THE GANG STEALS THE SCRIPT

An eSports Data Analysis

Shane Fuller

ESPORTS INTRODUCTION

- An esport is a muiltiplayer video game played competitively for spectators
- In its annual report, the marketing researcher is predicting 2018 revenues to hit $905 million, a staggering 38% increase from 2017's $655 million
- Twitch, the primary streaming website for competitive esports, receives about 15 million daily active users, and shows hundreds of different games for people to spectate

COMPETITIVE MELEE

- Personally, my game of interest in the esports community, is competitive Super Smash Brothers Melee
- Established in in 2001 for the Gamecube, the game is a Nintendo product in which the goal is to knock the opponents off of the stage
- The competitive community established in the early 2000's, and there have been about 2.8 million dollars in earnings from 2,431 major tournaments

https://www.youtube.com/watch?v=QIZfapuj9Kg&list=PLjqjFqIMEtWeXI7UinpWNdjKIB7MXsJ73&index=21

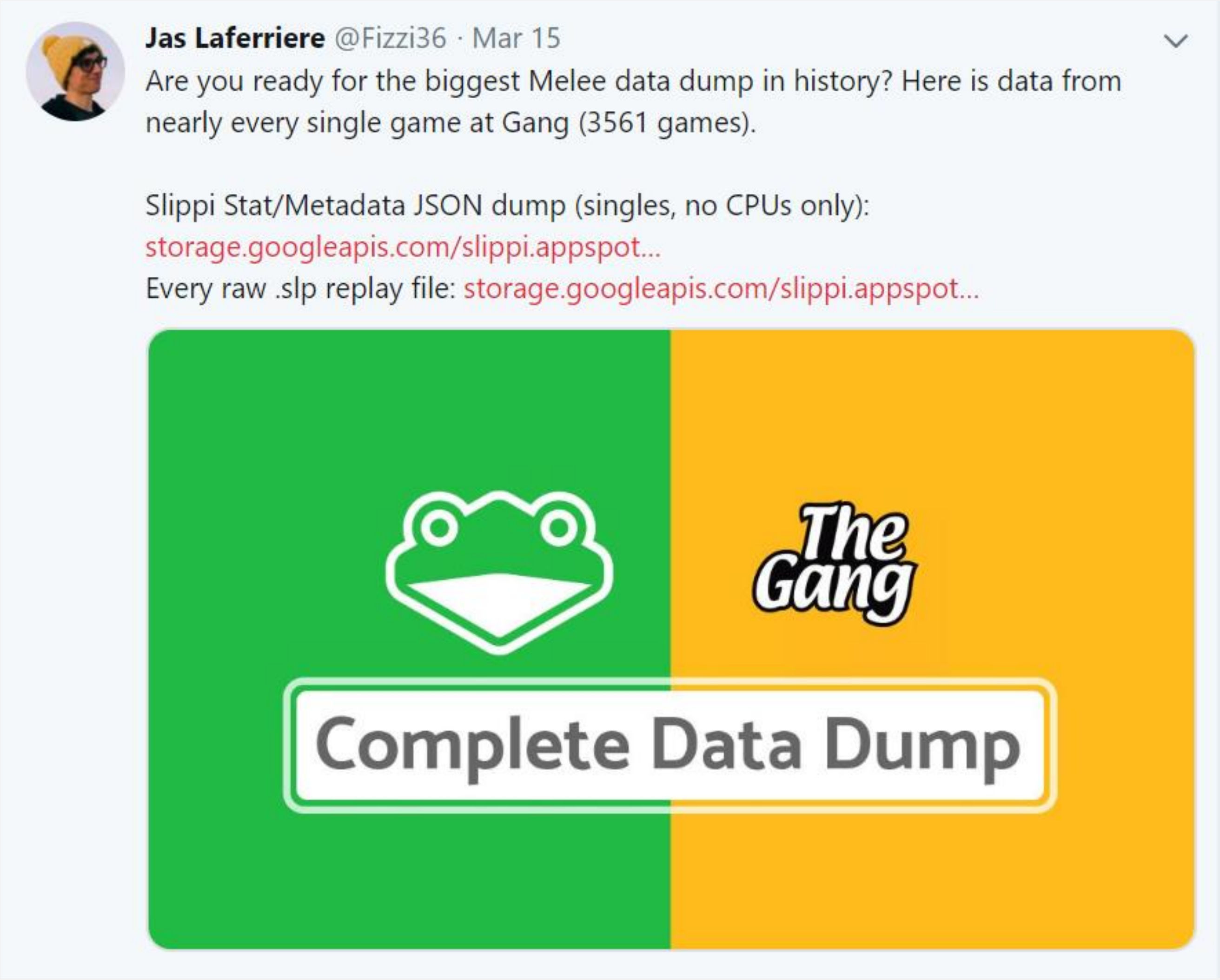
PROJECT SLIPPI

- Project Slippi at its core is a Melee Data Framework
- Announced in early 2015, the project has been primarily developed in and implemented this year
- Provides analysis, visualizations, and replays for every game played at a tournament

https://slippi.gg/

DATA COLLECTION

- The data was collected from Project Slippi at the tournament The Gang Steals the Script
- Collected from a network of 25 consoles throughout the venue
- Matches are both tournament and friendlies, and range in skill



Jas Laferriere @Fizzi36 · Mar 15

Are you ready for the biggest Melee data dump in history? Here is data from nearly every single game at Gang (3561 games).

