for Rating {rating}")
    plt.show()

# 3. Most Frequent Words (with word removal)

from collections import Counter

all_words = [word for review in df['cleaned_review'] for word in
review.split()]
word_counts = Counter(all_words)

# Remove unwanted words from the word counts
words_to_remove = {'game', 'play', 'time','get', 'one', 'like', 'really',
'would', 'much', 'make', 'ever', 'thing'}
for word in words_to_remove:
    del word_counts[word]

most_common_words = word_counts.most_common(20)

plt.figure(figsize=(10, 5))
```

```
plt.bar(*zip(*most_common_words))
plt.xticks(rotation=45)
plt.title('Most Frequent Words')
plt.xlabel('Word')
plt.ylabel('Frequency')
plt.show()
```

6.4 Recommendation Code

```
import pandas as pd
import matplotlib.pyplot as plt
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from collections import Counter
from wordcloud import WordCloud

def extract_file(file_name):
    file = pd.read_csv(file_name)
    return file

file_name = "C:/Users/hp/Desktop/Code/train data 6 games.csv"
dataframe = extract_file(file_name=file_name)
dataframe.head(5)

print(f'Rows: {dataframe.shape[0]} Columns: {dataframe.shape[1]}')

dataframe.info()


# %%
dataframe[dataframe.isnull().any(axis=1)]

dataframe = dataframe.dropna()

print(dataframe.isnull().any(axis=1).sum())

dataframe.shape


# %%
list_app_id = dataframe['app_id'].unique().tolist()
print(f'Total number of games analyzed: {len(list_app_id)}')


# %%
# Games with a value of 1 are recommended by the community. Games with -1 are
not recommended by the community
review_games = pd.Series(dataframe['review_score'].value_counts())
review_games
```

```
# %%
# We can see that about 14% of the reviews do not recommend the listed games
pd.Series(dataframe['review_score'].value_counts(normalize=True))


# %%
# Reviews with a value of 0 are recommended by the community. Reviews with 1
are not recommended by the community
review_com = pd.Series(dataframe['review_votes'].value_counts())
review_com


# %%
# Pie chart to show the total number of games recommended and not recommended
by the community
plt.figure(figsize=(8, 8))  # Set the figure size
plt.pie(review_games.values, labels=['Recommended', 'Not Recommended'],
autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Game Recommendations')

# Show the pie chart
plt.axis('equal')  # Ensures that the pie chart is drawn as a circle
plt.show()


# %%
# Pie chart to show the total number of reviews recommended and not
recommended by the community
plt.figure(figsize=(8, 8))  # Set the figure size
plt.pie(review_com.values, labels=['Recommended', 'Not Recommended'],
autopct='%1.1f%%', startangle=140)

plt.title('Distribution of Review Recommendations')

# Show the pie chart
plt.axis('equal')  # Ensures that the pie chart is drawn as a circle
plt.show()


# %%
# games with the highest recommendation score (Horizontal bar chart)
top_games = dataframe.groupby('app_name')['review_score'].sum().nlargest(10)
plt.barh(top_games.index, top_games.values)
for i, value in enumerate(top_games.values):
    plt.text(value, i, str(value))
plt.xlim(0, 88000)
```

```
plt.title('Most Recommended Games')
plt.show()


# %%
# Check the degree of recommendation of reviews within the most recommended
games (Bar chart)
comments_top_games =
dataframe.groupby('app_name')['review_votes'].sum().nlargest(10)
plt.barh(comments_top_games.index, comments_top_games.values)
for i, value in enumerate(comments_top_games.values):
    plt.text(value, i, str(value))
plt.xlim(0, 16000)
plt.title('Most Recommended Comments')
plt.show()



# %%
import nltk
from nltk.tokenize import word_tokenize
from nltk.corpus import stopwords
from nltk.sentiment import SentimentIntensityAnalyzer
import pandas as pd
import matplotlib.pyplot as plt
from wordcloud import WordCloud
import spacy

# Load spaCy model
nlp = spacy.load('en_core_web_sm')

# Define the path and download NLTK data if not already done
nltk_data_path = 'C:/nltk_data'
nltk.data.path.append(nltk_data_path)
nltk.download('punkt', download_dir=nltk_data_path)
nltk.download('stopwords', download_dir=nltk_data_path)

# Load stopwords
stop_words = set(stopwords.words('english'))

# Define SentimentIntensityAnalyzer
sid = SentimentIntensityAnalyzer()

# Sample data
dataframe_sentiments = dataframe.sample(n=10, random_state=42)

# Function to tokenize using spaCy
def spacy_tokenize(text):
    return [token.text for token in nlp(text) if token.is_alpha]
```

