Andrew Riggs

CS376B, Spring 2017

4/13/17

**Predicting Videogame Playtime Through Classification**

**Introduction**

Videogames are becoming an increasingly substantial portion of the entertainment industry, with numerous companies and organizations dealing in the funding, creation, publishing, and marketing of these games. In addition, many of these companies work solely within the realm of videogames, making their main source of income the profits made by videogame sales.

With the rise of more advanced computer hardware, more ambitious games could be created and viably sold to a large consumer base. By steadily increasing the scope and quality of games, expectations for the average game have been rising steadily. A game that may have sold at the top of the market in 2000 will likely now be considered heavily flawed, by the standards of 2017.

The increased expectations of the consumer base presents a new challenge, however. To meet this new quality metric, the companies and studios involved must often drastically increase their monetary investment in a game to allow for the inclusion of new technologies and features expected from a modern product.

In addition to the increase in monetary investment, a greater amount of time must also be allotted to the development of these products to allow for the completion of these new features.

Industry common technologies such as fully acted motion capture, user modification support, fully supported multiplayer interactions, and immersive high-definition graphics are all incredibly consuming in terms of both money and time.

As a result, each of these games becomes a rather substantial investment. The amount of money involved increases the sales figures required for profit, and the amount of time involved drastically limits the number of games that can be put out by a single studio. This combines to make every game a quite significant risk.

One way that videogame creators and publishers have begun to counteract this risk is by creating games that are specifically designed to have a long lifespan. By creating games that will be played for numerous years, and inserting additional repeatable transactions into those games, the risk can be somewhat decreased. In addition, the game is no longer just a single burst of revenue at release, but instead provides a more consistent income.

With this in mind, this project will attempt to predict what kind of attributes are important to creating a game that maintains a high average playtime. Numerous classification models will be created, evaluated, and used to this end.

**Methodology**

This project followed the CRISP-DM project framework. Business understanding was largely covered in the introduction section. As such, there were five main phases.

1. Data Acquisition
2. Data Understanding
3. Data Preparation
4. Data Modelling
5. Model Evaluation

***Data Acquisition***

The data used in this project is from Steam, an online games distribution platform that is essentially the de facto game marketplace of the PC market. To obtain this data, a website called SteamSpy (steamspy.com) was used. This website takes in the raw sales figures and derives from them a range of features, such as average playtime and game tags.

When retrieved from the website, the data was initially in JSON format. After the intervention of Professor Read, the data was converted into a CSV file as well as an Rda file.