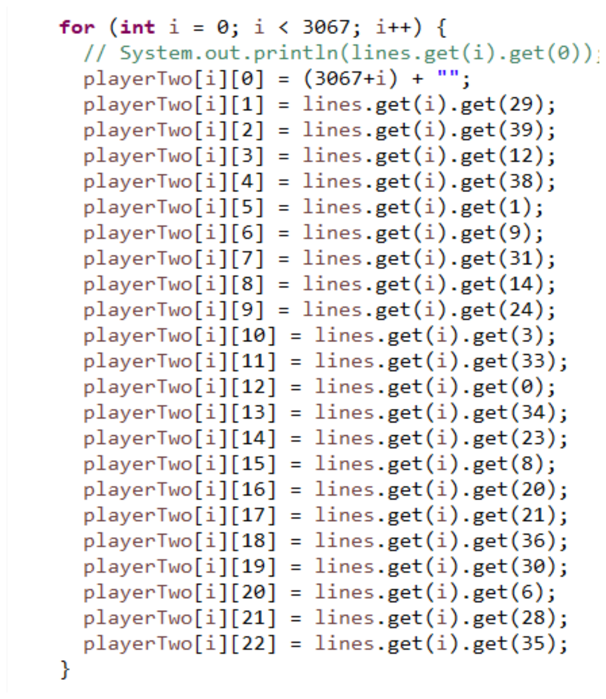
Slippi Stat/Metadata JSON dump (singles, no CPUs only): storage.googleapis.com/slippi.appspot...
Every raw .slp replay file: storage.googleapis.com/slippi.appspot...

DATA UNDERSTANDING

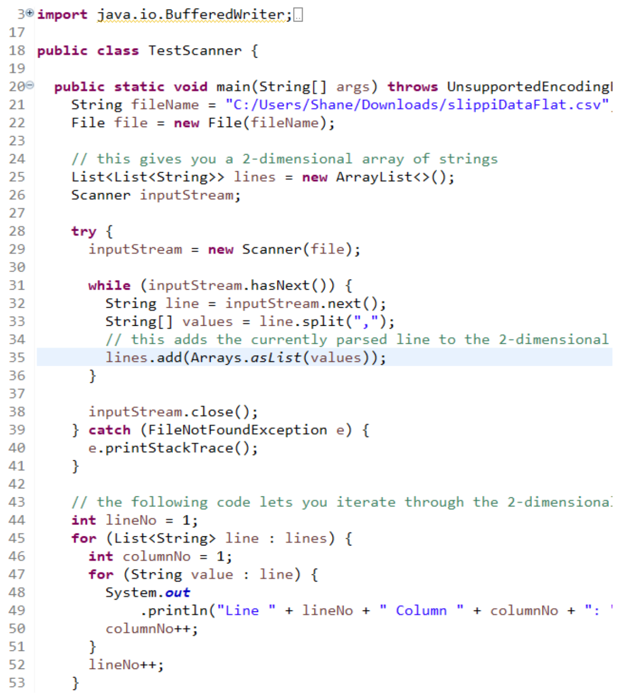
- First step was to review the format of a .slp file, which is a replay of all the inputs completed by both players in a game
- Understand the different documents within the JSON document, and what each subheading information held
- Headings included Root, Stocks, Conversions, Moves, ActionCounts, Overall, SuccessfulConversions, InputsPerMinute, etc.
- After understanding each feature in the file, the next step was to find the variables that I felt impacted the players win condition most heavily

DATA PREPARATION

- Converted the JSON data dump into flat CSV files, extracting selected features
- Wrote Java script to distinguish the stats for each individual player
- Renamed the column headings and added columns to maintain data after reformatting
- Imported into R for cleaning step, included deleting rows with only N/A values, deleting unhelpful or confusing columns, and converting the values to numeric types from character types
- This led to the final Analytics Base Table, gangScriptData.csv



