TEXT MINING PROJECT

Predicting Airbnb Delistings: A Natural Language Processing Approach

Group 11
Arda Kaykusuz | 20230766
Arif Maharramov | 20230770

Bruno Jardim | Miguel Cruz

June 2024

Contents

1. Introduction

Airbnb is a widely recognized online platform and hospitality marketplace that offers a diverse range of accommodations, including apartments, houses, and individual rooms listed by hosts. The platform has revolutionized how people find and book accommodations, providing unique and personalized lodging experiences across the globe. In this project, our primary objective is to develop a sophisticated Natural Language Processing (NLP) pipeline capable of predicting whether a listed property will be unlisted in the upcoming quarter. By leveraging advanced NLP techniques, we aim to create a robust model that can accurately forecast the unlisting status of properties on Airbnb.

2. Data Exploration

First, we started to define our datasets and understand the details of each CSV file. We have 2 different datasets which are train/test and train/test_review. While the train/test dataset consists of property description, host description, and unlisted/listed information, the train/test review CSV consists of comments and index columns.

2.1. Data Type Conversion

We started to examine the dataset checking their row counts and columns' data types. Then, we realized that some data types might be problematic when we use the data, so we decided to convert some of them not to handle the following parts.

2.2. Duplicate Variables

After preparing the data we wanted to look into the target variable to see if the dataset is unbalanced. As seen in **Figure 1, the dataset** was quite unbalanced. While 72,7% of listings are unlisted 27,3% are listed.

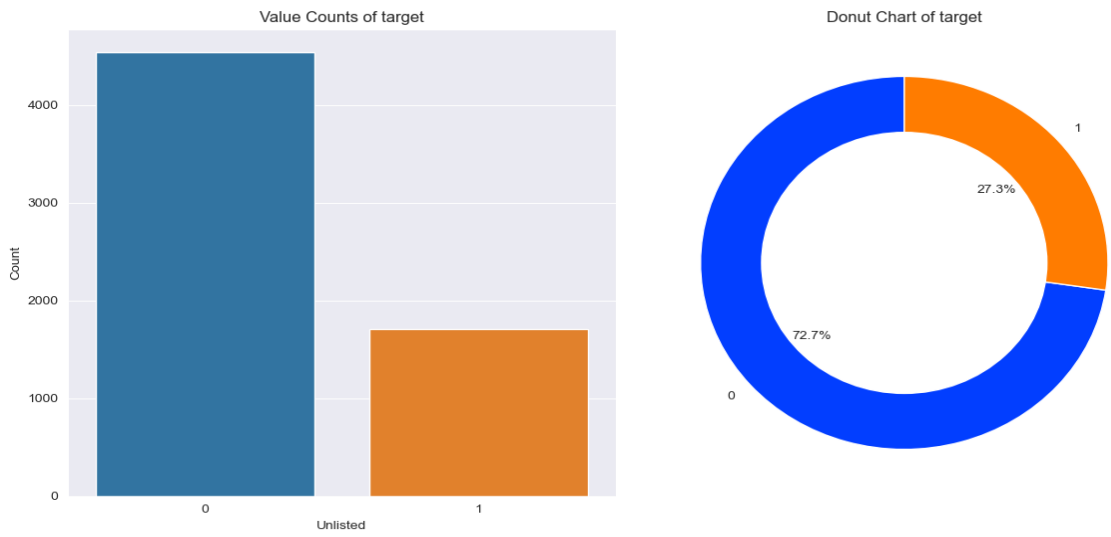

Figure 1. Value Counts of Target

The next step was to examine the duplicate values. By trying various variations, we detected duplicate data in two datasets and decided that they should be eliminated in the next steps. In the train dataset, which we specified as df_trained, 192 duplicates were detected. We saw that there are a couple of the same text in the other columns and considered that the variables are coming from HTML. In the following chapters, we decided to remove or organize them. In addition to this, we saw short explanations in the comments column and some emojis.

2.3. Missing Values

In this section, we tried to define missing values on the dataset. Luckily, we have not observed any missing values in any dataset. While going through the analysis, as mentioned previously, we saw short words, symbols, and numbers. There was no missing value overall but there were many useless explanations for our analysis. We detected some useless texts, which we listed below.

- We realized that our dataset included emojis. To address this, we encoded emojis as sentiments and excluded these instances, allowing us to focus on detecting other common missing values in the dataset.
- There were so many HTML symbols like “/b>
”.
- There was only punctuation in the comments, host about, and description columns.

All of them were detected and we decided to handle all of them during the preprocessing section.

2.4. Some Inferences Made During Exploration

We wanted to see some distributions during the exploration. While examining the distribution of comments per property, we observed that the distribution of comments per property is heavily right-skewed, indicating that most properties receive a small number of comments, with a significant peak around 0-50 comments. This means that many properties attract minimal feedback. The distribution has a long tail, extending up to 800 comments, representing a few properties with an unusually high number of comments. The sharp decline after the initial peak highlights that while some listings are very popular, the majority receive far fewer comments. This insight is important for understanding engagement levels and the popularity of Airbnb listings. **(Figure 2)**

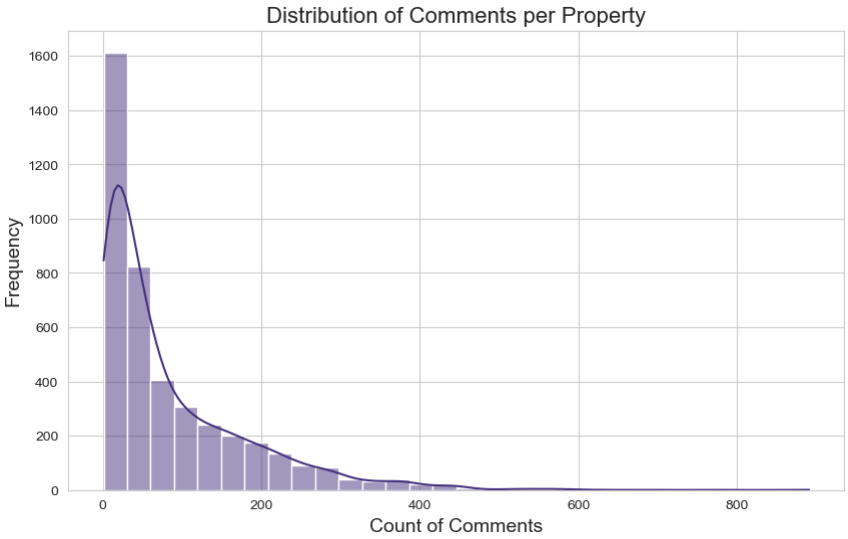

Figure 2. Distributions of Comments per Property

In addition to this when looking into the distribution of comments between unlisted and listed properties, it was seen in unlisted properties (left histogram), **(Figure 3)** the distribution is more evenly spread across a range of comment counts, with a significant number of properties having very few comments. Conversely, listed properties (right histogram) exhibit a heavily right-skewed distribution, where most properties receive a small number of comments, and a small number of properties receive a very high number of comments. This contrast suggests that properties with higher engagement (more comments) are more likely to remain listed on Airbnb.

