# Steam Game User Suggestion Prediction

Raluca-Andreea Gînga

$11^{th}$ of March, 2022

**Abstract**

In this documentation, there are presented several solutions for detecting and classifying users suggestions based on their reviews on Steam Games. The approaches presented in the next few pages include methods like TF-IDF to BERT and other pretrained models for Word Embeddings followed by the application of classical machine learning algorithms (some of them giving very good results), deep learning techniques used on some several datasets tha contain both the meta information extracted from the reviews, such as sentiment of the review, profanity checker, but the reviews representation as well.

## 1 Introduction

Computers are, nowadays, an absolute necessity that changes the world in which we live: how we work, how we shop, how we take care of our health, etc. In recent years, the development of the Internet and the involvement of the media has made it easier for people to access reviews, but also to give people the opportunity to give reviews, opinions about certain products.

The rapid growth of the popularity of video games has led to the creation of a billion-dollar industry. Due to the scale and popularity of this industry, developing a successful game is challenging.

Various studies have shown that gamers are that group of users who are extremely difficult to satisfy [1], making game quality an important issue. In order to better understand the quality of games in terms of user opinions, it is imperative to understand the fears, perplexities, dissatisfaction of gamers that is absolutely essential for game developers.

**Steam** is a global digital video game distribution platform for Windows, Linux, and MacOS developed by Valve Corporation. This allows users to buy and play computer games. In October 2012, Valve expanded the service to include software in addition to games. Although originally developed for use on Microsoft Windows, the client has expanded to include OS X and Linux versions and customers with limited functionality on the Xbox 360 and PlayStation 3 consoles and for Windows Phone, Android and iOS mobile devices. Valve also built SteamOS, a Linux-based operating system built around the Steam client designed for its Steam Machine microconsole line and any personal computer that meets the minimum specifications.

In this approach, the role of this documentation is to analyze the opinions of certain users regarding the games purchased and played on the Steam platform, one of the most popular digital gaming platforms.

The present project has the main purpose to use Natural Language Processing and Text Mining techniques in order to assign a sentiment/suggestion of a review based on its text. The role of this documentation is to analyze the opinions of certain users regarding the games purchased and played on the Steam platform, one of the most popular digital gaming platforms.

The remainder of this documentation is organized as follows: section 2 describes the dataset and the exploratory data analysis and feature engineering parts. Section 3 describes the details of the implemented Machine Learning and Deep Learning methods, followed by their corresponding results on section 4. The last section (5) concludes the documentation about the implemented methods and comes with some further work that could be done in order to research more about this task.

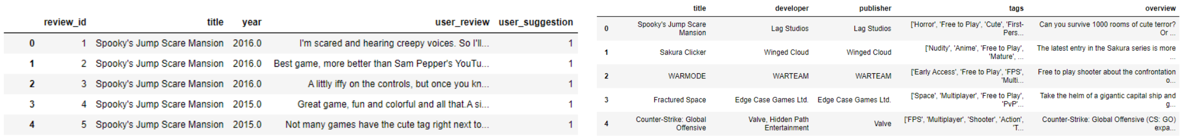
# 2 Dataset and Data Analysis

## 2.1 Overview of the dataset

The dataset of this task is consisting in three .csv files:

- ”train” file containing the training dataset of approximately 18 thousands of reviews with ”user_suggestion” as label
- ”test” file containing the testing dataset of approximately 8 thousands of reviews without the label (that should be predicted)
- ”game_overview” that contains the details of the games, such as: title, developer, publisher, tags, overview

