Poker Hand Classification

Lauren Flemmer
UC Berkeley
lflemmer@berkeley.edu

John H. Lee
UC Berkeley
j-h-lee@berkeley.edu

Andrea Padilla
UC Berkeley
andrea.padilla@berkeley.edu

Abstract

Machine learning is commonly used for classification in cases where there is not an easily observable pattern between classes of the data. For this project we examine the performance of machine learning methods for classifying poker hands. This is a problem that is fairly easy for humans and can be perfectly solved using a conditional algorithm but one that simple machine learning techniques struggle with due to the numerous relations in the data that need to be taken into account. We were able to build fairly accurate models after applying a linear transform to the data but had some problems due to the imbalance in class sizes.

Link to Video Presentation: https://youtu.be/3osjLbTPAiI

Link to Github Repository: https://github.berkeley.edu/lflemmer/poker$_h and_c lassification$

1 Introduction

Poker is an extremely common card game with people spending billions of dollars gambling on it every year. The basic goal of the game is to have the best hand among all players remaining at the end of a round. Although a large number of variants exist, almost all use the same ranking of five card hands. It is fairly easy to determine the hand upon observing the cards as it depends on matching suits, matching rank, and ascending ranks. This problem can be solved fairly easily using a conditional algorithm based on the relation between cards in the hand. However, this is a much more difficult problem for machine learning approaches as the "value" of a suit and rank vary widely depending on the other cards in the hand. The goal of this project is to see if we can adapt machine learning methods to handle these relational problems. A model classifying poker hands could be particularly useful in the field of online poker. For example, such a model could be used in a game or program that allows players to play poker against a bot. Or, additional poker-playing models could be built that, using this poker hand classification model, adapt and improve their poker “strategy” to better understand the strength of their hand.

1.1 Data

Our project uses the Poker Hand data set from the UCI machine learning repository [4] [3]. The data set contains five card hands drawn from a standard deck of 52 cards. Each observation contains the suit and rank for each card in the hand along with the hand. There are four possible values of suit: Heart, Spade, Diamond, and Club, encoded as 1, 2, 3, and 4. There are 13 possible ranks encoded as a numeric value for all. There are 10 possible hands: nothing, one pair, two pair, three of a kind, straight, flush, full house, four of a kind, straight flush, and royal flush, encoded as 0-9.

The data is split into a test and training set with 25,009 and 999,999 observations respectively. There is a large class imbalance within the data as shown in figure 1. This is due to the widely varying likelihoods of each hand. We can see that weaker hands, containing either nothing or one pair, are significantly more common than other hands.

Preprint. Under review.