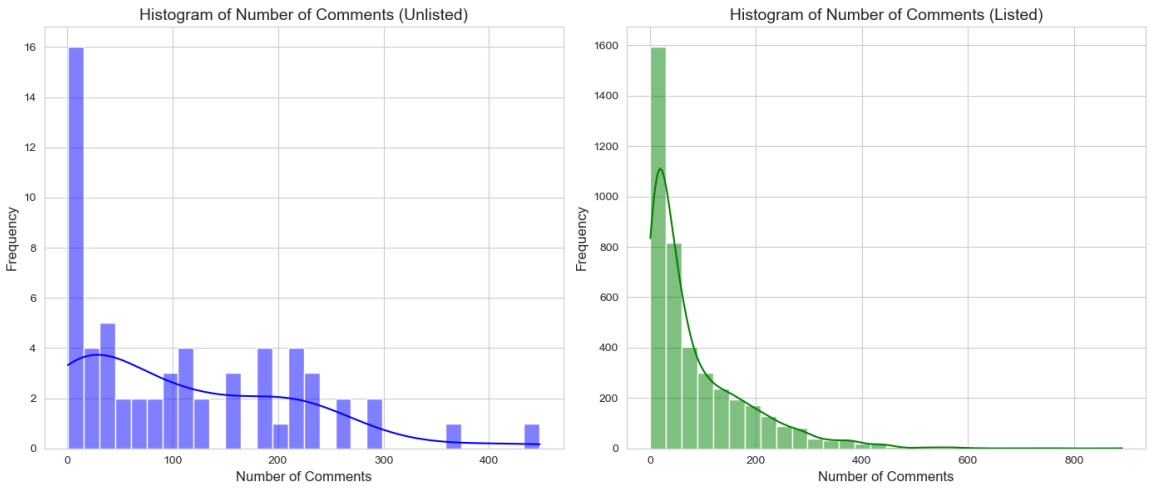

Figure 3. Histogram of Number of Comments for Unlisted and Listed Listing

Also, as seen in **Figure 4**, the proportion of properties with no comments is significantly higher for unlisted properties compared to listed ones. Specifically, over 80% of unlisted properties have no comments, whereas the proportion is much lower for listed properties. This suggests that engagement is a strong indicator of whether a property remains listed on Airbnb. Properties with higher engagement are more likely to stay active on the platform.

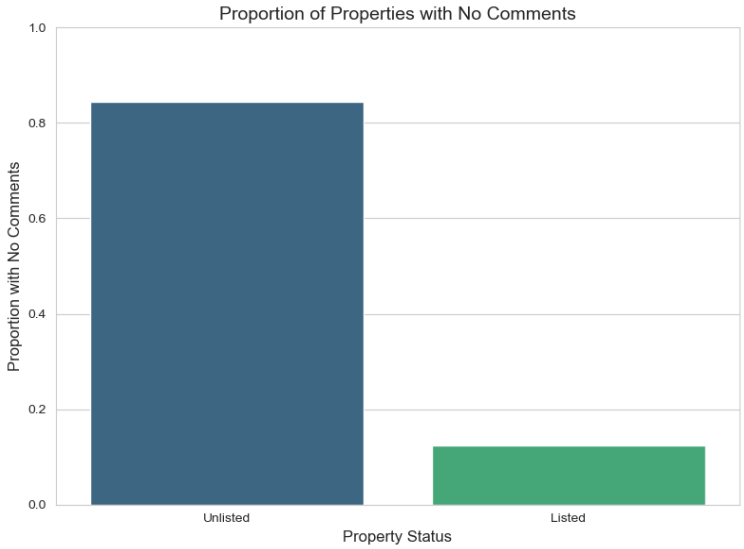

Figure 4. The proportion of Properties with No Comments

2.5. Language Detection

Since Airbnb is used all over the world, we wanted to see the languages of listings. To do this, we imported a library that can detect languages. After analyzing the language distribution, we found that English is the most used language, followed by Portuguese, French, and Spanish. **(Figure 5)**

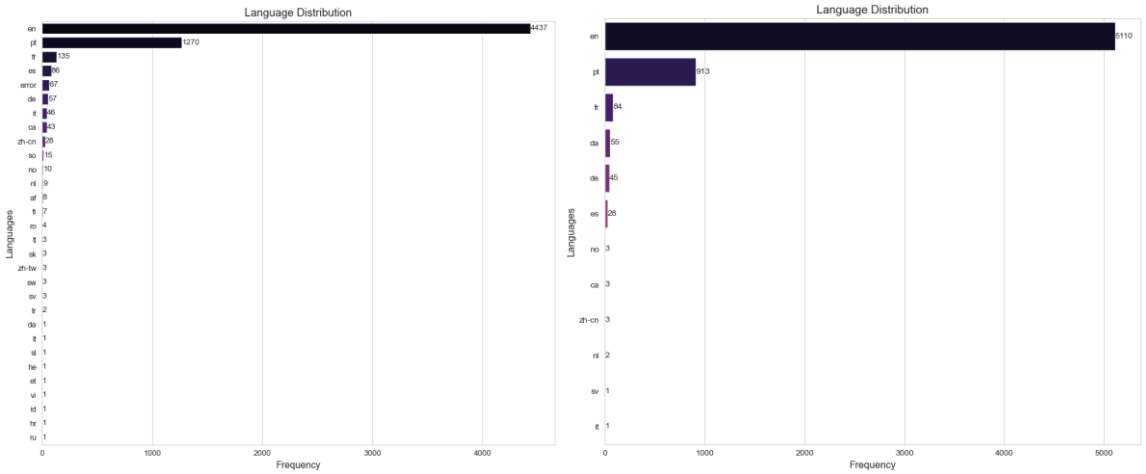

Figure 5. Language Count Over “Host About” and “Description” Columns

2.6. Word Count Distribution and Most Recurring Words

We noticed some interesting trends in the word counts for different sections. Property descriptions are quite consistent in length, with a peak of around 150 words. On the other hand, host descriptions show a lot of variability, with most falling between 0-200 words, but some extending beyond 1000 words. Guest comments tend to be shorter, with a sharp peak at lower word counts, although there are some longer comments. These trends suggest that

property descriptions are usually concise, host descriptions vary widely, and guest comments are brief.

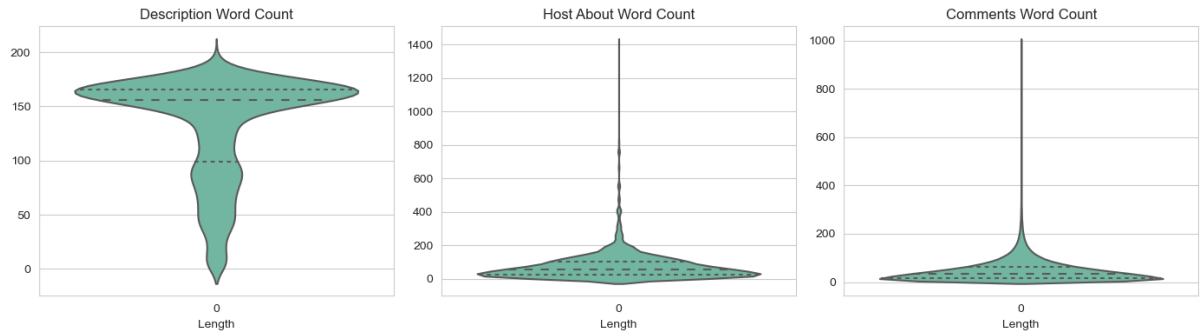

Figure 7. Violin Plots of Word Count for Each Section

In addition to examining violin plots, we wanted to identify the most repeated words. To conduct the analysis Figure 7 gave us the results properly. We saw that the most repetitive words are stop words that we expected. As seen previously HTML marks are also one of the most repetitive words in our dataset even if they are not real words.