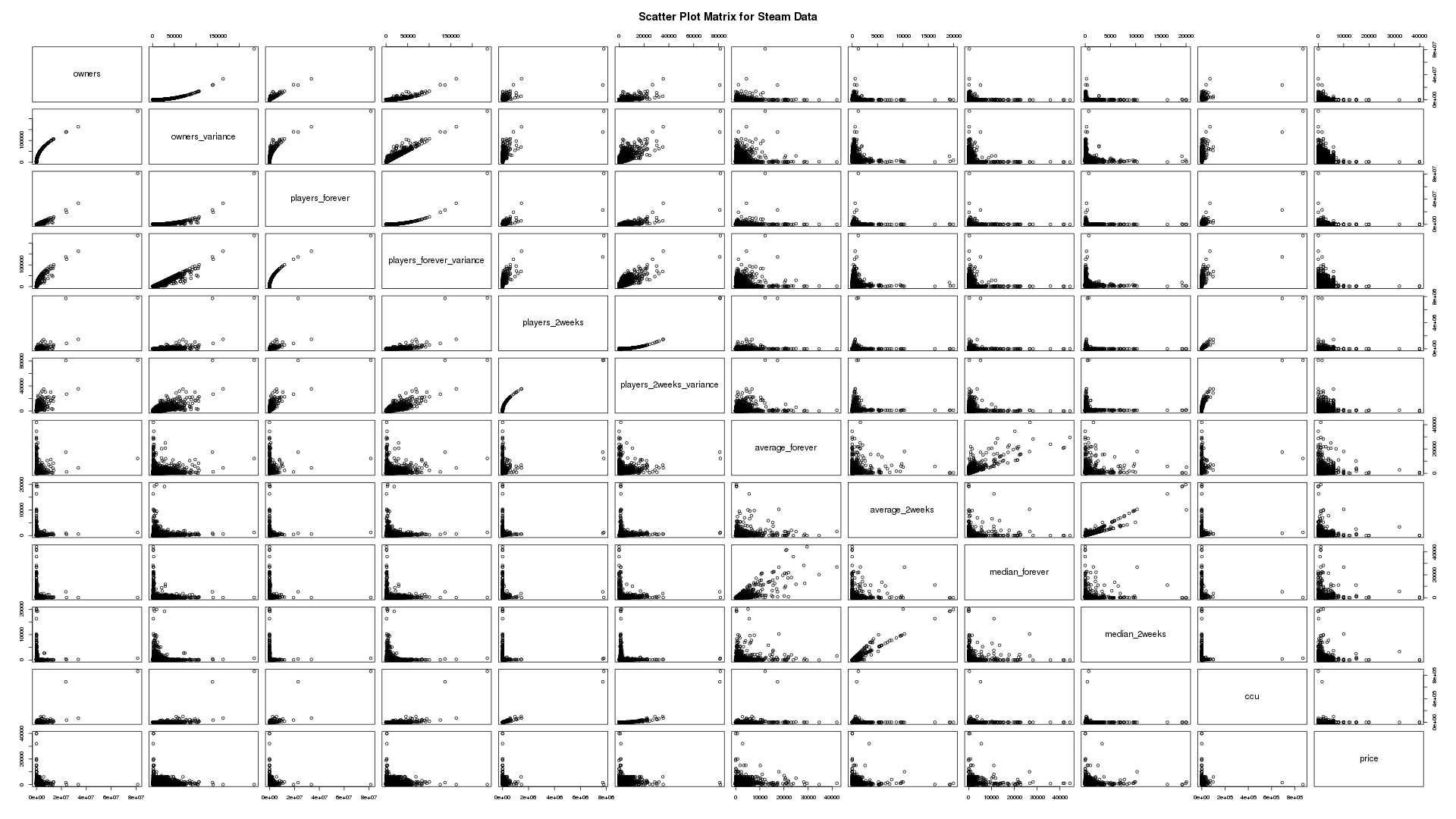


Figure 1- Initial Scatter Plot Matrix of the Dataset

***Data Understanding***

Data exploration occurred via R scripts, using R Studio.

The initial dataset was made up of 11956 instances, with 347 variables each. It was immediately apparent that certain features of the dataset were not going to be particularly relevant, such as the name of each game or the steam-assigned application ID number.

It also quickly became apparent that the target feature (average\_forever) was heavily skewed towards 0. This made a decent amount of sense and likely did not represent an error in the data itself, as steam is a store that anyone can easily publish a product to. There are likely many products on the store that have never been played for any significant amount of time.

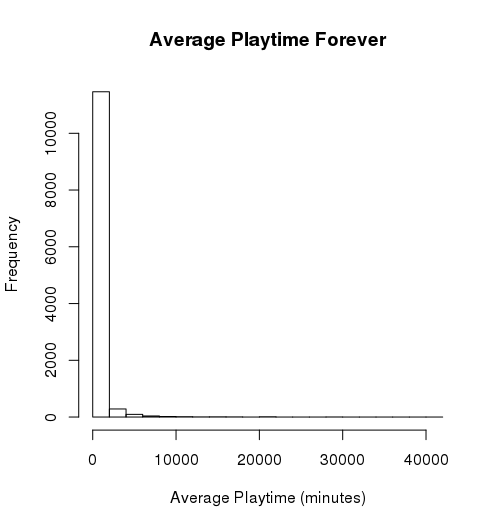


Figure 2- Distribution of Values in average\_forever

In addition to this, there were quite a few descriptive features that were highly correlated with each other. These features seemed to correlate so highly primarily because they were all derived features that described similar aspects of the data.

For example, there was players\_forever and players\_2weeks, describing total number of players for all time and the total number of players for the last two weeks.

These features seemed to be largely irrelevant to the target feature, yet gave a great deal of correlation. These pieces of data were effectively different metrics of how many people were playing the game. As such, they would likely have created an overfit model that was too reliant upon data that was not useful for real world predictions.