An overview of the datasets format is shown in the figures from below:



| | review_id | title | year | user_review | user_suggestion |
|---|---|---|---|---|---|
| 0 | 1 | Spooky's Jump Scare Mansion | 2016.0 | I'm scared and hearing creepy voices. So I'll... | 1 |
| 1 | 2 | Spooky's Jump Scare Mansion | 2016.0 | Best game, more better than Sam Pepper's YouTu... | 1 |
| 2 | 3 | Spooky's Jump Scare Mansion | 2016.0 | A littly iffy on the controls, but once you kn... | 1 |
| 3 | 4 | Spooky's Jump Scare Mansion | 2015.0 | Great game, fun and colorful and all that.A si... | 1 |
| 4 | 5 | Spooky's Jump Scare Mansion | 2015.0 | Not many games have the cute tag right next to... | 1 |

| | title | developer | publisher | tags | overview |
|---|---|---|---|---|---|
| 0 | Spooky's Jump Scare Mansion | Lag Studios | Lag Studios | ['Horror', 'Free to Play', 'Cute', 'First-Pers... | Can you survive 1000 rooms of cute terror? Or ... |
| 1 | Sakura Clicker | Winged Cloud | Winged Cloud | ['Nudity', 'Anime', 'Free to Play', 'Mature', ... | The latest entry in the Sakura series is more ... |
| 2 | WARMODE | WARTEAM | WARTEAM | ['Early Access', 'Free to Play', 'FPS', 'Multi... | Free to play shooter about the confrontation o... |
| 3 | Fractured Space | Edge Case Games Ltd. | Edge Case Games Ltd. | ['Space', 'Multiplayer', 'Free to Play', 'PvP'... | Take the helm of a gigantic capital ship and g... |
| 4 | Counter-Strike: Global Offensive | Valve, Hidden Path Entertainment | Valve | ['FPS', 'Multiplayer', 'Shooter', 'Action', 'T... | Counter-Strike: Global Offensive (CS: GO) expa... |

Figure 1: Training (left) dataset. Game Overview (right) dataset.

## 2.2 Data Analysis & Feature Engineering

In order to understand more about our datasets, we decided to do some exploratory data analysis and create some new features as well to the existing datasets.

As we can see from the below figure (figure 2), the classes are pretty well balanced (although there are some imbalances, there are not so many imbalances to use data balancing methods, such as SMOTE, undersampling, oversampling and so on, so we'll stick to that if we get good results).

As we can see in the following figures (3), words like ”great”, ”free”, ”fun”, ”well”, ”better”, ”good”, ”love” belongs to the positive user suggestion, whereas ”kill”, ”bad”, ”terrible” belongs to the negative user suggestion.

In figure 4, we analyzed the sentiment of the reviews written by users with regards to year feature. As we can see, there are mostly positive reviews for each year, having a positive trend, whereas negative reviews obtained its maximum in 2018 year.

Another thing we analyzed was the number of reviews (positive and negative) given for each game. As we can see from figure 5, there are clearly most of the negative reviews given to Robocraft, Heroes & Generals, War Thunder and Bless Online, whereas there are good and positive reviews on Fractured Space, Path of Exile, Creativerse and Eternal Card Game.