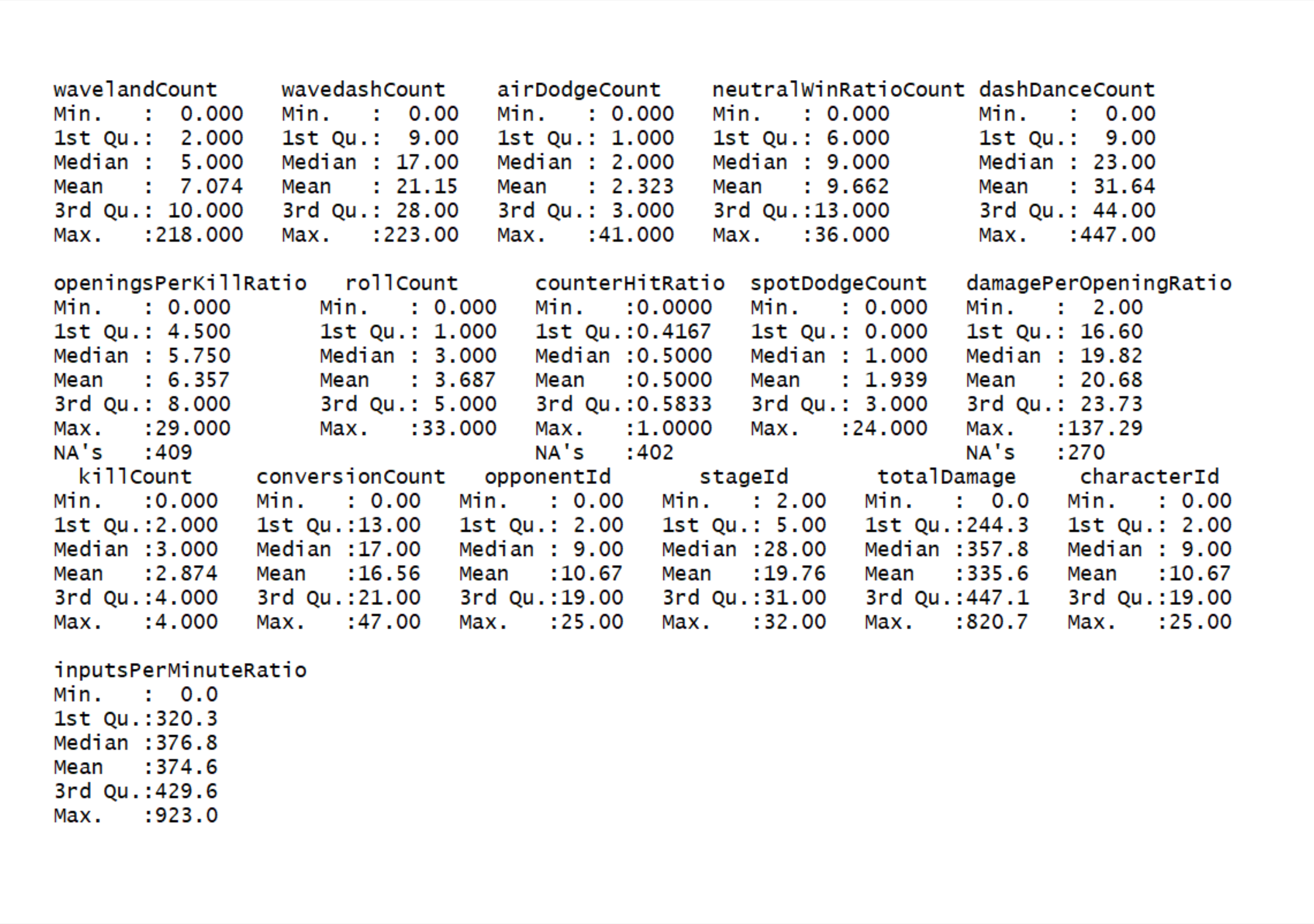
```
for (int i = 0; i < 3067; i++) {
  // System.out.println(lines.get(i).get(0));
  playerTwo[i][0] = (3067+i) + "";
  playerTwo[i][1] = lines.get(i).get(29);
  playerTwo[i][2] = lines.get(i).get(39);
  playerTwo[i][3] = lines.get(i).get(12);
  playerTwo[i][4] = lines.get(i).get(38);
  playerTwo[i][5] = lines.get(i).get(1);
  playerTwo[i][6] = lines.get(i).get(9);
  playerTwo[i][7] = lines.get(i).get(31);
  playerTwo[i][8] = lines.get(i).get(14);
  playerTwo[i][9] = lines.get(i).get(24);
  playerTwo[i][10] = lines.get(i).get(3);
  playerTwo[i][11] = lines.get(i).get(33);
  playerTwo[i][12] = lines.get(i).get(0);
  playerTwo[i][13] = lines.get(i).get(34);
  playerTwo[i][14] = lines.get(i).get(23);
  playerTwo[i][15] = lines.get(i).get(8);
  playerTwo[i][16] = lines.get(i).get(20);
  playerTwo[i][17] = lines.get(i).get(21);
  playerTwo[i][18] = lines.get(i).get(36);
  playerTwo[i][19] = lines.get(i).get(30);
  playerTwo[i][20] = lines.get(i).get(6);
  playerTwo[i][21] = lines.get(i).get(28);
  playerTwo[i][22] = lines.get(i).get(35);
}
```



```
import java.io.BufferedWriter;

public class TestScanner {

  public static void main(String[] args) throws UnsupportedEncoding
    String fileName = "C:/Users/Shane/Downloads/slippiDataFlat.csv"
    File file = new File(fileName);

    // this gives you a 2-dimensional array of strings
    List<List<String>> lines = new ArrayList<>();
    Scanner inputStream;

    try {
      inputStream = new Scanner(file);

      while (inputStream.hasNext()) {
        String line = inputStream.next();
        String[] values = line.split(",");
        // this adds the currently parsed line to the 2-dimensional
        lines.add(Arrays.asList(values));
      }

      inputStream.close();
    } catch (FileNotFoundException e) {
      e.printStackTrace();
    }

    // the following code lets you iterate through the 2-dimensiona
    int lineNo = 1;
    for (List<String> line : lines) {
      int columnNo = 1;
      for (String value : line) {
        System.out
            .println("Line " + lineNo + " Column " + columnNo + ": 
        columnNo++;
      }
      lineNo++;
    }
```

ANALYTICS BASE TABLE

- 17 columns wide
- 6,131 rows deep
- 14 numeric values
  - 4 ratios
  - 10 integer counts
- 3 Identifiers
  - Stage Id
  - Character Id
  - Opponent Id