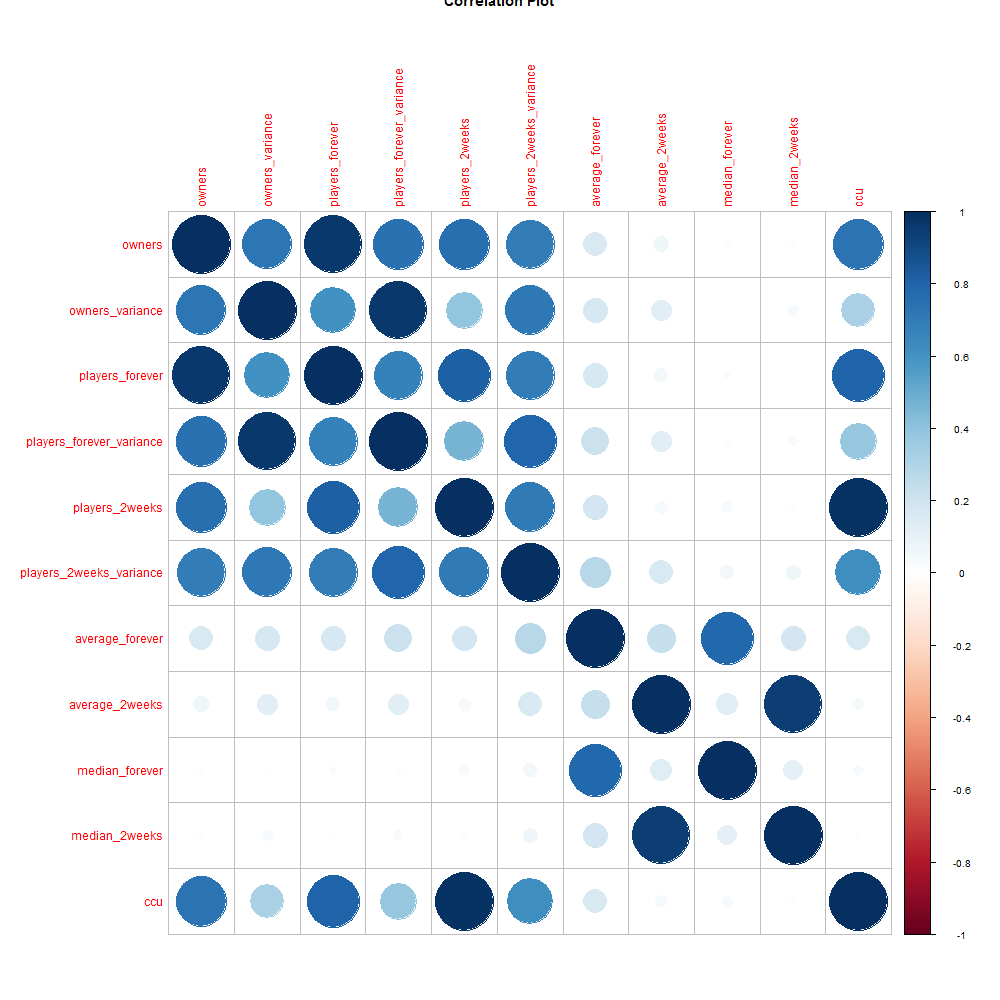


Figure 3- Correlation Plot for the Numeric Non-Tag Features

The majority of features in the dataset represent the tags a game can be tagged with to denote certain gameplay features or genres. Each instance in the feature was either a numeric value denoting the number of times the game was tagged with that label, or NA value.

This format was somewhat problematic, as it brought unnecessary weight to the tag values and left many NA values to contend with.

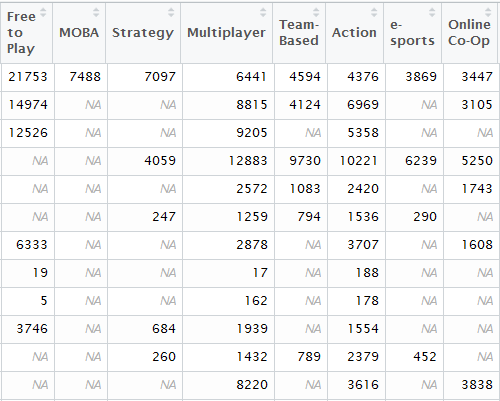


Figure 4- Initial State of the Dataset's Tag Features

As a result of the tag formatting, there was also an excessive number of features, some of which hardly ever actually occurred.