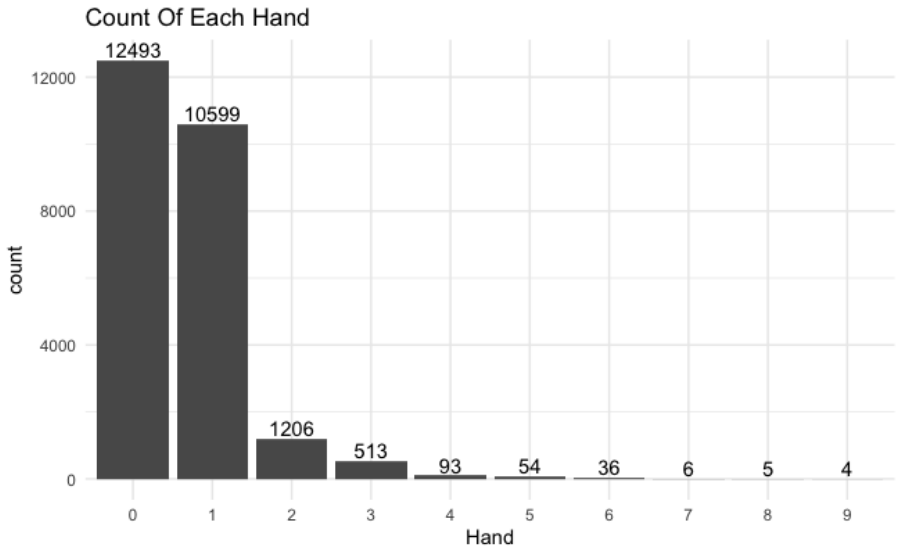

Figure 1: Frequency of each hand in the training set

2 Methodology

2.1 Data Transformation

In later sections we will see how the data performs in its original form. After initial modeling and rereading Cambronero's paper [2] we decided to recreate the linear transformation from the paper. The data is transformed from 11 dimensions to 18 dimensions. Since the original structure of the data encodes each of the five cards' ranks and suits using distinct values, there aren't any features that represent how many of each rank and suit are in the current poker hand. Rather than having the suit and rank of each card represented individually, we can instead take a tally of each rank and suit for every hand. Because the poker hand is so reliant on having cards of the same rank or suit, it is much more meaningful to have that information than the rank and suit of each card. Furthermore, the order in which cards are listed does not matter so losing that information does not hinder the classification.

Original Data
Features 1 through 13: The 13 card ranks, i.e, 1: Ace, 2: Two, 3: Three, ..., 10: Ten, 11: Jack, 12: Queen, 13: King.

Features 14 through 17: The 4 card suits, i.e, 1: Hearts, 2: Spades, 3: Diamonds, 4: Clubs.

Feature 18: The poker hand, i.e, 0: Nothing, 1: One pair, 2: Two pair, 3: Three of a kind, 4: Straight,5: Flush, 6: Full house, 7: Four of a kind, 8: Straight flush, 9: Royal flush.

Data After Transformation
Features 1 through 13: The frequencies of each card rank in the current hand, i.e, 1: # of Aces, 2: # of Twos, 3: # of Threes, ..., 10: # of Tens, 11: # of Jacks, 12: # of Queens, 13: # of Kings.

Features 14 through 17: The frequencies of each card suit in the current hand, i.e, 1: # of Hearts, 2: # of Spades, 3: # of Diamonds, 4: # of Clubs.

Feature 18: The poker hand, i.e, 0: Nothing, 1: One pair, 2: Two pair, 3: Three of a kind, 4: Straight,5: Flush, 6: Full house, 7: Four of a kind, 8: Straight flush, 9: Royal flush.

2.2 Modeling

In order to classify the sets of cards into the 10 poker hands, we implemented various classification methods, with the goal of comparing their classification accuracy on the poker hands. We trained SVM, K-Nearest Neighbors, and XGBoost models, each with different performance and classification

capabilities. Models were all trained on the same training set with 25,009 observations and then tested using a random sample with n = 100,000 taken from the test set. Due to this, a different test set was used for each model. Given that each model was trained on a different test set we can be more sure that model performance is accurately assessed and not overfitting the test set. The individual models are described in more detail below.

2.2.1 SVM

We started with using SVMs as this is a basic algorithm for linearly separable groups of points. Given the complex relationship between the variables we did not expect SVMs to perform well. For the transformed data we also tried SVM with a radial kernel. A radial kernel was chosen as the value of cards does not matter as much as the the relation with other cards and a radial kernel should better capture sets of cards that are closer together.

2.2.2 K-Nearest Neighbors

The second model we trained was K-Nearest Neighbors, with the goal of classifying sets of cards that are close together in the Euclidean space as the same poker hand. Since there are 10 poker hands, we trained a K-NN model with K=10, in an attempt to classify the sets of cards into the 10 hands. This model was trained on the entire training set, then tested on a random sample of the 999,999 test data points, using both the original data set and the transformed data set. Results from this model are discussed in "Results".

2.2.3 Random Forest

Random Forests were used as they often have high predictive power even in cases with nonlinear relationships. Furthermore, random forests are able to capture more complex relationships between variables that other models may not capture. Random forests were trained with $\sqrt{p}$ variables at each split and 500 trees grown.