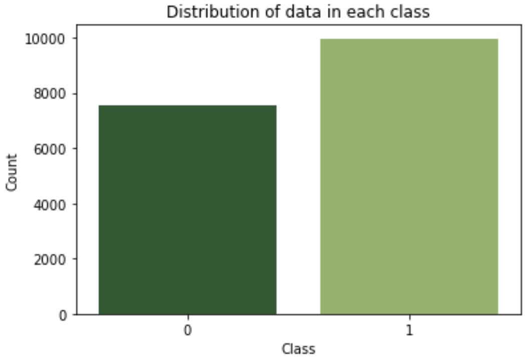


Figure 2: Distribution of reviews in each class

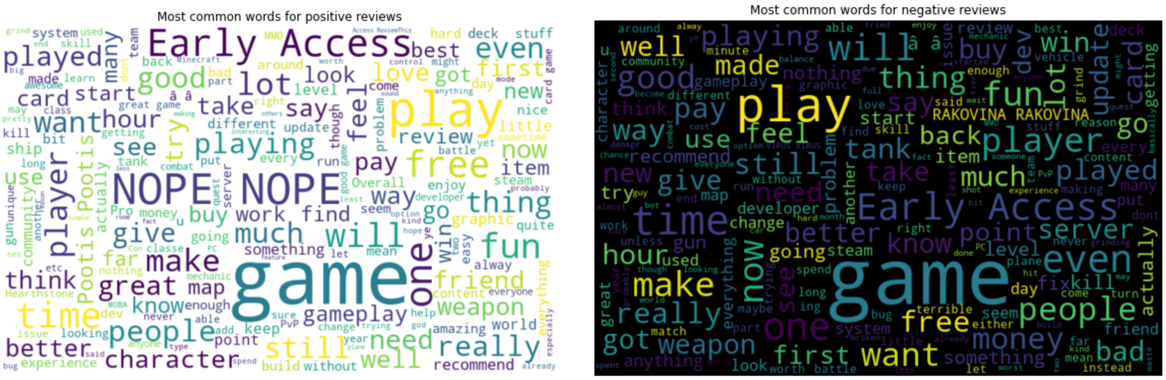


Figure 3: Positive (left) words. Negative (right) words.

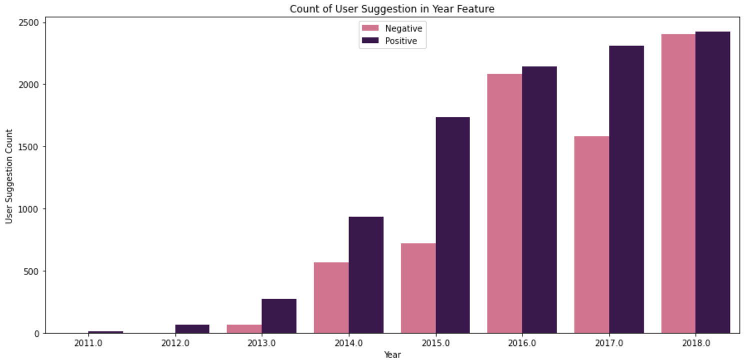


Figure 4: Count of user suggestions in year feature

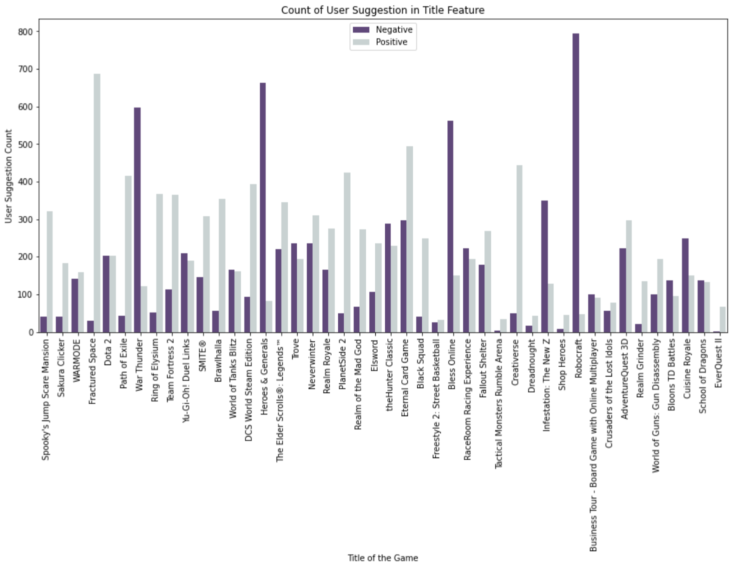


Figure 5: Count of user suggestions in year feature

Another step that we took was to analyze if there are reviews written in other languages. For this, we used Langdetect module [1] in order to detect the language of our reviews.

After language detection, there was noticed a 98.77%-1.23% English-nonEnglish split on the training dataset and 98.92%-1.08% on the test dataset. Since we had little data containing non-English reviews, we decided to drop them for our dataset. It could have been a good idea to keep those reviews with non-English, but there were some reviews with unknown language (maybe some random texts written) and we couldn't have captured anything in this way.