***Data Preparation***

Preparation of the data was also done via R scripts using R Studio.

**Tag Feature Format**

The first change that was made to the data was done to rectify the aforementioned issue with the tag feature format. Had the original format been used, classifiers would likely have been reading into the individual values to too great an extent, leading to deceptive results.

In addition, the presence of NA values would have caused problems for many classifiers later on.

The formatting of the tag features was changed to be a binary value of 1 or 0. The instance would be assigned a 0 for that feature if it previously had a NA value. The instance would be assigned a 1 for that feature if it previously had a positive numeric value.

This produced a feature format that would work much more effectively with the classifiers that would be used later in the project, and avoided the use of missing values.

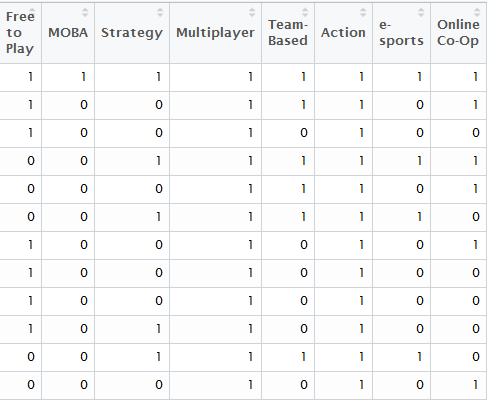


Figure 5- Final State of the Dataset's Tag Features

**Removal of Irrelevant and Highly Correlated Descriptive Features**

This change was necessary to avoid creating a model that was overfitting based on highly correlated descriptive features that held no value in relation to the target feature.

First, irrelevant features that held no bearing on the target feature were removed to avoid misleading classifiers. This included the Steam application ID, and the name of the game itself. Both of these features held values that never repeated, and were only used for game identification, making them completely irrelevant to the target feature.

Next, a single tag feature simply called ‘tag’ was removed from the dataset. This feature held no values, and was simply left over from the conversion process from JSON to CSV.

Following this, the aforementioned highly correlated descriptive features needed to be addressed. As previously stated, these features were all different metrics of how often the game was being bought/played. As a result of this feature similarity, their data was often very highly correlated, as they functioned based on the same fundamental calculations.

To avoid classifiers relying on and overvaluing these irrelevant features, they had to be removed. A total of 10 features were removed from the dataset for this reason. All of these features can be seen in Figure 2, along with the target feature (average\_forever).

**Further Removing of NA values**

Some of the values in the price feature of the dataset had the value of NA, denoting that there was no pricing information available. This did not, however, denote that the game had a price of 0, as there were far more values in the feature which had a value of 0 than NA.

Upon further inspection of these NA values, it became clear that instances that had a price of NA represented products that were not standalone products. These instances were products that had been bundled with other products, and never sold on their own, making pricing information unavailable. Often these were simply tools, patches, or addons for the actual product itself.

Due to this, instances with values of NA in the price feature have been removed from the dataset.

**Type Conversion**

Despite representing numeric values, the price and score\_rank features were nominal initially. To allow for proper comparison and classification, these features were converted from nominal to numeric features.

**Further Pruning of the Dataset**

At this point, the dataset still has 11690 instances with 334 distinct features, with some of these features only occurring a few times. To avoid leading the classifiers astray, the size of the dataset must be reduced.

To do so, the number of times each tag occurred in across the dataset was recorded. If this sum was less than 20, the feature was removed from the dataset.

This managed to remove a total of 48 features from the dataset.

After removing these features, each instance now had to be checked to make sure they were not now devoid of all tags.

To do so, the number of tags applied to each instance was recorded. If this sum was less than 1, the entire instance was removed from the dataset, as it had exclusively been tagged with the now removed features.

This managed to remove 768 instances from the dataset.