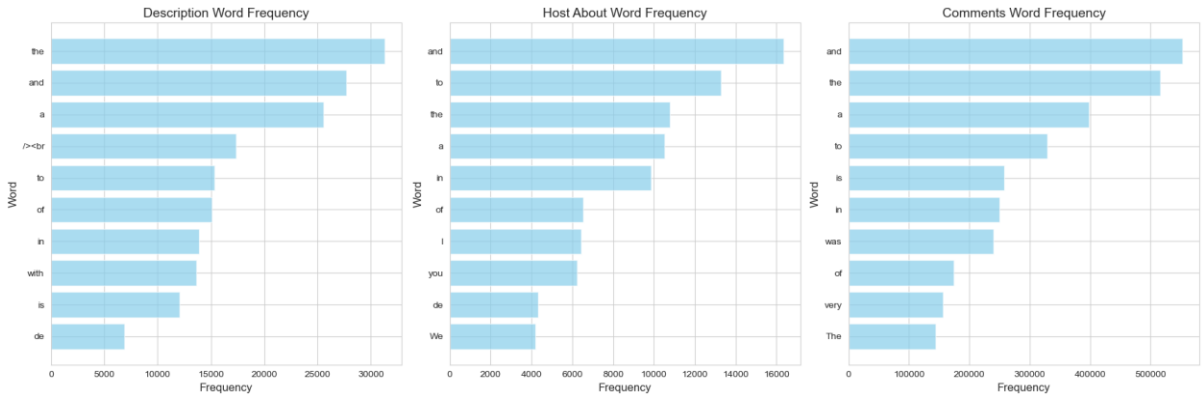

Figure 8. Most Repetitive Words

3. Data Preprocessing

First, we started handling duplicate variables. Then we applied various strategies for text standardization:

3.1. Text Standardization and Extra Methods

We took multiple steps to standardize the text. Firstly, we converted all the text to lowercase to ensure consistency. We then removed specific patterns such as money values, HTML signs, emails (like the "@ "sign), and URLs with special tokens to maintain their essential meaning while minimizing the vocabulary size. Also, we excluded punctuation marks and numerical data, including numbers and Roman numerals, to focus solely on the linguistic content. To remove them, we created functions to find these symbols first. Then, stop words which are

non-informative words for our analysis were removed from the text. As we mentioned and explained in **Figure 8** stop words were covering the text as expected but after removing this, we got more clear and explainable data for our analysis. In addition to this for our model, it was going to be beneficial.

As mentioned previously, especially in the comments columns we detected so much emoji. Instead of removing them, we decided to keep them. In particular, the usage of emojis is a part of our lives. That is why we thought it was going to be beneficial to infer from the analysis. By importing emoji library, we converted them to the text and removed underscores. In this way, we extended the comment section by applying this strategy.

Apart from these steps, we converted abbreviations like *can't* and *don't*. In this way, we got rid of all the signs that could affect the model's performance and made it simpler.

After removing strange signs, and emoji handlings, we applied stemming to reduce words to their root forms, normalizing variations and enabling better grouping of similar words, thus reducing the vocabulary size. Stemming was chosen over lemmatization for its greater effectiveness in simplifying the lexicon.

The preprocessing involved two rounds. In the first round, stemming and other steps were applied only to the properties dataset, while the reviews data remained unchanged. This approach ensured that the processed datasets would be suitable for generating TF-IDF representations later in the analysis.

3.2. Sentiment Analysis

We conducted a sentiment analysis to understand if there is a difference between listed and unlisted properties. As seen in Figure 9, both listed and unlisted properties have a similar distribution of sentiment in their reviews, with a large majority being positive, followed by neutral, and a small proportion being negative.

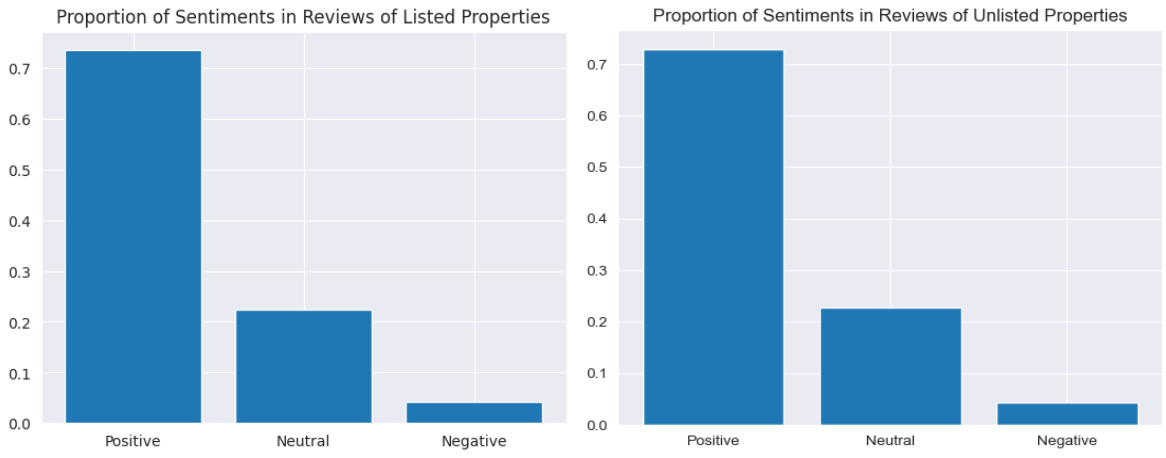

Figure 9. The proportion of Sentiments in Reviews of Listed&Unlisted Properties

4. Feature Engineering & Classification

4.1. Embedding Methods

Embeddings play essential roles while building NLP models because they convert words and phrases into numerical vectors that capture their semantic meaning, allowing machine learning models to understand and process text more effectively by recognizing patterns, relationships, and context within the data. [1]

In our project, we opted to use GloVe embedding because it creates the word embeddings through the global word-word co-occurrence frequencies from a large corpus, thus offering a holistic view of word associations and is also faster to train on large corpora.[2] Secondly, TF-IDF was selected because of its capacity to calculate the importance of each term with regards to the overall documents in a collection, which can help identify relevant words and is easy to interpret.[3] As third embeddings method, we utilized Word2Vec embeddings in our classification problem due to their powerful advantages. It allows for the efficient handling of large vocabularies and improves the performance of downstream models like Logistic Regression, LSTMs, and MLPs by providing rich feature representations. In the following part, it is mentioned the classifiers. [4]
We also tried to use DistilBERT and XLM-Roberta embeddings methods, but in our analysis, they do not perform very well so we discarded from our analysis.

4.2. Classification Methods

We implemented Logistic Regression for classification in our analysis due to its several key advantages. It offers simplicity and interpretability, allowing us to understand the influence of individual features on the classification outcome. Its computational efficiency ensures quick training even on large text datasets. Logistic Regression also provides flexibility in feature engineering, supports regularization to prevent overfitting, and scales well with large datasets. Furthermore, it produces probabilistic outputs that can enhance decision-making processes.[5] These advantages led us to believe that Logistic Regression would yield reliable results for our analysis. After implementing the logistic regression, it shows satisfactory results. The outputs of the logistic regression models are in **Figure 10, Figure 11.**

