Data Mining Steam Games

By

Christopher Sostre

Thommy Bou

Kevin Markman-Lopez

John Luzada

(2)

In this project, we use data analysis to figure out what genre a game is given a description. The word data is a collection of information made by somebody or someone to be used in a way that would benefit the user.Data analysis on the other hand is the study of data and using it to solve a problem or conflict and evaluate it through different metrics like F measure or mean squared error.

The dataset that our project is based on is a dataset consisting of video games from the Steam library. Steam is a notable gaming hub that many gamers rely on to access all sorts of gaming titles and make the most out of their gaming experience. This dataset details significant features of a Steam game such as rating scores from various video game news websites and their users, median time to complete the game, game difficulty, number of owners, and genre(s) of each game. There are other useful & slightly more complex features such as description with text mining potential.

(3)

The original raw dataset we started with consists of 45 features and 53981 data points. Some of these features such as /gfq\_difficulty require some cleaning before performing any EDA or modeling. Overall, this Steam dataset shows potential for providing meaningful insight into understanding the video game market and for predicting substantial metrics for video games.

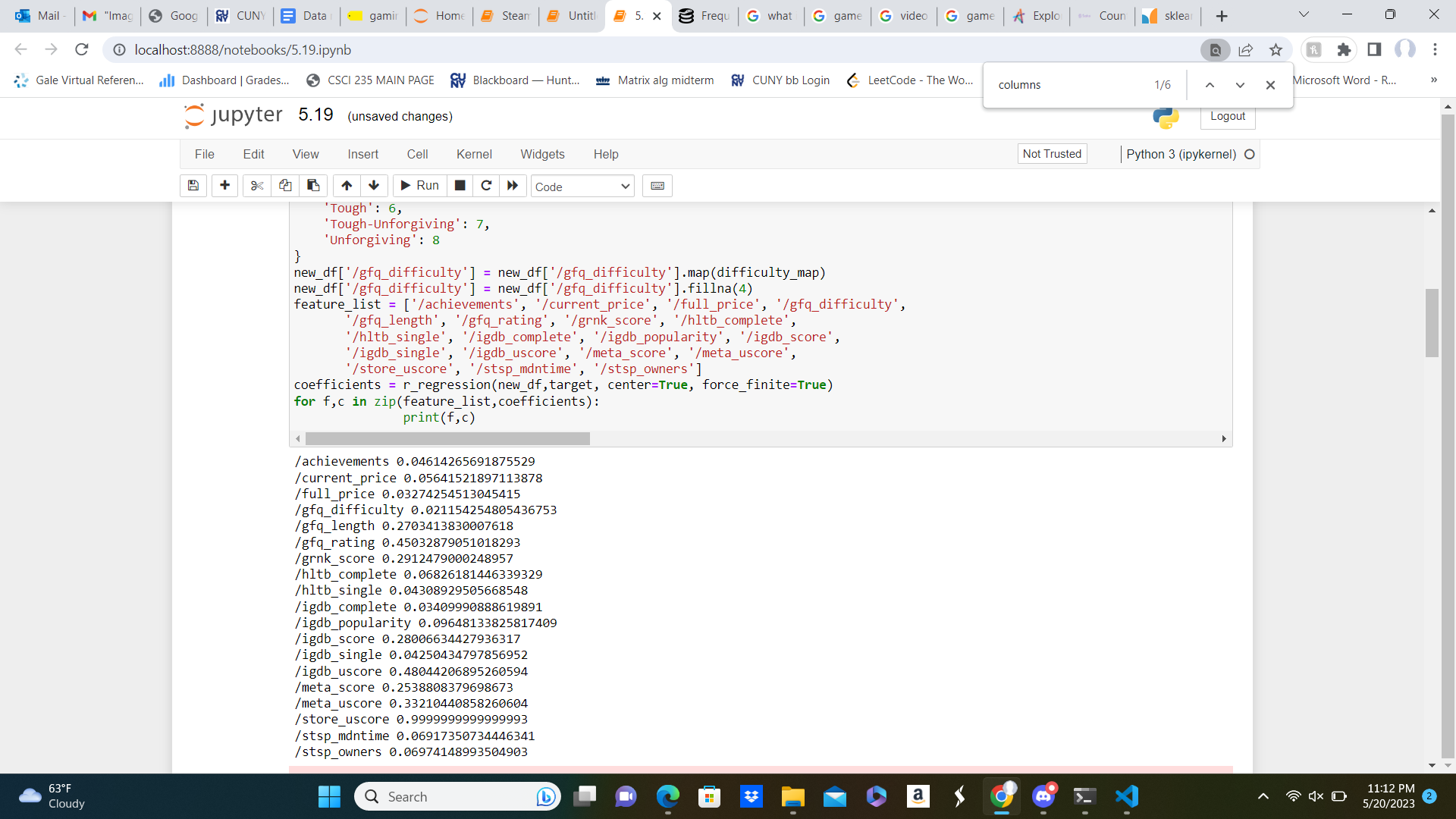
(4)