As a result of these removals, the dataset now had a total of 10922 instances with 286 distinct features. This size is much more manageable.

**Discretizing the Target Feature and Skewed Data**

The target feature of the dataset is initially numeric. While this may work with some models, many classifiers cannot use numeric targets.

To rectify this, the target feature must be discretized. This will be done using the equal frequency binning.

Equal frequency binning will be used due to the aforementioned skew towards 0 in the dataset’s target feature.

10 bins were used to contain the data.

***Data Modelling***

With the modifications to the dataset complete, numerous classifiers could now be run on the dataset. For these classifiers, the dataset was split into training and test sets. Two thirds of the dataset went to making the training set, while one third of the dataset went to making the testing set.

Modelling was done with a selection of five classifiers, with the implementation of each described below.

**Naïve Bayes**

The binned dataset was used for the implementation of Naïve Bayes. This model was created using the e1071 package for R.

**C4.5**

The binned dataset was used for the implementation of C4.5. This model was created using the RWeka package for R.

**RIPPER**

JRIP rules:

===========

(Singleplayer <= 0) and (VR Only >= 1) and (Action <= 0) and (price >= 499) and (Casual >= 1) => average\_forever=[ 21, 65) (29.0/12.0)

(price <= 899) and (VR Only >= 1) => average\_forever=[ 0, 21) (188.0/79.0)

(score\_rank <= 8) and (Strategy <= 0) and (price <= 59) and (Massively Multiplayer <= 0) and (Indie >= 1) => average\_forever=[ 0, 21) (23.0/7.0)

(Singleplayer >= 1) and (score\_rank >= 59) and (price >= 1599) and (Adventure >= 1) and (Difficult >= 1) => average\_forever=[480, 952) (28.0/11.0)

(Multiplayer >= 1) and (price >= 3999) => average\_forever=[952,41947] (68.0/19.0)

(price >= 1674) and (Sandbox >= 1) => average\_forever=[952,41947] (116.0/54.0)

(Indie <= 0) and (price >= 1499) and (Simulation >= 1) and (score\_rank >= 41) => average\_forever=[952,41947] (57.0/19.0)

(Indie <= 0) and (price >= 999) and (Strategy >= 1) and (Multiplayer >= 1) => average\_forever=[952,41947] (94.0/42.0)

(Indie <= 0) and (price >= 999) and (Turn-Based >= 1) and (score\_rank >= 70) => average\_forever=[952,41947] (19.0/4.0)

(Indie <= 0) and (Action <= 0) and (price >= 5999) => average\_forever=[952,41947] (18.0/7.0)

(Indie <= 0) and (price >= 999) and (Strategy >= 1) and (score\_rank >= 16) and (Turn-Based Strategy >= 1) => average\_forever=[952,41947] (18.0/7.0)

(Indie <= 0) and (Multiplayer >= 1) and (MMORPG >= 1) and (Third Person >= 1) => average\_forever=[952,41947] (13.0/3.0)

=> average\_forever=[164, 203) (5835.0/5182.0)

Number of Rules : 13

The binned dataset was used for the implementation of RIPPER. This model was created using the RWeka package for R. The model ended up having a total of 13 rules, and favored heavily classifying via the use of price, score rank, and the more commonly occurring tags.