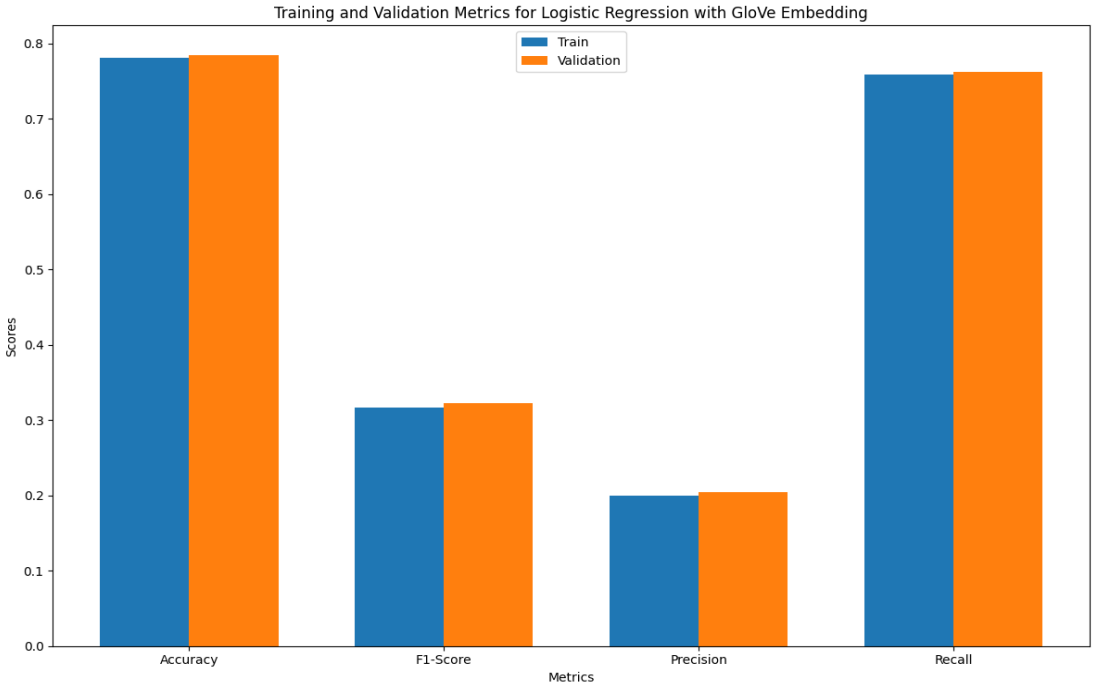

Figure 10. Training and Validation Scores for Logistic Regression with Glove Embeddings

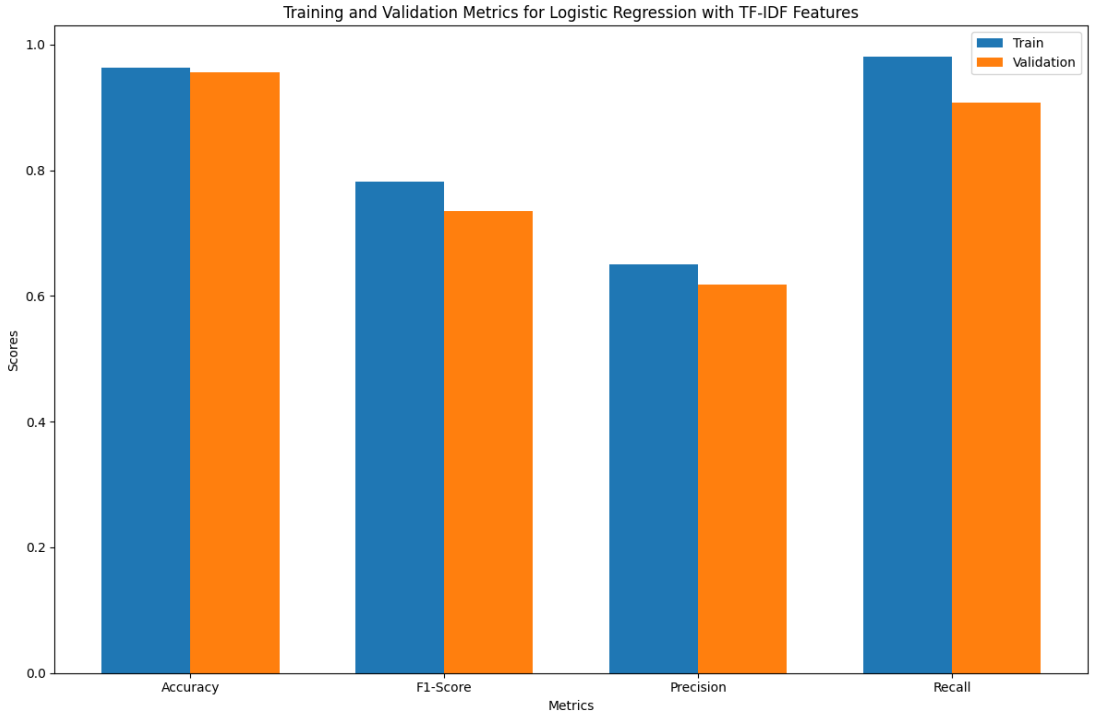

Figure 11. Training and Validation Scores for Logistic Regression with TF-IDF Features

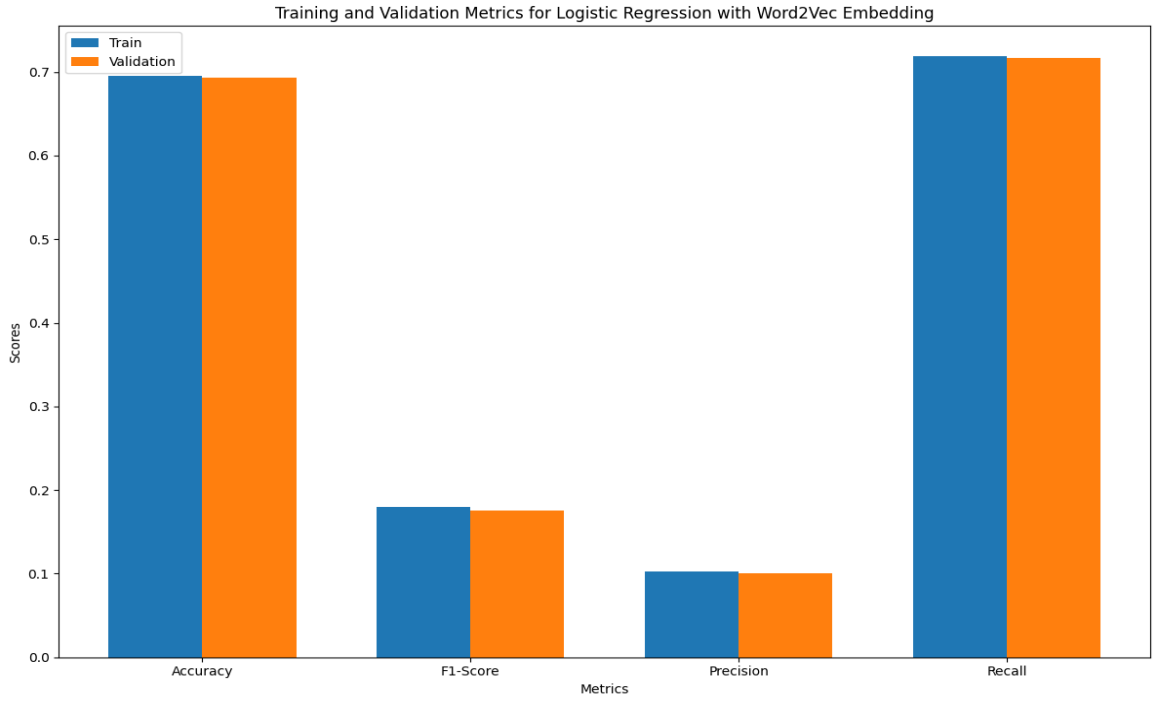

Figure 12. Training and Validation Scores for Logistic Regression with Word2Vec Embeddings

When we analyzed the output and focused on the F1-Score due to the unbalanced data, Logistic Regression with TF-IDF clearly outperformed other models. While it had the highest F1-Score of 0.781967, indicated a better balance between precision and recall. Glove embeddings, despite achieving reasonable recall, had significantly lower F1-Scores around 0.32, highlighting their poor precision. And lastly, Word2Vec embeddings performed the worst, with F1-Scores around 0.18, reflecting both low precision and recall. Therefore, TF-IDF features are the most effective for this logistic regression model.

Secondly, we decided to use Long Short-Term Memory (LSTM) networks for our study because of few reasons. LSTMs are specially designed to understand the long-term dependencies in the data and are therefore suitable for capturing the context and the structure in the text. They help to solve the vanishing gradients problem quite well and provide a better learning of the long-term dependencies.[6] In addition to this, LSTMs are very versatile in terms of input data's length and can accept sequences of any length which makes them ideal for processing natural language. It improves the efficiency of update operations on long-term value storage, especially on tasks such as sentiment analysis, language modeling, and text generation. These strengths made us believe that LSTMs would prove useful in improving our classification task. As seen in Figure **13-14-15** it gives good results with different embedding techniques.