```
# Analyze sentiment
sentiments = []
for review in dataframe_sentiments['review_text']:
    tokens = spacy_tokenize(review.lower())
    filtered_tokens = [word for word in tokens if word not in stop_words]
    sentiment_score = sid.polarity_scores(' '.join(filtered_tokens))
    if sentiment_score['compound'] >= 0.05:
        sentiment = 'positive'
    elif sentiment_score['compound'] <= -0.05:
        sentiment = 'negative'
    else:
        sentiment = 'neutral'
    sentiments.append(sentiment)

# Add sentiment to DataFrame
dataframe_sentiments['sentiment'] = sentiments
print(dataframe_sentiments.head(10))

# Generate Word Cloud
text = ' '.join(dataframe_sentiments['review_text'].astype(str))  # Combine
all reviews into one string
wordcloud = WordCloud(width=800, height=400, background_color='white',
stopwords=stop_words).generate(text)

# Display the Word Cloud
plt.figure(figsize=(10, 5))
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```

6.5 Sentiment Analysis Code

```
import pandas as pd
from textblob import TextBlob
from nltk.sentiment.vader import SentimentIntensityAnalyzer
import nltk
from transformers import DistilBertTokenizer,
DistilBertForSequenceClassification, pipeline
import torch

# Download VADER lexicon
nltk.download('vader_lexicon')

# Load the CSV file
file_path = r'C:\Users\hp\Desktop\Code\steam_reviews_all_games.csv'
df = pd.read_csv(file_path)
```

```
# Ensure all reviews are strings and handle missing values
df['review'] = df['review'].astype(str).fillna('')

# Initialize VADER sentiment analyzer
sid = SentimentIntensityAnalyzer()

# Load pre-trained fine-tuned DistilBERT model for sentiment analysis
tokenizer = DistilBertTokenizer.from_pretrained('distilbert-base-uncased-
finetuned-sst-2-english')
model = DistilBertForSequenceClassification.from_pretrained('distilbert-base-
uncased-finetuned-sst-2-english')

# Load sentiment-analysis pipeline with DistilBERT
sentiment_pipeline = pipeline('sentiment-analysis', model=model,
tokenizer=tokenizer)

# Function to analyze sentiment using TextBlob
def analyze_sentiment_textblob(review):
    analysis = TextBlob(review)
    return analysis.sentiment.polarity

# Function to analyze sentiment using VADER
def analyze_sentiment_vader(review):
    scores = sid.polarity_scores(review)
    return scores['compound']

# Function to analyze sentiment using DistilBERT
def analyze_sentiment_bert(review):
    # Ensure text length does not exceed model's max length
    inputs = tokenizer(review, truncation=True, padding=True, max_length=512,
return_tensors='pt')
    outputs = model(**inputs)
    logits = outputs.logits
    prediction = torch.argmax(logits, dim=-1)
    return 'POSITIVE' if prediction.item() == 1 else 'NEGATIVE'

# Apply sentiment analysis with TextBlob
df['sentiment_textblob'] = df['review'].apply(analyze_sentiment_textblob)
# Apply sentiment analysis with VADER
df['sentiment_vader'] = df['review'].apply(analyze_sentiment_vader)
# Apply sentiment analysis with DistilBERT
df['sentiment_bert'] = df['review'].apply(analyze_sentiment_bert)

# Save the results to a new CSV file
output_path =
r'C:\Users\hp\Desktop\Code\steam_reviews_all_games_with_sentiments.csv'
df.to_csv(output_path, index=False)
print(f"Sentiment analysis completed and saved to '{output_path}'")
```

7 Bibliography

Al Mursyidy Fadhlurrahman, J. *et al.* (2023) 'Sentiment Analysis of Game Reviews on STEAM using BERT, BiLSTM, and CRF', in *2023 International Conference on Electrical Engineering and Informatics (ICEEI)*. *2023 International Conference on Electrical Engineering and Informatics (ICEEI)*, Bandung, Indonesia: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/ICEEI59426.2023.10346219.

Bais, R. and Odek, P. (2017) 'Sentiment Classification on Steam Reviews', in. Available at: https://www.semanticscholar.org/paper/Sentiment-Classification-on-Steam-Reviews-Bais-Odek/6884bf544932e3bc0aad8e29204c6c801dcd1056 (Accessed: 20 August 2024).