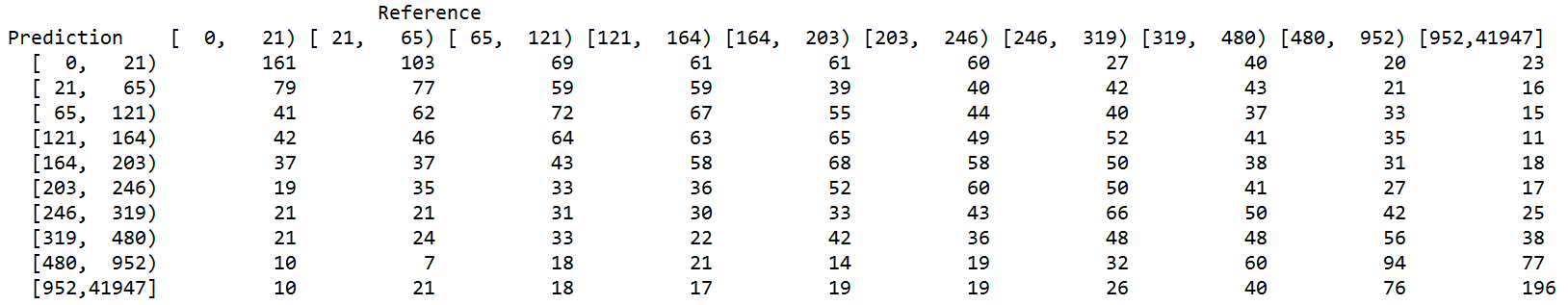


Figure 6- Confusion Matrix for C4.5

**Random Tree**

The regular dataset was used for this implementation of Random Tree. This model was created using the Weka KnowledgeFlow environment.

The tree that was created is rather large and unwieldy, with the size of the tree totaling to 12185.

**PART**

The binned dataset was used for this implementation of PART. This model was created using the Weka KnowledgeFlow environment. The total number of rules generated by the model was 1791.

***Model Evaluation***

*\*\*Naïve Bayes unexpectedly broke at time of writing, will try to fix as soon as possible.\*\**

After each model was created with the training set, they were then tested against the testing set. All testing of the models occurred in the environments they were created and trained in.

**C4.5**

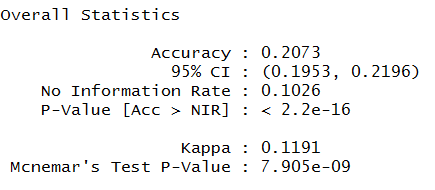


Figure 7- Statistics for C4.5

The overall accuracy for the C4.5 implementation was 20.73%. This is rather low, and denotes that the model is not able to properly predict the class of the target feature.

In the context of this project, this means that the model fails to predict the playtime of the game.

This is also echoed in the incredibly low kappa statistic found in Figure 7. With a kappa statistic of only 0.1191, the model is hardly any better than guessing.

In addition, the confusion matrix aids in this assessment of the model, as the model seems to be spreading its predictions rather evenly. There are some clusters of correct guesses in both the lowest and the highest value bins, but for everything in between the predictions seem to be nearly random in nature.

Overall, this model proves to be ineffective in determining the average playtime of a game via the descriptive features. As denoted by the low accuracy, kappa statistic, and suspicious confusion matrix, this model is effectively useless.

**RIPPER**

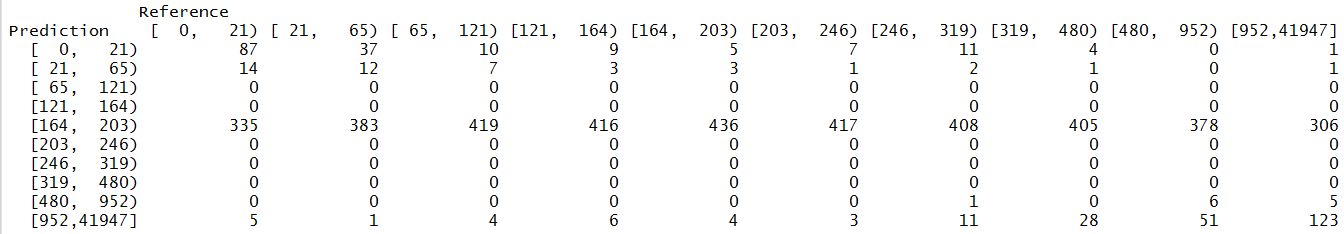


Figure 8-Confusion Matrix for RIPPER

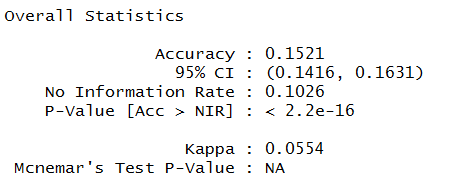


Figure 4- Statistics for RIPPER

The overall accuracy for RIPPER was 15.21%. This is even lower than the previous model, making this implementation of RIPPER another entirely unviable model.

In addition to the lower accuracy, the kappa statistic for this model is also lower. Here, the kappa statistic is only 0.0554, indicating that the model is effectively guessing randomly.