We generated some visual aids to help us conceptualize the data we were working with. The image on the left is a representation of the frequency of each genre’s appearance in our data set. Somewhat surprisingly the most common genre was “indie”. It turns out that 96% of steam games are classified as indie games. Steam gives developers the ability to put their own games up on the platform, which is why indie is such a common genre.

The fact that there are so many games classified as the genre “indie” is a major reason why such a high percentage of the games in our data set were classified with more than one genre. Moreover, it was even more common for games to have three genres than any other amount of genres. Which certainly made the task of classifying genres more difficult with our model when any given game could be multiple genres at once.

(5)

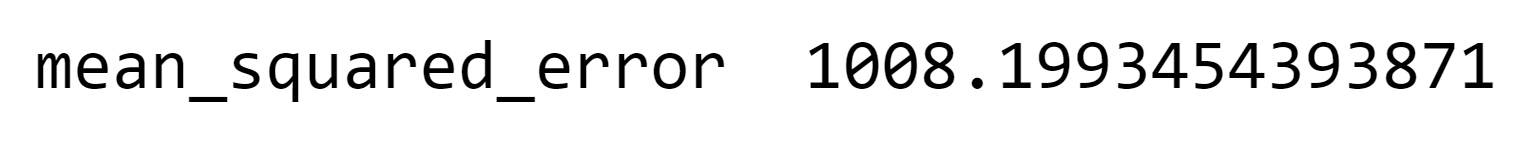
Before doing any sort of model building or data preprocessing, we performed various techniques of EDA to further our understanding of the dataset. We computed Pearson's correlation coefficient on all numerical features on store\_uscore.



Many of our numerical features seem to not hold a strong correlation towards Steam user score, a possible target for our project. This is in large part due to the preprocessing required, nan (not a number) values, and the dataset not being informative towards user score.

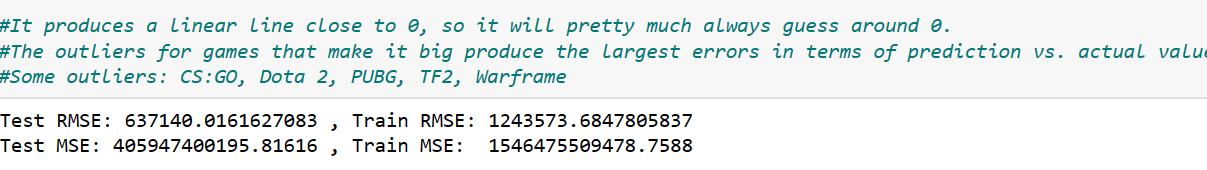
(6)

With all numerical features of the dataset in a data frame, we built our first baseline linear regression model that regressed on the target store\_uscore.



(7)

The attribute /gfq\_diffculty describes the difficulty of a video game within this order from least to greatest ('Simple', 'Simple-Easy', 'Easy', 'Easy-Just Right', 'Just Right', 'Just Right-Tough', 'Tough', 'Tough-Unforgiving', 'Unforgiving'). To input this feature correctly into the model we ordinally encoded each value to map to a score from 0-8. We also imputed nan values to the average difficulty of 4. After doing this and dropping useless columns such as URLs we went forward with more model building but this time to predict on ‘/stsp\_owners’ the number of owners of a game. The results are shown below.



These baseline models performed poorly in predicting substantial metrics of games such as the number of owners and Steam user score. Our dataset required a lot more preprocessing as well as integration of objective feature values.