Another step that we took was to analyze how many reviews contain swearings and profanity language. This thing was done using profanity check[2] library.

There were found out 445 train reviews containing profanity language, whereas in test dataset there were noticed 184 reviews. We decided to use this as a feature.

Another feature we decided to use was sentiment analysis and polarity of each review. For this, we used Flair[3] which predicts in a very professional and precised way the sentiment of each review written.

Because we thought ”game_overview” dataset could provide to us some useful information, we decided to use from that dataset the tags associated to each game that was reviewed and label encoding them.

As preprocessing step, we used the following techniques:

- removing Early Access Review string from some of the users' reviews
- handling camelcase: ”heRocks” - ”he Rocks”
- extra whitespaces removal
- text decoding

[1] Lang detect https://pypi.org/project/langdetect/

[2] Profanity Check https://pypi.org/project/profanity-check/.

[3] Flair https://github.com/flairNLP/flair.

- special characters cleaning
- text lowercase
- tokenization
- stopwords removal
- removal of words shorter than 3 characters
- lemmatization

Thus, we created a final dataset (including the reviews and meta information) having the following features:

- a binary feature representing if the review contains profanity or not
- a binary feature representing the sentiment of the reviews (1 for positive and 0 for negative). Here, we also compared the accuracy between Flair representation and the actual label, obtaining almost 81% for both train, and validation datasets
- cleaned reviews
- the tags label-encoded for each game reviewed

# 3 Methods

## 3.1 Machine Learning models on three types of datasets

The first approach used was to apply different machine learning models on three types of datasets:

1. Reviews only - this includes only the clean reviews, without any feature
2. Combination between the meta information about the reviews and the reviews as well

The Machine Learning models that were used for all of these 2 types of dataset representation were:

- Logistic Regression
- Multinomial Naive Bayes
- Stochastic Gradient Descent
- Support Vector Machines
- Random Forest
- Light GBM
- XGBoost

We note that all of these models were supposed to 5-cross validation in order to have a precise representation of the real accuracy.

For the reviews dataset, we performed TF-IDF Vectorizer with 'word' analyzer and a features maximum of 5000 in order to obtain the numerical representations of the reviews.

### 3.1.1 Hyperparameter Optimization

Because it can be noticed in results section that Light GBM provided very good results on combined meta info + reviews datasets, we decided to use Bayesian Optimization in order to find the proper parameters for Light GBM model. In the following tables, we could see the chosen parameters that provided the best results on a 5-fold cross validation.

| | learning rate | number of leaves | feature fraction | bagging fraction | maximum depth |
|---|---|---|---|---|---|
| Combined Dataset | 0.023 | 42 | 0.307 | 0.801 | 14 |

## 3.2 Deep Learning techniques

Because we've seen that the combination between meta information and reviews have provided good results on Machine Learning algorithms, we decided to explore some Deep Learning techniques/algorithms in order to see if they provide better results. Also, some approaches used did also use the reviews only representation.

1. 3-Layer Neural Networks

   This algorithm used 3 dense layers with 128, 64 and 32 neurons with 'relu' activation and a final layer with 'softmax' activation. The optimized used was Adam, the loss was sparse categorical crossentropy and accuracy used as main metric. We also provided to the neural network model a class weight because we noticed that imbalance regarding the user suggestions distribution and we wanted to have a more appropriate way to balance them and avoid the situations in which the network predicts only the majority class.

2. Bert Embeddings + Bert for Sequence Classifier

   In this approach, we used BertTokenizer for tokenizing the reviews datasets and a classification BERT model - BertForSequenceClassification - that contains a single linear classification layer on top and that, surprinsingly, didn't provide the expected results, although this methods was used in a lot of Natural Language Processing competitions.