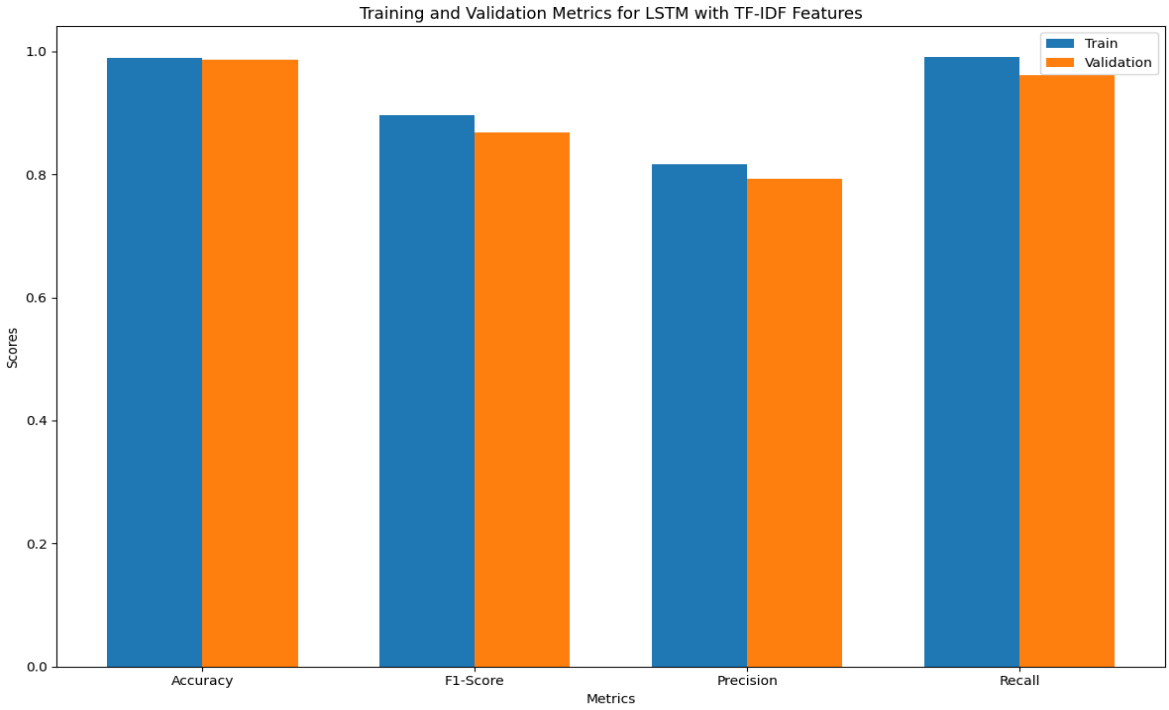

Figure 13. Training and Validation Scores for LSTM with TF-IDF Features

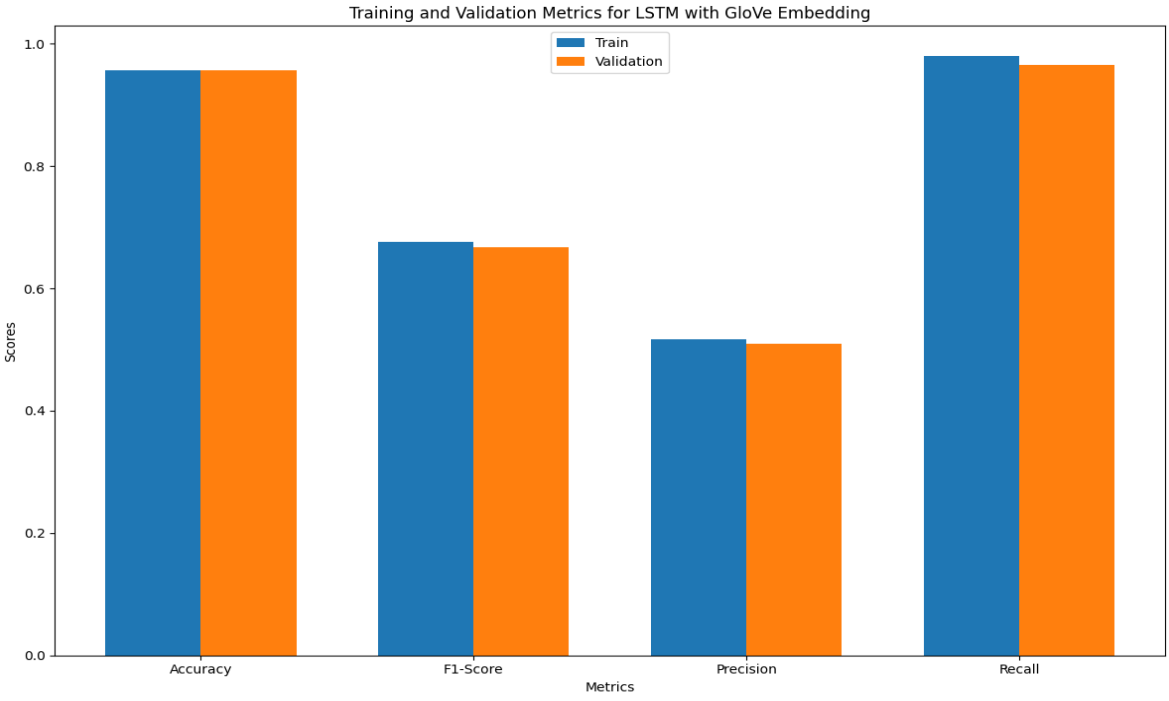

Figure 14. Training and Validation Scores for LSTM with Glove Embeddings

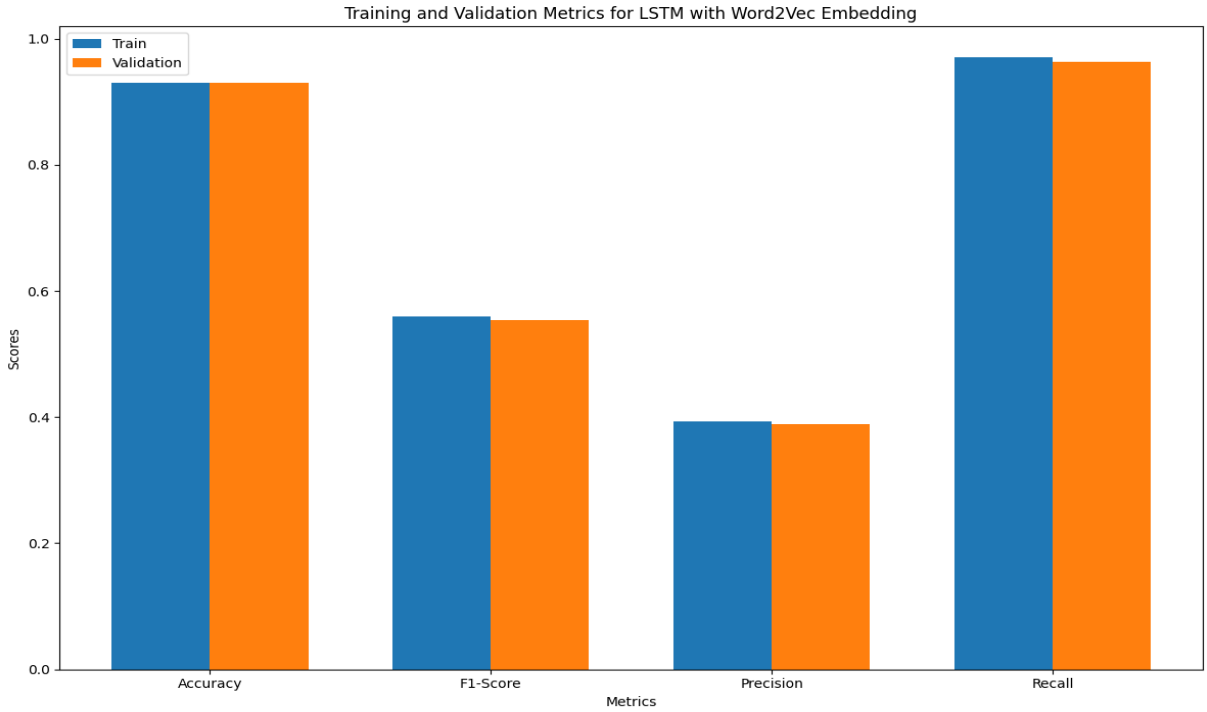

Figure 15. Training and Validation Scores for LSTM Word2Vec Embeddings

As in logistic Regression, F-IDF features yielded the best performance here too, especially when considering the F1-score. The highest F1-Score of 0.895626 indicated an excellent balance between precision and recall, making it highly effective for our unbalanced dataset. LSTM with Glove embeddings performed moderately well, with F1-Scores around 0.67, showing decent recall but lower precision. LSTM with Word2Vec embeddings showed the weakest performance among the LSTM models, with F1-Scores around 0.56, reflecting poorer precision despite high recall. Thus, LSTM with TF-IDF features is the most robust model in this context.

In the third model, we have chosen a Multi-Layer Perceptron (MLP) for our classification problem for the following reasons. MLPs are highly adaptive and capable of capturing complex, non-linear mapping functions because of the ability to stack layers of interconnected neurons.[7] They are quite easy to set up. MLPs are also relatively insensitive to the amount of training data provided to them and can generalize well if provided with enough training data and if care is taken to use techniques such as cross validation to prevent over fitting. Moreover, they are efficient in terms of computational complexity for moderate size of data and can be easily fine-tuned to get better results. We got the results faster than some other models. As can be seen in Figure 17-18-19 MLP also showed a good performance on our analysis.