2.2.4 XGBoost

Because the particular poker hand of a set of cards depends on complex relationships between the cards, it seemed appropriate to model these relationships using a method that learns non-linear decision boundaries. Models like SVM, although powerful, are not as robust as tree-based models, because they utilize linear decision boundaries for classification. We trained an XGBoost classifier with 10 classes, using the relative sizes of the classes as the sample weights of the model. These sample weights ensure that the importance of the classes in the model are based on how frequently they occur in the training data. Since we are performing a multi-class classification, the objective function for training this model was the multi-class classification error rate. Results from our XGBoost model are discussed further in "Results". From the feature importance plot in Figure 4, we can see that the most important feature (from the transformed data) in classifying poker hands is the number of Jacks in the current hand. This is followed closely by the number of 3's, 5's and 6's, respectively. A very interesting observation is that the features encoding the number of cards in a given suit are the least important features in classifying poker hands.

2.2.5 Neural Network

We decided to try using neural networks for reasons similar to those mentioned above. Neural networks are able to recognize patterns that are complex and non-linear. Intuitively, it also makes sense to use a neural network in classification which can easily be done by humans, because the aim of a neural network is to learn like a human brain. We trained the neural net with various types of weights, including proportional as well as inverse. Another option considered was using the square root of the proportion of each class, so that there would be less of an imbalance in weights. Ultimately none of these made a significant difference so weights were omitted and equal weights were used for all observations. Though there are varying opinions on the number of nodes to use, we chose to use some amount between the number of input variables and the number of classes, per the paper we were following [2]. This means for the model using original data we chose 10 nodes (10 input, 10 classes) and for the transformed data we chose 14 nodes (17 input, 10 classes). As discussed in the paper from Cambronero [2], this model is sensitive to feature scaling so the data was scaled to a

range of [0, 1]. Results are shown in the table below on the original and transformed data, both using a single hidden layer. A visual of the final neural net is also available in the appendix as Figure 2. The visualization function comes from Marcus Beck [1] and was created to work with the R package "nnet".

3 Results

3.1 Results on Original Data

Accuracy of Models on Original Data

Model	Nothing	One pair	Two pair	Three of a kind	Straight	Flush	Full house	Four of a kind	Straight flush	Royal flush	**Overall**
SVM	0.7674	0.3880	0.0000	0.0000	0.3077	0.0000	0.0000	0.0000	0.0000	0.0000	0.5506
K-NN	0.5939	0.5726	0.5068	0.5037	0.5008	0.5000	0.5000	0.5000	0.5000	0.5000	0.5688
RF	0.8238	0.4976	0.0004	0.0014	0.0026	0.0053	0.0000	0.0000	0.0000	0.0000	0.6242
XGB	0.7923	0.5136	0.3164	0.2318	0.1827	0.2192	0.0174	0.0000	0.0000	1.0000	0.6347
NN	0.8469	0.2857	0.0017	0.0009	0.0000	0.2475	0.0000	0.0000	0.0000	0.0000	0.5463

3.2 Results on Transformed Data

Accuracy of Models on Transformed Data

Model	Nothing	One pair	Two pair	Three of a kind	Straight	Flush	Full house	Four of a kind	Straight flush	Royal flush	**Overall**
SVM	1.0000	0.0079	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	1.0000	0.5064
KSVM	1.0000	1.0000	1.0000	1.0000	0.2770	0.8030	0.0000	0.0000	0.0000	0.0000	0.9951
K-NN	0.7792	0.7227	0.5043	0.5007	0.5013	0.5466	0.5000	0.5000	0.5000	0.5000	0.7416
RF	1.0000	1.0000	1.0000	1.0000	0.7930	0.3495	0.0629	0.0435	0.0000	0.0000	0.9964
XGB	0.9983	0.9999	1.0000	0.9990	0.9902	0.9043	0.2602	0.3600	0.0000	1.0000	0.9976
NN	0.9722	0.6689	0.1046	0.0018	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.7748