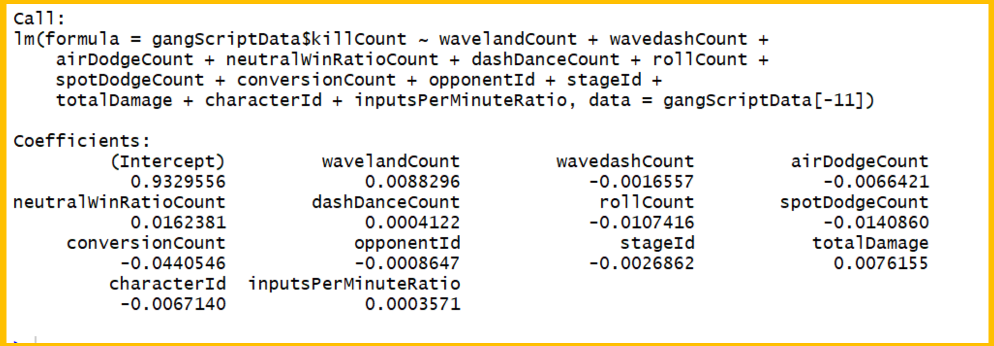


```
wavelandCount     wavedashCount    airDodgeCount    neutralWinRatioCount dashDanceCount
Min.   :  0.000   Min.   :  0.00   Min.   : 0.000   Min.   : 0.000       Min.   :  0.00
1st Qu.:  2.000   1st Qu.:  9.00   1st Qu.: 1.000   1st Qu.: 6.000       1st Qu.:  9.00
Median :  5.000   Median : 17.00   Median : 2.000   Median : 9.000       Median : 23.00
Mean   :  7.074   Mean   : 21.15   Mean   : 2.323   Mean   : 9.662       Mean   : 31.64
3rd Qu.: 10.000   3rd Qu.: 28.00   3rd Qu.: 3.000   3rd Qu.:13.000       3rd Qu.: 44.00
Max.   :218.000   Max.   :223.00   Max.   :41.000   Max.   :36.000       Max.   :447.00

openingsPerKillRatio   rollCount      counterHitRatio  spotDodgeCount   damagePerOpeningRatio
Min.   : 0.000       Min.   : 0.000   Min.   :0.0000   Min.   : 0.000   Min.   :  2.00
1st Qu.: 4.500       1st Qu.: 1.000   1st Qu.:0.4167   1st Qu.: 0.000   1st Qu.: 16.60
Median : 5.750       Median : 3.000   Median :0.5000   Median : 1.000   Median : 19.82
Mean   : 6.357       Mean   : 3.687   Mean   :0.5000   Mean   : 1.939   Mean   : 20.68
3rd Qu.: 8.000       3rd Qu.: 5.000   3rd Qu.:0.5833   3rd Qu.: 3.000   3rd Qu.: 23.73
Max.   :29.000       Max.   :33.000   Max.   :1.0000   Max.   :24.000   Max.   :137.29
NA's   :409                           NA's   :402                       NA's   :270
  killCount     conversionCount   opponentId       stageId       totalDamage     characterId
Min.   :0.000   Min.   : 0.00   Min.   : 0.00   Min.   : 2.00   Min.   :  0.0   Min.   : 0.00
1st Qu.:2.000   1st Qu.:13.00   1st Qu.: 2.00   1st Qu.: 5.00   1st Qu.:244.3   1st Qu.: 2.00
Median :3.000   Median :17.00   Median : 9.00   Median :28.00   Median :357.8   Median : 9.00
Mean   :2.874   Mean   :16.56   Mean   :10.67   Mean   :19.76   Mean   :335.6   Mean   :10.67
3rd Qu.:4.000   3rd Qu.:21.00   3rd Qu.:19.00   3rd Qu.:31.00   3rd Qu.:447.1   3rd Qu.:19.00
Max.   :4.000   Max.   :47.00   Max.   :25.00   Max.   :32.00   Max.   :820.7   Max.   :25.00

inputsPerMinuteRatio
Min.   :  0.0
1st Qu.:320.3
Median :376.8
Mean   :374.6
3rd Qu.:429.6
Max.   :923.0
```

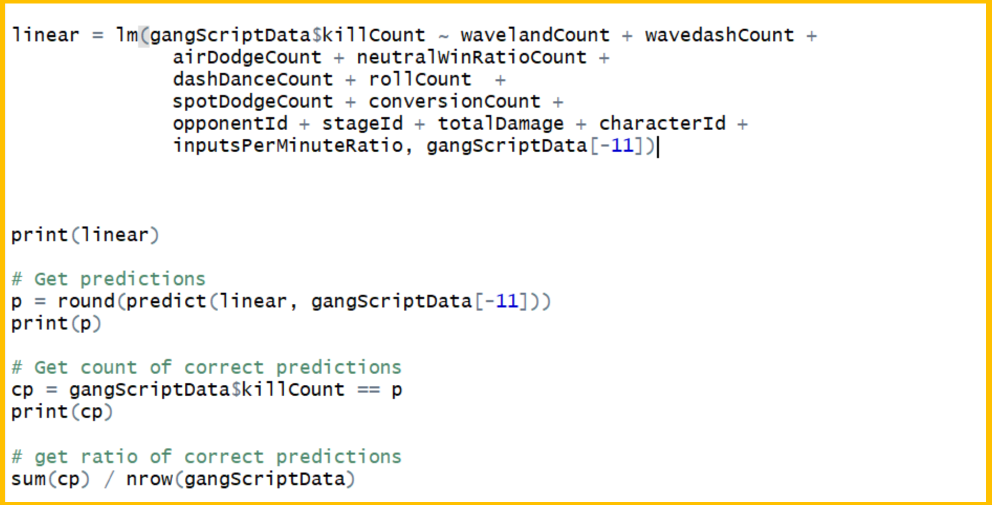
MODELING

- Linear Model
- OneR Model
- Naïve Bayes Model
- Decision Tree Model
- Rules Model