3. Sentence XLM-R + Bidirectional Gated Recurrent Unit

   For this technique, we used SentenceTransformers and provided a 'XLM-r-bert-base' model for embedding both the training and testing datasets. Then, as a deep learning architecture, we used 2 bidirectional GRU of 64 and 128 neurons, then a Dense layer of 256 neurons, a batch normalization, a dropout of 0.2, another dense layer of 64 neurons along with batch normalization and dropout, followed by a dense layer of 16 neurons and again, a batch normalization and concluding with the 10 classes. The model was trained on 50 epochs with Reduce LR On Plateau and Early Stopping in case the validation loss won't improve.

4. Word Embeddings (Word2Vec and FastText) + Long-Short Term Memory (LSTM)

   For this technique, we decided to use language models as well and to see what kind of results we're obtaining using them. We used a vector size of 300, a sliding window of 2, we tokenized the reviews and padded them to a maximum length of 300. We then formed an embedding matrix containing the language models representations of our reviews, being then fed to a LSTM classifier with 128 neurons. We also used "Reduce LR On Plateau" and "Early Stopping" to prevent the model not overfit.

# 4 Results and Analysis

## 4.1 Machine Learning models on 2 types of datasets

The results from the following table (1) contain the accuracy on 5-fold cross validation on the train dataset. As it can be seen from the below table, Light GBM and Logistic Regression seem to perform the best among all of the other Machine Learning algorithms. Comparing the modeling time of each algorithm, Light GBM is a very fast model and it also provided very good results. We declared it as the winner for all of the datasets, along with Logistic Regression, both in computation time, but in best accuracy as well.

| | Logistic Regression | Multinomial NB | SGD | SVM | Random Forest | Light GBM | XGBoost |
|---|---|---|---|---|---|---|---|
| Reviews | **84.81%** | 83.57% | 84.79% | 83.93% | 81.94% | 83.19% | 81.99% |
| Meta + Reviews | **87.64%** | 82.42% | 87.43% | 87.25% | 86.13% | **87.4%** | 86.68% |

Table 1: Machine Learning results on 5-fold cross validation.

In table 2, there are presented the results of the algorithms on the ground-truth dataset (test dataset). Again, it can be seen that the best performing algorithm is Light GBM, providing even better results on test dataset than in the training dataset.

| | Logistic Regression | Multinomial NB | SGD | SVM | Random Forest | Light GBM | XGBoost |
|---|---|---|---|---|---|---|---|
| Reviews | **85.33%** | 83.65% | 85.59% | 84.17% | 82.58% | 83.22% | 81.74% |
| Meta + Reviews | **88.19%** | 83.91% | 87.27% | 88.25% | 86.57% | **87.27%** | 86.77% |

Table 2: Machine Learning results on 5-fold cross validation.

In figure 6, it is shown the confusion matrix of Logistic Regression on both only reviews dataset and combined reviews with meta information.

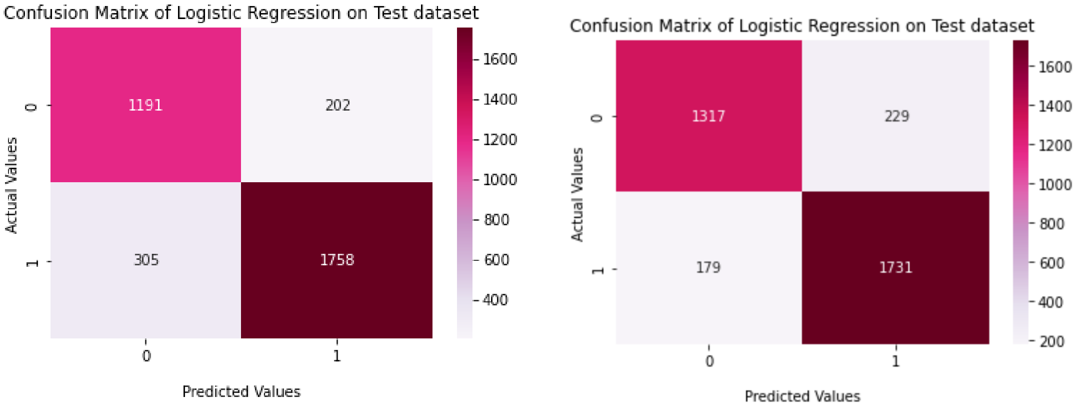


Figure 6: Confusion Matrix of Logistic Regression on Reviews only dataset and Combined Meta Information + Reviews datasets. (Left image - reviews only and Right image - meta + reviews)