4 Discussion

Based on the results we can see that the transformation led to significant increases in performance for all models other than SVM with a linear kernel. We should be careful about the high overall accuracy of the models as this is due to the large class imbalance in the data. At best, models are highly accurate at classifying hands up to a Flush and then performance drops off. On the transformed data, we found that our SVM with a Radial Kernel and XGBoost models performed the best in classifying the poker hands, with over 99% accuracy. However, the Radial Kernel SVM model had 0% accuracy in classifying the last four poker hands. Our XGBoost model, while still not great at classifying those less frequent poker hands, had higher classification accuracy for those last four hands, and a more balanced classification accuracy among all poker hands than our other models.

In almost all cases, model performance decreased as the value of the hand increased likely due to the imbalance in class size. It should be noted that the estimates of model performance for the hands Four of a kind, Straight Flush, and Royal Flush are unlikely to be accurate due to the small number of observations in the test sample.

Figures 4 and 5 in the appendix show the feature importance for XGBoost and the Random Forest trained on the transformed data. Both models find the values on the cards to be more important than the suit of the cards in the hand.

5 Conclusions

Although many of our models have high accuracy, they are weakened by the class imbalance and classification error within these smaller classes. Further work could be done to address this class imbalance, such as oversampling minority classes or undersampling majority classes, in order to help the models better learn associations between cards in those less frequent poker hands. Additionally, performing hyperparameter tuning with our tree-based models and Neural Network would likely improve the overall performance of these models, and possibly improve our low classification accuracy on less frequent poker hands.

6 Appendix

Below are figures generated after training some of our models on the transformed data.