LINEAR MODEL



```
Call:
lm(formula = gangScriptData$killCount ~ wavelandCount + wavedashCount + 
    airDodgeCount + neutralWinRatioCount + dashDanceCount + rollCount + 
    spotDodgeCount + conversionCount + opponentId + stageId + 
    totalDamage + characterId + inputsPerMinuteRatio, data = gangScriptData[-11])

Coefficients:
         (Intercept)         wavelandCount         wavedashCount         airDodgeCount  
           0.9329556             0.0088296            -0.0016557            -0.0066421  
neutralWinRatioCount        dashDanceCount             rollCount        spotDodgeCount  
           0.0162381             0.0004122            -0.0107416            -0.0140860  
     conversionCount            opponentId               stageId           totalDamage  
          -0.0440546            -0.0008647            -0.0026862             0.0076155  
         characterId  inputsPerMinuteRatio  
          -0.0067140             0.0003571  
```

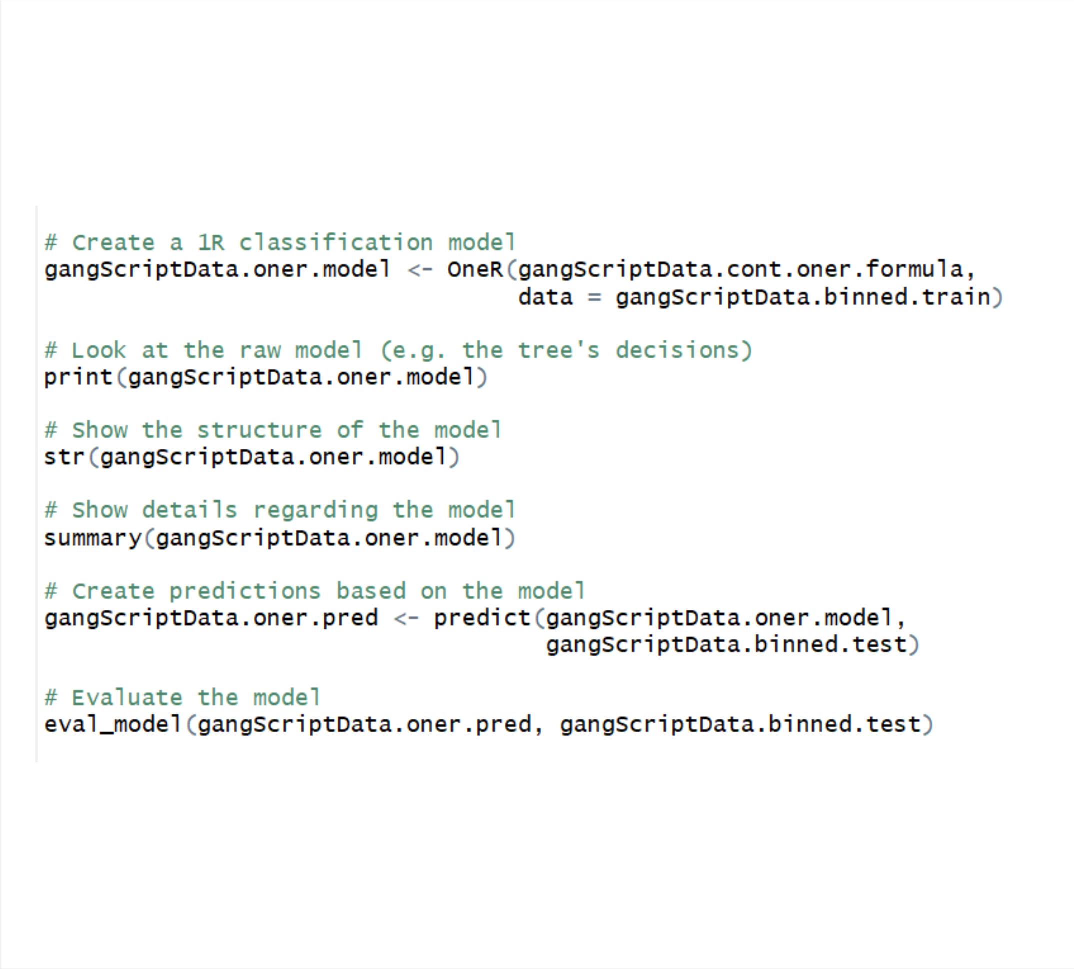


```
linear = lm(gangScriptData$killCount ~ wavelandCount + wavedashCount +
               airDodgeCount + neutralWinRatioCount +
               dashDanceCount + rollCount  +
               spotDodgeCount + conversionCount +
               opponentId + stageId + totalDamage + characterId +
               inputsPerMinuteRatio, gangScriptData[-11])


print(linear)

# Get predictions
p = round(predict(linear, gangScriptData[-11]))
print(p)

# Get count of correct predictions
cp = gangScriptData$killCount == p
print(cp)

# get ratio of correct predictions
sum(cp) / nrow(gangScriptData)
```