(8)(9)(10)

Within week 2 and week 3, the next significant step made in preprocessing our dataset was imputing many 0 values in user review scores with the overall average value in each feature. In addition, we created two new columns within our dataset, ‘ Average User Score’ the average value from all five user scores from columns **User Score (GameFAQs)**, **User Score (GRNK), User Score (IGDB), User Score (Metacritic), User Score (Steam).**

(9)

The second column we added in our dataset **User Score Class** is a discretization of **Average User Score** 0-60 falling in the Bad class, 61-80 belonging to the OK class, and 81-100 belonging to the Good class. Columns such as scores from video game reviewers were dropped since there is money bias in these reviews. Also, data points with 0 owners were dropped, reducing our dataset slightly. Searching through other video game datasets for more features with objective metrics we came across a dataset in Kaggle containing the number of mentions a game received in the media as a feature called Presence. This feature had a .18 correlation coefficient with the number of Steam game owners. Since linear correlations were not seen with the features, we wanted to see if there were monotonic relationships between the features and number of steam game owners. We ran Spearman's correlation on numerical features to the number of owners using the SciPy library.

(11)

SignificanceResult represents the Spearman correlation coefficient and p-value represents the probability that given the coefficient the two features are actually uncorrelated. The feature coefficients that stand out are Presence, Medium Time (SteamSpy), Popularity (IGDB), and Game Length (GameFAQs). These coefficient values tell us that the number of owners monotonically increases as the values of the features increase. Surprisingly Average User Score and User Score Class did not hold the same high coefficient. This suggests that games with high review scores from users do not necessarily mean that the game is owned and played by many people.

Even with this newly cleaned and preprocessed dataset the models we continued to build still showed high bias and variance in predicting substantial features of the dataset such as the number of owners and user ratings. Going forward with this project we decided to shift our direction toward predicting the genre of a game by text-mining a potentially useful feature description.

(12)

In this analysis, we used a bag of words which is a natural language processing model that puts words inside of a vector and its count. We employed this technique to analyze a Steam PC game dataset and predict genres based on the description column. As a team, we first preprocessed the data by tokenizing the descriptions into individual words. Next, we created a vocabulary or a "bag" of all unique words presented in the dataset. We then represented each game description as a vector, where each element of the vector corresponded to a word in the vocabulary, and its value indicated the frequency of that word in the description. By converting the textual descriptions into numerical vectors, we were able to apply machine learning algorithms to predict the genre of each game. This approach allowed us to focus on the presence or absence of specific words rather than their order, enabling efficient genre classification. By leveraging the bag-of-words technique, we gained valuable insights into the relationships between game descriptions and genres, facilitating more accurate predictions in the realm of PC gaming.

(13)

We used a Python library called nltk (Natural Language Toolkit) to tokenize the words in the description, and also apply a stopwords filter. Stopwords are commonly used words like "and", "the", or, "is" which do not carry significant meaning for analysis purposes. Next, we used a library called beautiful soup to stem the words in the data set. Stemming is a technique used to reduce words to their base or root form, which can help in consolidating variations of the same word. However, we encountered a challenge where the stemming process resulted in a significant number of misspelled words. We tried to implement a spell checker but found that it wasn’t very effective, so we decided to go with a more brute-force approach. We started to build a separate list of words that contained the misspelled words that were being left behind by the stemmer. This list was combined with the stop words list when building the bag of words to manually remove the misspelled words. As this process went on we realized that this approach was not going to be helpful. The list of misspelled words was getting longer and longer, furthermore, the list was just getting populated with the next most common words every time a word further up in the list was deleted. Meaning we were deleting high-frequency misspelled words and replacing them with just as many lower-frequency misspelled words.

This issue of generating misspelled words can have implications for building models with the resulting bag of words. Misspelled words can introduce noise and inaccuracies in text analysis tasks, potentially affecting the performance and reliability of the models. Models trained on a dataset with a high proportion of misspelled words might struggle to generalize well to new data or provide accurate predictions.

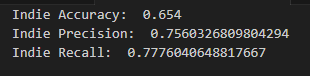
In addition, we improved on our stemming by not stemming words, but rather lemmatizing them. Stemming words gave us words that did not necessarily contribute to being useful. Lemmatization aims to reduce words to their base or dictionary form and taking into consideration the part of speech that word occupies, which can help in standardizing the representation of words. This normalization can be especially useful when dealing with misspelled words because it attempts to transform them to their correct base form.