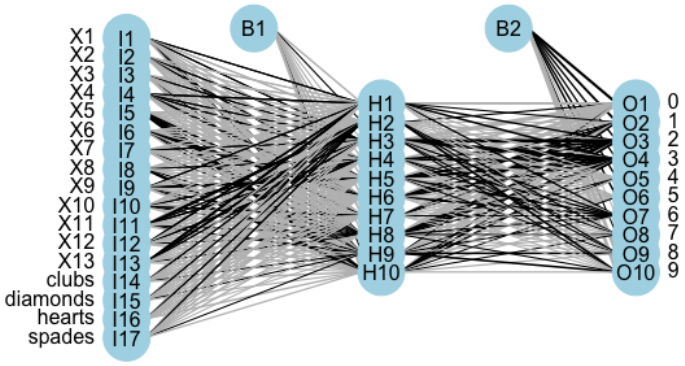

Figure 2: Neural net on transformed data

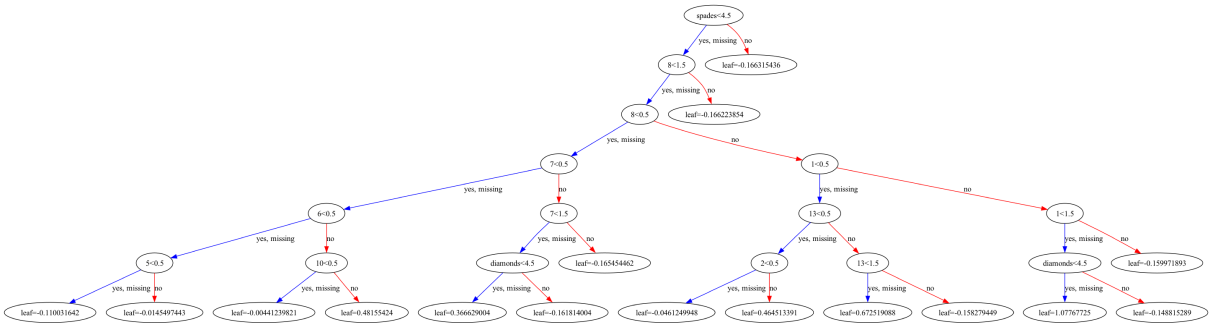

Figure 3: XGBoost model trained on transformed data

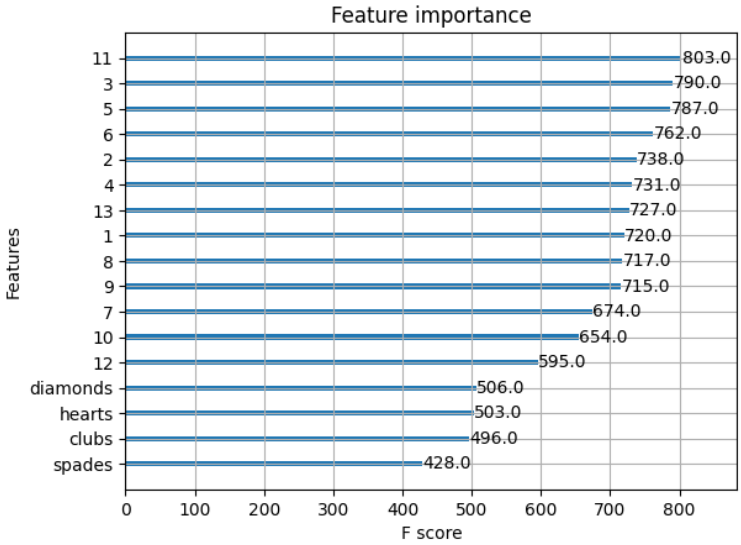

Figure 4: Feature importance of XGBoost on transformed data

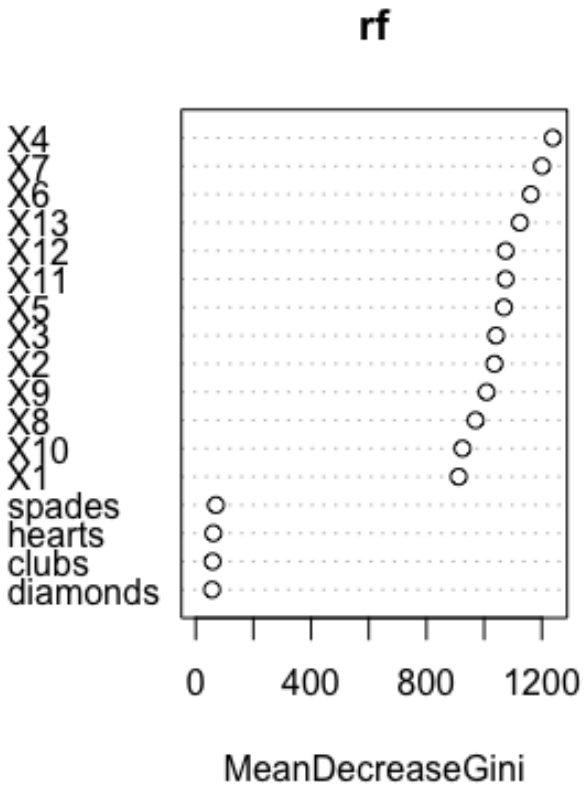

Figure 5: Random Forest Feature Importance on Transformed Data

References

[1] Marcus Beck. *Visualizing Neural Networks in R – update*. June 2014. URL: https://beckmw.wordpress.com/2013/11/14/visualizing-neural-networks-in-r-update/.

[2] Walinton Cambronero. *Poker Hand Dataset: A Machine Learning Analysis and a Practical Linear Transformation.* URL: https://walintonc.github.io/papers/ml_pokerhand.pdf.

[3] Robert. Cattral and Franz Oppacher. *UCI Machine Learning Repository*. 2019. URL: http://archive.ics.uci.edu/ml.

[4] Dheeru Dua and Casey Graff. *Poker Hand Data Set*. 2017. URL: https://archive.ics.uci.edu/ml/datasets/Poker+Hand.