- The linear model was the first model that was run
- Surprisingly not a bad success rate, as the model had an accuracy of about 43.1%
- One reason for success is that 45% of the data has a target killCount of 4, so the model was more prone to guess 4
- Intercept suggests that without these movement features, the player is expected to get about 1 kill a game (0.933)

ONER MODEL

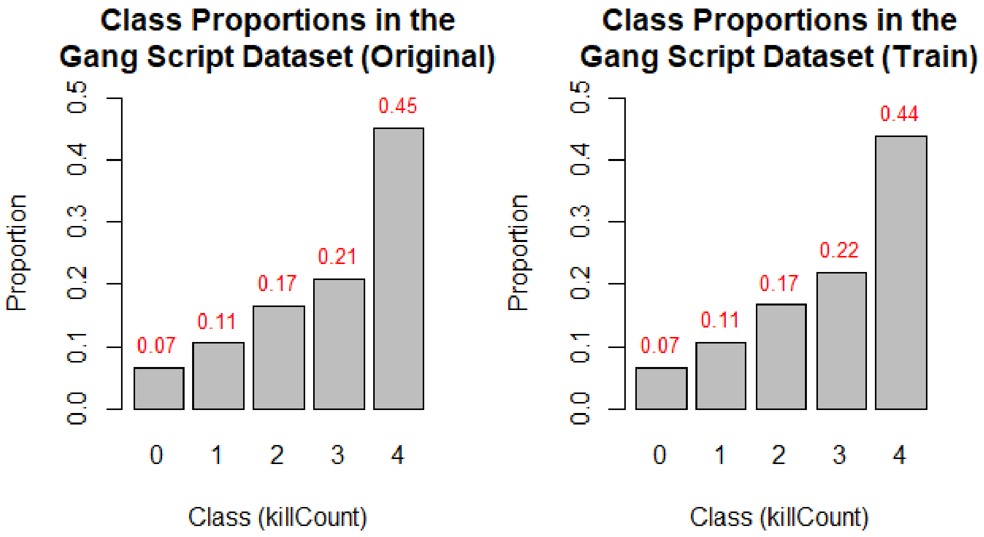
- Supervised binning was performed to reduce the risk of overfitting
- Due to nature of OneR Model, severe overfitting still occurred
- The model memorized the training set, and set on the feature totalDamage to predict the overall killCount
- The accuracy resulted in 0%, which was expected with this model



```
# Create a 1R classification model
gangScriptData.oner.model <- OneR(gangScriptData.cont.oner.formula,
                                  data = gangScriptData.binned.train)

# Look at the raw model (e.g. the tree's decisions)
print(gangScriptData.oner.model)

# Show the structure of the model
str(gangScriptData.oner.model)

# Show details regarding the model
summary(gangScriptData.oner.model)

# Create predictions based on the model
gangScriptData.oner.pred <- predict(gangScriptData.oner.model,
                                    gangScriptData.binned.test)

# Evaluate the model
eval_model(gangScriptData.oner.pred, gangScriptData.binned.test)
```

NAÏVE BAYES MODEL

- Supervised binning was performed to reduce the risk of overfitting
- Problems in the model came with unproper binning, however the data presented would not be successful with a Naïve Bayes model anyways
- Accuracy resulted in 0%