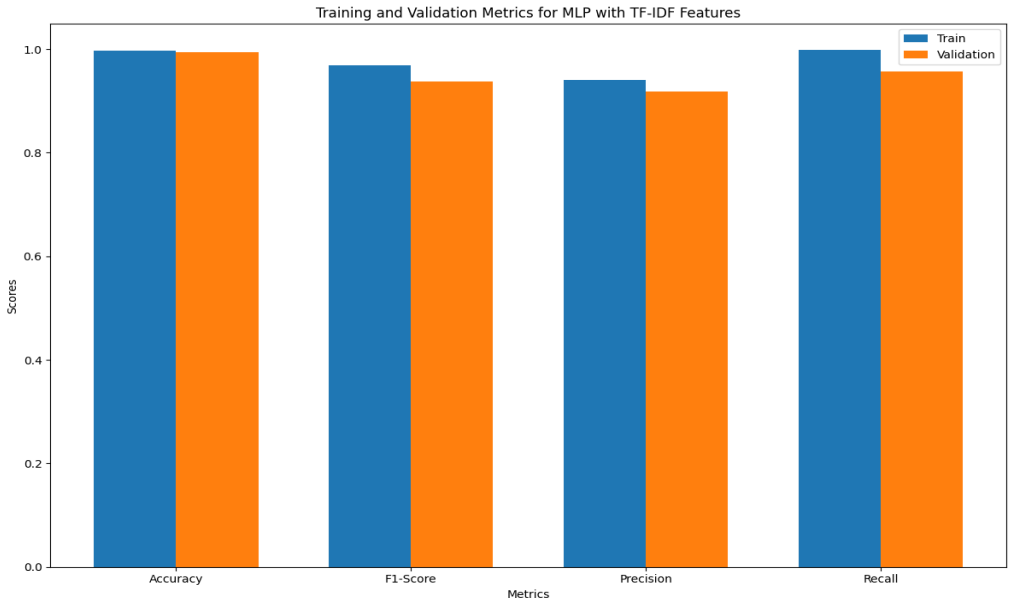

Figure 16. Training and Validation Scores for MLP with TF-IDF Features

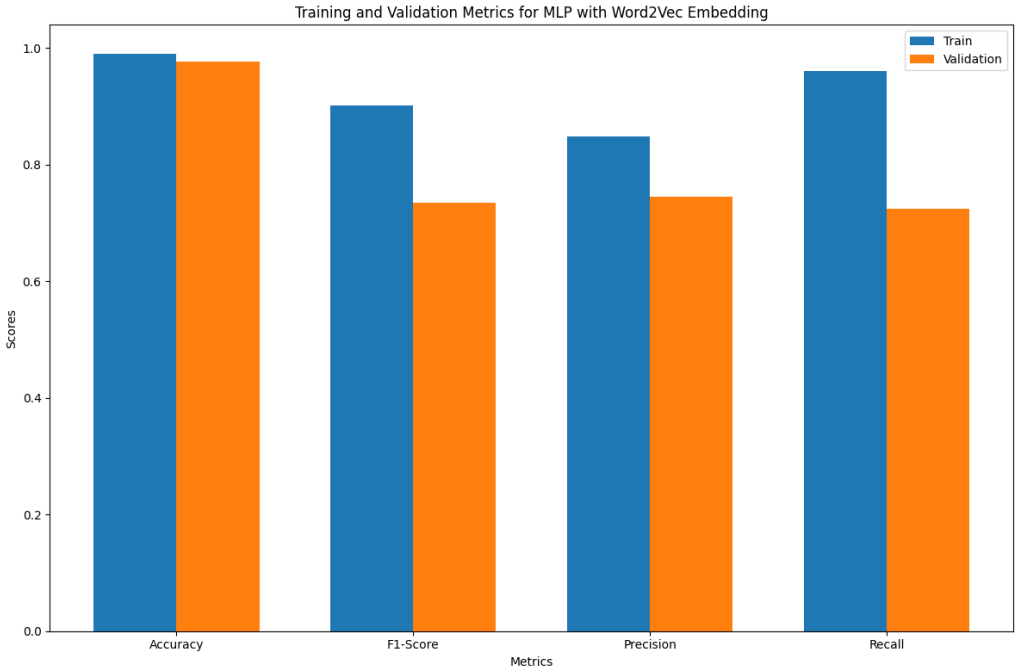

Figure 17. Training and Validation Scores for MLP Word2Vec Features

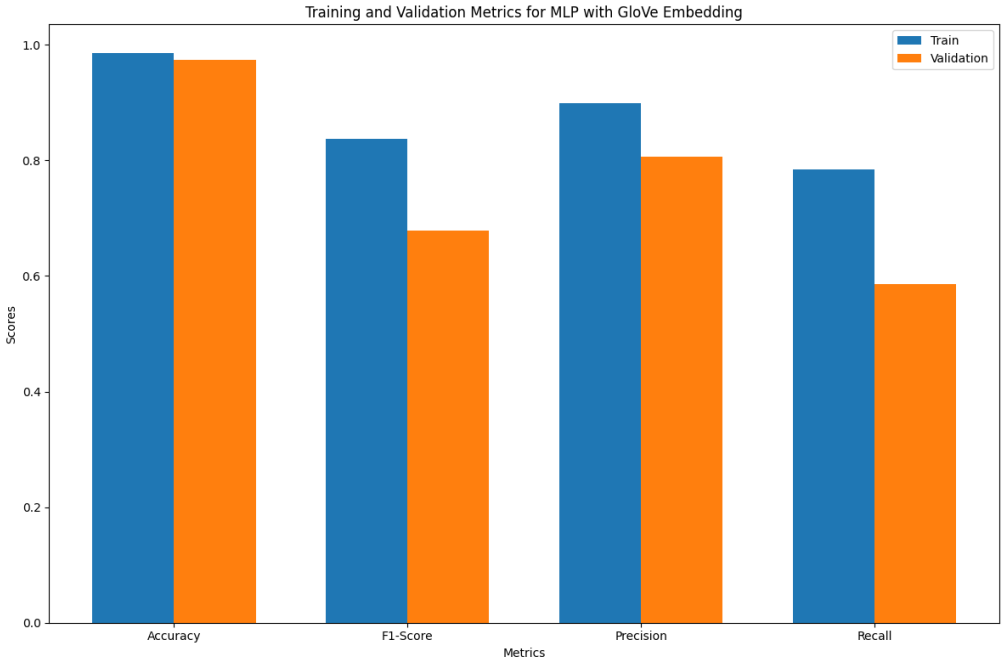

Figure 18. Training and Validation Scores for MLP with Glove Embeddings

When we analyzed the data, MLP with Glove embeddings showed a significant drop in performance, with F1-Scores of 0.837658 and 0.678565, indicating better precision than recall. MLP with Word2Vec embeddings performed moderately well, with F1-Scores around 0.901042 and 0.734263, but still falls short compared to TF-IDF features.

Lastly, we used Gated Recurrent Units (GRUs) for classification since they provide multiple benefits over other models. They are less parameterized, computationally efficient, and easier to train as compared to LSTMs, but they do not lag in performance. In this aspect, GRUs can overcome the vanishing gradient problem, and learn long-term dependencies in text stable. These benefits make us extend the analysis to incorporate GRUs with the purpose of increasing the model variety and accuracy.

Additionally, in our project we also tried to implement a KNN (K- Nearest Neighbor) classifier for our classification model but it does not perform as expected and it was extremely slower than the other models. Due to computational problems, we decided not to add them to our analysis.