When looking at the confusion matrix, the problem with the model quickly becomes apparent. While there are some values distributed across the entire possible range, the vast majority have simply been classified as belonging to a single grouping.

This indicates that the model is not at all effective in predicting the average playtime of a game, and is instead classifying nearly the entire dataset into a single class. What little accuracy this model has can be explained by sheer luck, and the presence of values that actually do belong to that single class.

**Random Tree**

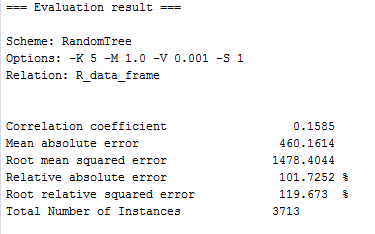


Figure 9- Statistics for Random Tree

The Random Tree model follows the project’s trend of not being a particularly useful model for predictive classification.

Both the mean absolute error and the root mean squared error are rather high, at 460.1614 and 1478.4044, respectively. This implies that the error rate of the tree is rather high, and cannot be relied upon to properly classify instances of the testing set.

In addition, the relative absolute error and root relative squared error are also remarkably high, giving further evidence that the model cannot properly classify the data.

**PART**

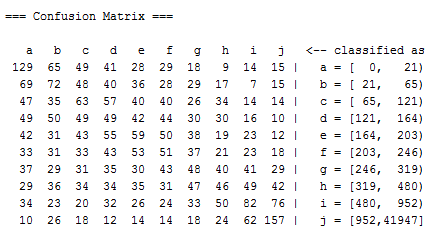


Figure 10- Confusion Matrix for PART

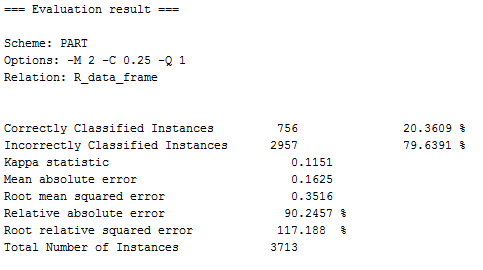


Figure 11- Statistics for PART

The overall accuracy of this implementation of PART is 20.3609%, still an incredibly low number. This would imply that this model is also useless for predicting the average playtime of a game.

The kappa statistic would back this up, with a value of 0.1151. This continues to point towards the implication that the model is not much better, if at all better, than guessing randomly.

The relative absolute error and relative root squared error continue to point towards this conclusion. They are both rather high, and seem to imply the model is very likely to commit errors when attempting to predict and instance’s classification.