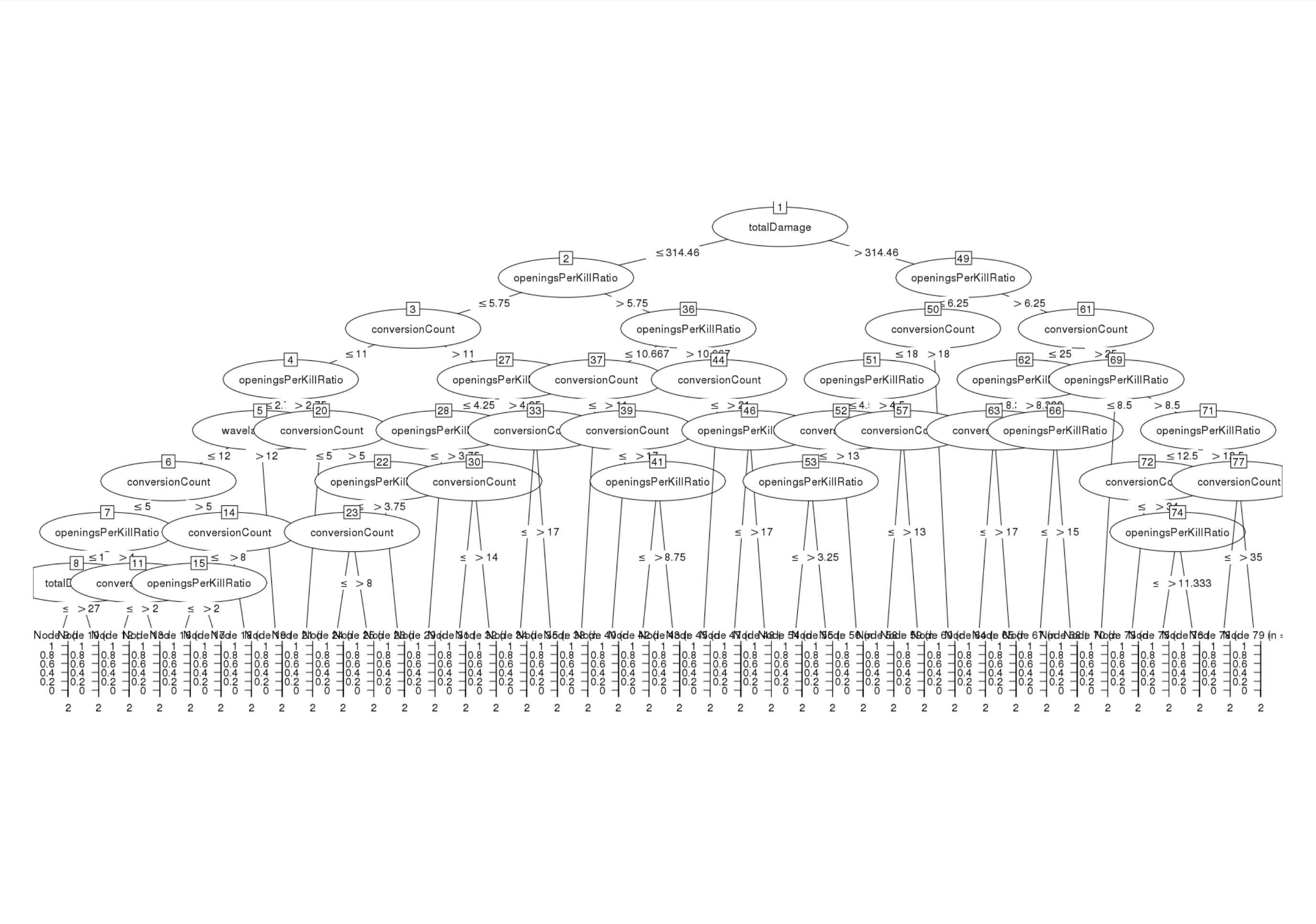


Class Proportions in the Gang Script Dataset (Original)

Class Proportions in the Gang Script Dataset (Train)