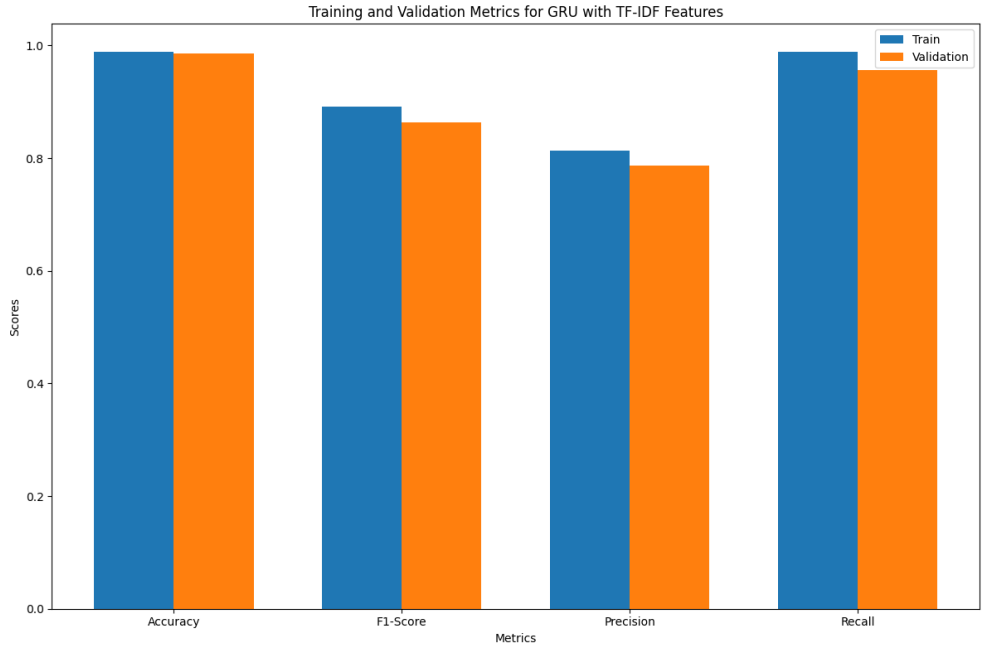

Figure 19. Training and Validation Scores for Gru TF-IDF Features

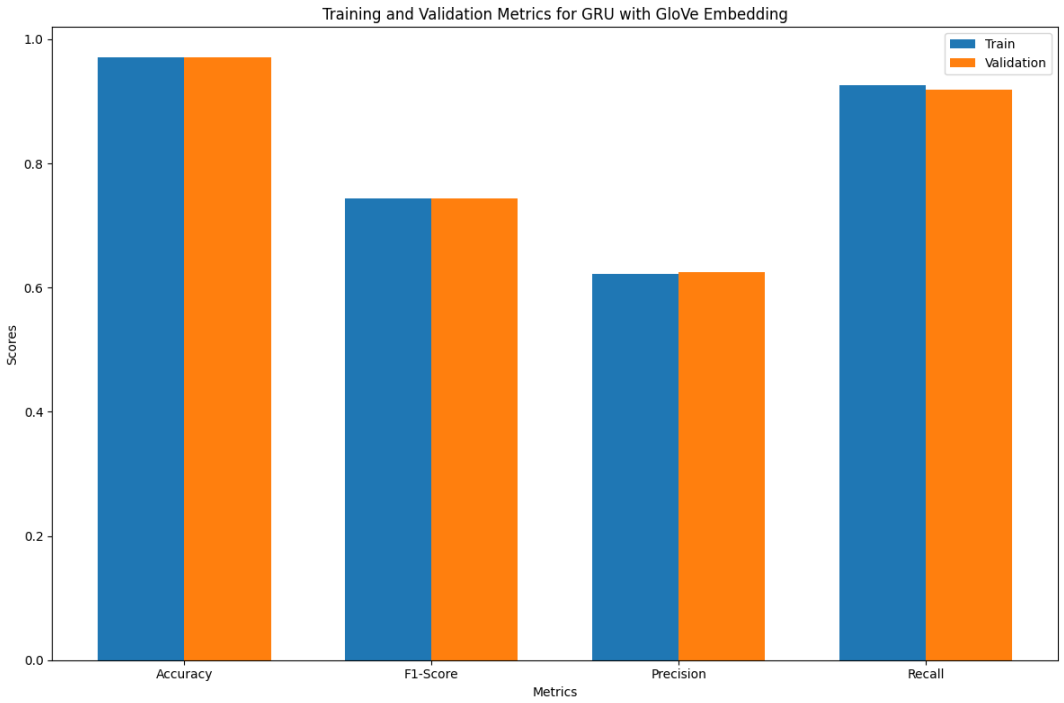

Figure 20. Training and Validation Scores for Gru with Glove Embedding

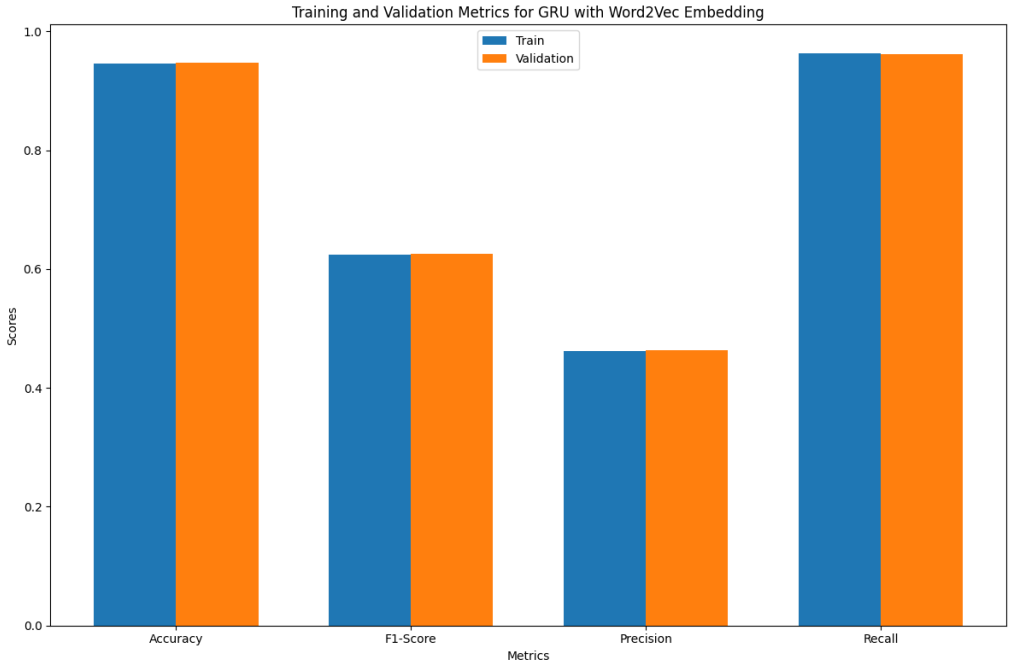

Figure 21. Training and Validation Scores for GRU with Word2Vec Embedding

In the analysis of GRU it was seen that using TF-IDF features results in the best performance. The F1-Scores of 0.892136 and 0.863729 indicated a strong balance between precision and recall. GRU with Glove embeddings showed moderate performance, with F1-Scores around 0.744, reflecting reasonable precision and high recall. Lastly, GRU with Word2Vec embeddings performed the weakest, with F1-Scores around 0.625. Therefore, GRU with TF-IDF features was the most robust model for our analysis.

Models & Embedding Techniques	**Metrics**	**Train**	**Validation**
Logistic Regression with Glove Emb.	Accuracy	0.780648	0.774569
	F1-Score	0.31629	0.302628
	Precision	0.199783	0.184638
	Recall	0.758813	0.73196
Logistic Regression With TF-IDF	Accuracy	0.963425	0.956003
	F1-Score	0.781967	0.73535
	Precision	0.650117	0.617951
	Recall	0.980905	0.907818
Logistic Regression with Word2Vec Emb.	Accuracy	0.69516	0.693154
	F1-Score	0.17934	0.175489
	Precision	0.102446	0.099987
	Recall	0.719019	0.716616
LSTM With Glove Emb.	Accuracy	0.95659	0.956112
	F1-Score	0.676615	0.667362
	Precision	0.51658	0.509723
	Recall	0.980312	0.966163
LSTM With TF-IDF	Accuracy	0.989300	0.986743

	F1-Score	0.895626	0.868497
	Precision	0.816990	0.792425
	Recall	0.991010	0.960725
LSTM With Word2Vec Emb.	Accuracy	0.929280	0.929335
	F1-Score	0.559774	0.554156
	Precision	0.393304	0.388882
	Recall	0.970579	0.963746
MLP With Glove Emb.	Accuracy	0.985917	0.974711
	F1-Score	0.837658	0.678565
	Precision	0.898774	0.806237
	Recall	0.784324	0.585801
MLP With TF-IDF Features	Accuracy	0.997020	0.994204
	F1-Score	0.968802	0.937713
	Precision	0.940411	0.918817
	Recall	0.998960	0.957402
MLP With Word2Vec Emb.	Accuracy	0.990229	0.976115
	F1-Score	0.901042	0.734263
	Precision	0.848710	0.744641
	Recall	0.960253	0.724169
GRU with Glove Emb.	Accuracy	0.970515	0.971131
	F1-Score	0.744276	0.743674
	Precision	0.622075	0.624512
	Recall	0.926226	0.919033
GRU with TF-IDF Features	Accuracy	0.988925	0.986247
	F1-Score	0.892136	0.863729
	Precision	0.812752	0.787366
	Recall	0.988707	0.956495
GRU with Word2Vec Emb	Accuracy	0.946437	0.947659
	F1-Score	0.624961	0.626082
	Precision	0.462496	0.464129
	Recall	0.963373	0.961631

Table 1. Performance of Models Comparison

5. Discussion of results

Considering the exceptionally high performance metrics of the MLP model, we concluded that there is a significant risk of overfitting. This model's high accuracy and F1 scores on both training and validation sets might not generalize well to new data, indicating it might be less reliable in real-world applications. Given our imbalanced dataset, focusing on the F1 score is crucial for making an informed decision.

The LSTM with TF-IDF features stands out as a more balanced choice. It achieves an F1 score of 0.8956 on the training set and 0.8685 on the validation set, indicating it performs well on both. This consistency shows that it handles imbalanced data effectively without overfitting.