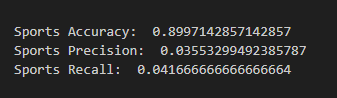
(14)

After preparing our bag of words, we proceeded to find a baseline accuracy for our future models, so we have something to compare to. At first, we found out the percentage occurrence of each genre, squared each percentage, and added it up for a simple baseline accuracy. However, since any game can have any number of genres attached, there is bound to be overlap. That impacted our baseline accuracy by having it be 1.13, meaning we would need to have more than 100% accuracy, which is illogical to have. So instead, we created a dummy feature, randomly consisting of 1s and 0s, and used that as our predictor for the highest occurring genre, which is Indie. The baseline accuracy yielded from this method was 0.5006249289853426, rounded down to 50% accuracy.

(15)

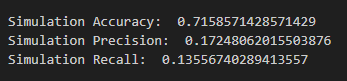
Once we found our baseline accuracy, we went on to use Naive Bayes for our model. As the bag of words and the dataset were uneven with the amount of data points by a couple hundred, we simply took the top 35,000 data points. We then applied our model to the genres and made each of them a different, singular target. However, we omitted the targets that had less than a 5 percent frequency. We found out that the most commonly occurring genres in Indie, Casual, Action, and Adventure had the lowest accuracies, but they also had the most notable precisions and recalls. The least frequent genre in Sports with a 5 percent presence, sported the highest accuracy, while also having the lowest precision and recall, meaning that the model successfully predicted that most games are not of the sports genre, but also gave us many false positives and false negatives.

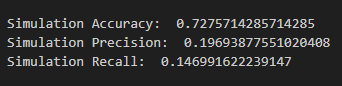




(16)

After implementing our lemmatizer, we tried to improve our model further by tweaking the class priors. By implementing our Naive Bayes model, it assumes the equal probability of each genre, but by tweaking the class priors for each genre, we can correctly assume the frequencies of each genre. However, the newest results were only marginally better in some aspects. For example, accuracy stayed within at least one percent of each other before and after lemmatization. Using the genres mentioned earlier, indie accuracy went up from 0.654 to 0.658, while the sports genre accuracy went from 0.899 to 0.906. The most notable change in precision was seen in the Simulation genre, which saw approximately a 2.5 percent increase in precision, although the general trend is that it stayed the same. For recall, some genres performed worse with lemmatization. To present this change, the free-to-play genre recall decreased from 0.763 to 0.0602, approximately a 1.6 percent decrease and one of the most notable decreases. However, genres such as simulation went from 0.1355 to 0.1469, an increase of more than a singular percent.





Why was this study done? It is because people are interested in what genre a game is. For them, it is best to know the genre beforehand so they can have some idea of whether they may like the game or not. Given that info, it is best to look at the description, datamine the required information, and come up with a valid conclusion that would serve the user and their endeavor.

We believe that although our Naive Bayes model worked well in terms of accuracy, our precision and recall for many of the genres, especially the ones with low presence, were very much lacking. It seems as if the model, for the most part, kept on predicting 0 for the data points that were positive, as well as predicting 1 for a small portion of data points that were negative. Another thing that made our model challenging to train was the fact that any game can have any number of genres attached to them. For example, some games had only one genre attached, while most games had approximately two to three genres on average. This produced an overlap of the frequencies of each genre, causing us to use each genre as its own singular target.

If we were to improve on our model, we could have implemented the AdaBoostClassifier from sklearn.ensemble. What the AdaBoostClassifier does is that it iterates through the samples multiple times, giving more weight to the challenging cases that we have in our dataset. We believe that it will improve our model by emphasizing the much more difficult cases, and it can achieve potentially higher accuracy, while also potentially increasing its precision and recall by reducing the amount of both false positives and false negatives that our model currently gives us for the underrepresented genres.

In conclusion, we have explored the task of video game genre prediction based on a bag of words representation of the given descriptions on its respective store pages. Our Naive Bayes model performed exceptionally well for accuracy, but especially lacked in the departments of precision and recall. Our model could have been enhanced with the boosting ensemble method through the use of the AdaBoostClassifier, which addresses class imbalance. By implementing ensemble methods, our model would have performed significantly better that could more accurately predict genre based on their respective descriptions using a bag of words.