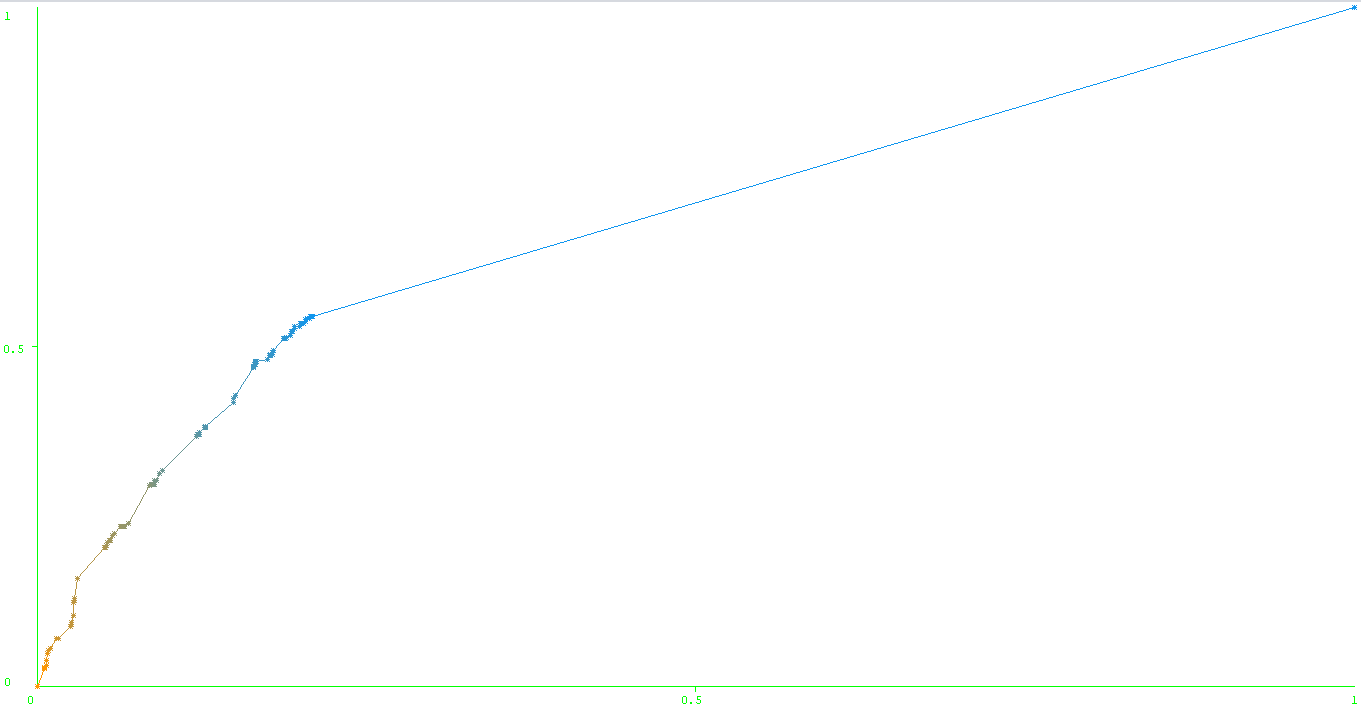
The confusion matrix further supports this, with the data once again seemingly spread across the various classes. While there are a decent number of correct classifications per class, there are not nearly enough for the model to be considered useful. Any correct classifications are still heavily outweighed by the sheer number of incorrect classifications. Once again, the model appears to be no better than randomly guessing. 

Figure 10-ROC Curve for PART

The ROC curve produced by evaluating the model is not particularly hopeful, either. While the graph begins well, it swiftly starts to move back towards the center line. While not a perfect 45 degree line, the ROC curve also makes clear that the model is not much better than guessing.

**Conclusions**

The original goal of this project was to find a way to predict how long a game would be played after purchase, based upon certain aspects of the game itself.

Determining the average playtime per owner of a game would assist in determining what types of games would have the most longevity, and therefore potentially create the most consistent revenue.

To that end, data was extracted from Steam using the SteamSpy API, converted into a usable format, and submitted to the CRISP-DM process to attempt to create viable models.

Numerous issues with the data were found in the data understanding phase, and these issues were addressed in the data preparation phase.

From here, the completed data was used to create, train, and test models in an attempt to predict the longevity of a game product.

Unfortunately, the project has failed to determine a method of doing so at this time. As can be seen from the evaluation of the created models, there was no method used that managed to create a model that could reliably determine how long a game would, on average, be played for.

The models themselves never showed themselves to be any better than randomly guessing, They all produced remarkably low accuracy rates, and very high error rates. In addition, a kappa score high enough to be considered better than guessing was never produced by any of the models.

This, combined with the consistently high mean absolute error and root mean squared error, points to the unfortunate conclusion that no models were produced capable of predicting a game’s longevity properly.

There could be many reasons for this, though the two most likely reasons are experimental error and a lack of an underlying relationship within the data.

It is entirely possible, and in fact likely, that this project has been carried out in a way that was not optimal. Time constraints and inexperience in the field of data science make the possibility of user error a very real consideration.

It is also possible there is simply no underlying relationship to be found in the dataset. This is not to say there aren’t factors that play into the longevity of a game, for there almost certainly are. This dataset, however, was largely limited to three key factors: price, score ranking, and the various tags that had been applied to a game.

It is possible that none of these factors have a sizable impact on the longevity of a game, and this is reflected in the lack of suitable models.

To improve in this regard, more extensive data on the development of a game might be required. Factors such as total budget, studio team size, development cycle length, technologies used, etc. may help to create a more complete picture of what a game is actually made up of. This, in turn, may potentially lead to suitable models.