Bian, P., Liu, L. and Sweetser, P. (2021) 'Detecting Spam Game Reviews on Steam with a Semi-Supervised Approach', in *The 16th International Conference on the Foundations of Digital Games (FDG) 2021*. *FDG'21: The 16th International Conference on the Foundations of Digital Games 2021*, Montreal QC Canada: ACM, pp. 1–10. Available at: https://doi.org/10.1145/3472538.3472547.

Dong, Q. *et al.* (2022) 'Network Public Opinion Sentiment Analysis based on Bert Model', in *2022 IEEE 10th International Conference on Information, Communication and Networks (ICICN)*. *2022 IEEE 10th International Conference on Information, Communication and Networks (ICICN)*, Zhangye, China: IEEE, pp. 662–666. Available at: https://doi.org/10.1109/ICICN56848.2022.10006589.

Eberhard, L. *et al.* (2018) 'Investigating Helpfulness of Video Game Reviews on the Steam Platform', in *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*. *2018 Fifth International Conference on Social Networks Analysis, Management and Security (SNAMS)*, Valencia: IEEE, pp. 43–50. Available at: https://doi.org/10.1109/SNAMS.2018.8554542.

Jorge, G.A.Z. and Pardo, T.A.S. (2023) 'SteamBR: a dataset for game reviews and evaluation of a state-of-the-art method for helpfulness prediction', in *Anais do XII Brazilian Workshop on Social Network Analysis and Mining (BraSNAM 2023)*. *Brazilian Workshop on Social Network Analysis and Mining*, Brasil: Sociedade Brasileira de Computação - SBC, pp. 210–215. Available at: https://doi.org/10.5753/brasnam.2023.230132.

Lin, D. *et al.* (2019) 'An empirical study of game reviews on the Steam platform', *Empirical Software Engineering*, 24(1), pp. 170–207. Available at: https://doi.org/10.1007/s10664-018-9627-4.

Mayangsari, M.K., Syarif, I. and Barakbah, A. (2023) 'Evaluation of Stratified K-Fold Cross Validation for Predicting Bug Severity in Game Review Classification', *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control* [Preprint]. Available at: https://doi.org/10.22219/kinetik.v8i3.1740.

Pang, B., Lee, L. and Vaithyanathan, S. (2002) 'Thumbs up?: sentiment classification using machine learning techniques', in *Proceedings of the ACL-02 conference on Empirical methods in natural language processing - EMNLP '02*. *the ACL-02 conference*, Not Known: Association

for Computational Linguistics, pp. 79–86. Available at: https://doi.org/10.3115/1118693.1118704.

Panwar, A. and Bhatnagar, V. (2022) 'Sentiment Analysis of Game Review Using Machine Learning in a Hadoop Ecosystem':, in I.R. Management Association (ed.) *Research Anthology on Implementing Sentiment Analysis Across Multiple Disciplines*. IGI Global, pp. 463–483. Available at: https://doi.org/10.4018/978-1-6684-6303-1.ch026.

Pongkhan, T. *et al.* (2024) 'Conversation Sentiment Analysis in League of Legends Game Community', *2024 IEEE International Conference on Cybernetics and Innovations (ICCI)*, pp. 1–6. Available at: https://doi.org/10.1109/ICCI60780.2024.10532675.

Taqiuddin, R., Bachtiar, F.A. and Purnomo, W. (2021) 'Opinion Spam Classification on Steam Review using Support Vector Machine with Lexicon-Based Features', *Kinetik: Game Technology, Information System, Computer Network, Computing, Electronics, and Control* [Preprint]. Available at: https://doi.org/10.22219/kinetik.v6i4.1323.

Urriza, I.M. and Clariño, M.A.A. (2021) 'Aspect-Based Sentiment Analysis of User Created Game Reviews', in *2021 24th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*. *2021 24th Conference of the Oriental COCOSDA International Committee for the Co-ordination and Standardisation of Speech Databases and Assessment Techniques (O-COCOSDA)*, pp. 76–81. Available at: https://doi.org/10.1109/O-COCOSDA202152914.2021.9660559.

Wang, Z., Chang, V. and Horvath, G. (2021) 'Explaining and Predicting Helpfulness and Funniness of Online Reviews on the Steam Platform':, *Journal of Global Information Management*, 29(6), pp. 1–23. Available at: https://doi.org/10.4018/JGIM.20211101.oa16.

Yu, Y. *et al.* (2021) 'Esports Game Updates and Player Perception: Data Analysis of PUBG Steam Reviews', in *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*. *2021 13th International Conference on Knowledge and Systems Engineering (KSE)*, Bangkok, Thailand: IEEE, pp. 1–6. Available at: https://doi.org/10.1109/KSE53942.2021.9648670.

https://pypi.org/project/steamreviews/