## 4.2 Deep Learning techniques

In the table from below (table 3), there are presented the results given by the various Deep Learning techniques. As we can see, we tried to experiment more only with the reviews given in the dataset, obtaining the best results on Bert Embeddings and Bert for Sequence Classification technique (it

| | 3-Layer NN | | Bert Emb + Bert Classif. | | XLM-R + BiGRU | | Word2Vec | | Fast Text | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Train | Val | Train | Val | Train | Val | Train | Val | Train | Val |
| Reviews | - | - | 97% | 86% | **89.52%** | **82.06%** | **93.97%** | **84.35%** | 94.34% | 83.45% |
| Meta + Reviews | **99.31%** | **87%** | - | - | - | - | - | - | - | - |

Table 3: Deep Learning techniques results.

is clearly from the differences between the train and test accuracies) and also, on the Word2Vec model embeddings that were fed into a LSTM model with Early Stopping.

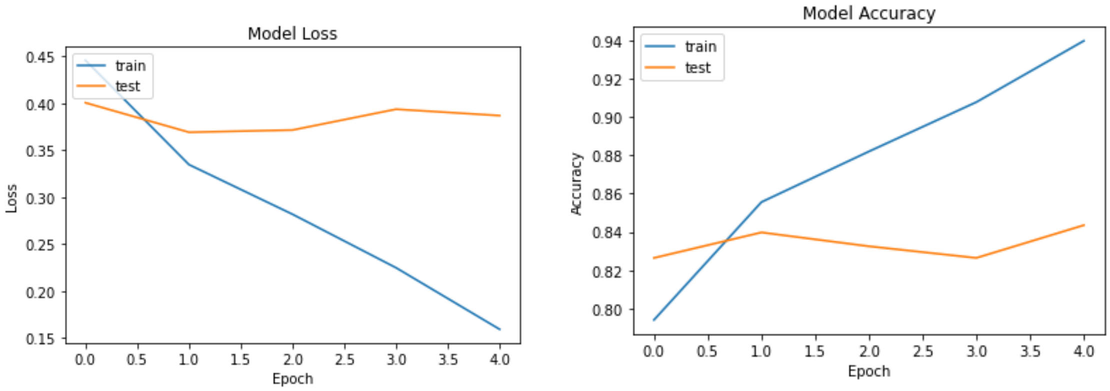


Figure 7: LSTM applied on Word2Vec Embeddings. (Left: Loss on train - test. Right: Accuracy on train - test. - Early Stopping applied)

## 5 Conclusion

In the current project, it was solved the problem of user suggestion predictions based on its Steam game review. There were applied various methods, including the application of feature extraction for reviews like TF-IDF, Bert Embeddings, XLM, followed by classical Machine Learning models to Deep Learning architectures. The enormous potential was given by Light Gradient Boosting Machine and Logistic Regression on the combined meta information (review sentiment, profanity detection) with reviews representations, obtaining approximately 89% accuracy on test dataset. Deep Learning didn't seem to impress on this task, even trying state of the art pretrained embeddings, like Bert Embeddings, XLM-R.

As a potential future work, we can experiment and try the following ideas:

- Dealing with the little imbalance of data (using SMOTE, oversampling & undersampling methods)
- Data augmentation - adding new data containing all of those 2 classes (scraping other Steam reviews)
- Adding other features regarding gaming terminology
- Topic modeling
- Mixup strategies for text classification
- Hierarchical Attention Networks that are used for document classification

# References

[1] Chambers C, Feng Wc, Sahu S, Saha D (2005) *Measurement-based characterization of a collection of online games*, Proceedings of the 5th ACM SIGCOMM conference on Internet Measurement, USENIX Association.