On the other hand, models using Glove and Word2Vec embeddings generally show lower F1 scores and more significant performance drops between training and validation, making them

less suitable for our needs. Logistic regression models, while simpler, do not achieve the same level of F1 score as the LSTM with TF-IDF.

6. Conclusion

In summary, our project focused on developing a sophisticated NLP pipeline to predict the unlisting status of Airbnb properties. We thoroughly examined given dataset, addressing issues such as duplicate entries, missing values, and data type inconsistencies. Through comprehensive data preprocessing, including text standardization and sentiment analysis, we prepared the data for effective modeling. Our feature engineering involved multiple embedding techniques such as GloVe, TF-IDF, and Word2Vec. We also tried to use embedding methods such as XLM-Roberta and Distilbert but because of computational difficulties we did not include them in the final report.

We implemented 4 classifiers: Logistic Regression, LSTM, MLP, and GRU. They were evaluated based on their performance metrics, particularly focusing on the F1 score due to the imbalanced nature of our dataset.

From our extensive analysis, it became evident that the LSTM with TF-IDF features emerged as the most balanced and robust model for our needs. Despite the impressive performance of the MLP model, concerns about overfitting led us to prioritize models with more consistent validation results. The LSTM with TF-IDF demonstrated strong F1 scores on both training and validation sets, indicating its capability to handle imbalanced data effectively without significant overfitting. Additionally, models using GloVe and Word2Vec embeddings showed lower F1 scores and more substantial performance variations, reinforcing our decision. Therefore, for a reliable and accurate prediction of property unlisting on Airbnb, the LSTM with TF-IDF features is our recommended choice, providing a balance of precision, recall, and overall robustness.

7. References

[1] Raj, J. S., Iliyasu, A. M., Bestak, R., & Baig Editors, Z. A. (n.d.). *Lecture Notes on Data Engineering and Communications Technologies 59 Innovative Data Communication Technologies and Application Proceedings of ICIDCA 2020*.
http://www.springer.com/series/15362

[2] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *EMNLP 2014 - 2014 Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference*, 1532–1543. https://doi.org/10.3115/v1/d14-1162

[3] Mohan, N. (n.d.). *NLP - Text Classification using TF-IDF Features*. Kaggle. Retrieved from https://www.kaggle.com/code/neerajmohan/nlp-text-classification-using-tf-idf-features

[4] Gupta, P. (2023, December 12). *Word Embeddings: A Survey of Methods and Applications*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2017/06/word-embeddings-count-word2veec/

[5] Brindha, S., Prabha, K., & Sukumaran, S. (2016). *A SURVEY ON CLASSIFICATION TECHNIQUES FOR TEXT MINING*.

[6] Kharwal, A. (2024, April 19). *LSTM for text classification*. Analytics Vidhya. Retrieved from https://www.analyticsvidhya.com/blog/2021/06/lstm-for-text-classification/

[7] GeeksforGeeks. (n.d.). *Classification using sklearn Multi-layer Perceptron*. Retrieved from https://www.geeksforgeeks.org/classification-using-sklearn-multi-layer-perceptron/