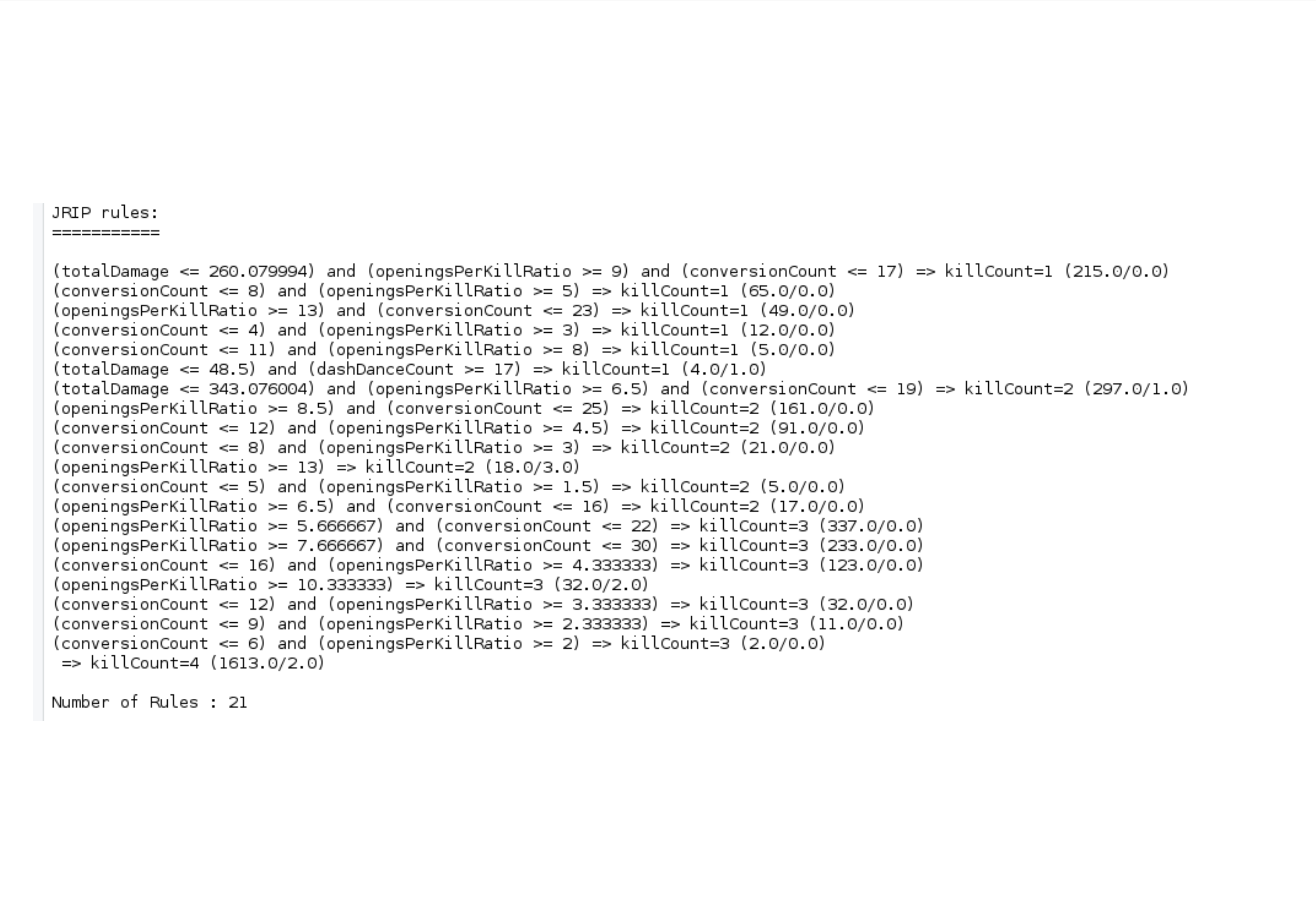
DECISION TREE MODEL

- The decision tree also relied heavily on the totalDamage, openingsPerKillRatio, and conversionCount
- After calculating the performance of nominal values, this model had an accuracy of 90.38%.
- Kappa Statistic: .8638



RULE SET MODEL

- The rules set model ended up creating 21 unique rules for the data set
- After calculating the performance of nominal values, this model had an accuracy of 10.63%.
- Kappa Statistic: -0.175



```
JRIP rules:
===========

(totalDamage <= 260.079994) and (openingsPerKillRatio >= 9) and (conversionCount <= 17) => killCount=1 (215.0/0.0)
(conversionCount <= 8) and (openingsPerKillRatio >= 5) => killCount=1 (65.0/0.0)
(openingsPerKillRatio >= 13) and (conversionCount <= 23) => killCount=1 (49.0/0.0)
(conversionCount <= 4) and (openingsPerKillRatio >= 3) => killCount=1 (12.0/0.0)
(conversionCount <= 11) and (openingsPerKillRatio >= 8) => killCount=1 (5.0/0.0)
(totalDamage <= 48.5) and (dashDanceCount >= 17) => killCount=1 (4.0/1.0)
(totalDamage <= 343.076004) and (openingsPerKillRatio >= 6.5) and (conversionCount <= 19) => killCount=2 (297.0/1.0)
(openingsPerKillRatio >= 8.5) and (conversionCount <= 25) => killCount=2 (161.0/0.0)
(conversionCount <= 12) and (openingsPerKillRatio >= 4.5) => killCount=2 (91.0/0.0)
(conversionCount <= 8) and (openingsPerKillRatio >= 3) => killCount=2 (21.0/0.0)
(openingsPerKillRatio >= 13) => killCount=2 (18.0/3.0)
(conversionCount <= 5) and (openingsPerKillRatio >= 1.5) => killCount=2 (5.0/0.0)
(openingsPerKillRatio >= 6.5) and (conversionCount <= 16) => killCount=2 (17.0/0.0)
(openingsPerKillRatio >= 5.666667) and (conversionCount <= 22) => killCount=3 (337.0/0.0)
(openingsPerKillRatio >= 7.666667) and (conversionCount <= 30) => killCount=3 (233.0/0.0)
(conversionCount <= 16) and (openingsPerKillRatio >= 4.333333) => killCount=3 (123.0/0.0)
(openingsPerKillRatio >= 10.333333) => killCount=3 (32.0/2.0)
(conversionCount <= 12) and (openingsPerKillRatio >= 3.333333) => killCount=3 (32.0/0.0)
(conversionCount <= 9) and (openingsPerKillRatio >= 2.333333) => killCount=3 (11.0/0.0)
(conversionCount <= 6) and (openingsPerKillRatio >= 2) => killCount=3 (2.0/0.0)
 => killCount=4 (1613.0/2.0)

Number of Rules : 21
```

EVALUATION

- Linear Model: 43.1% Accuracy
- OneR Model: 0% Accuracy
- Naïve Bayes Model: 0% Accuracy
- Decision Tree: 90.38% Accuracy
- Rules Model: 10.63% Accuracy

ISSUES WITH THE DATA

The models were not very successful, simply because the data provided is incredibly complicated and hard to truly evaluate properly. When playing a game against another individual, there are so many extraneous features that go into the match. Especially in Melee, with each minute played, there are thousands of decisions that need to be processed, small precise movements that need to be made, and all an extremely small frame of time. The moves a player makes in each circumstance is unique to that player alone, and when so many play styles leads to varying levels of success, it is very difficult to truly understand what allows one player to win and one player to lose. For something like this to be successful, many more features need to be collected, and analyzed on a much smaller scale, specially paying close attention to individual performances rather than generalizing by character.

FUTURE WORK

Expand the number of hooks in the Project Slippi software to collect more in game features

Specialize the features to be how individual characters interact with one another, and how they interact on specific stages

Take into account the different ranges of character skill, continue to see trends by skill tier to find how players can focus and develop

Work on finding out of game features to see how players interact in high pressure situations (etc. record heart rate, interview players on mental game)

CONCLUSION

Issues with complexity of data to get working, proper models

The most successful of the models was the Decision Tree model, with an accuracy of 90.38%

Moving forward, the effort to add to the data framework of Competitive Melee will be to look to character/player specific features, and metal game features

Special thanks to the Project Slippi team, especially Fizzi36 for the creation of the product, as well as answering any questions I had about the .slp files he provided

Shameless Plug: If this seems interesting to you, we operate tournaments every Friday here at Skidmore. Also planning to implement this software before the end of the year!

QUESTIONS/COMMENTS

Thank You!