**Poker Rule Induction**

**Team Members:** Tae Hong, David Tang, Rafael Mendoza, Andrew Gonzalez

Inspired by the [Poker Rule Induction](https://www.kaggle.com/c/poker-rule-induction/data) problem from Kaggle.

The objective of the Poker Rule Induction challenge is to create a machine-learned model that can detect poker hands.

The data, as provided by Kaggle, consists of five cards from a deck of playing cards excluding the jokers. Most of the work was done in Python3 using Jupyter Notebook, with the help of libraries such as Pandas, Numpy, and Scikit-Learn. Microsoft Excel was also used in handling the CSV file holding the data.

Each team member was responsible for writing the report and creating the presentation.

* **Tae Hong:** Focus on getting the best results possible with Random Forest.
* **David Tang:** Run KNN and Logistic Regression models for the problem.
* **Rafael Mendoza:** Create new data entries to supplement what is provided by Kaggle.
* **Andrew Gonzalez:** Add additional features to data set.

**Analyzing the data**

In the data set, each row represents a hand composed of five cards with a given suit and rank, along with its classifier. There are five features for the suit and rank of the card. The last column in each row is the classifier that dictates which poker hand the five cards form. S# is the feature that represents the suit of the card with values (1-4), which represents: Hearts(1), Spades(2), Diamonds(3), and Clubs(4). C# is the feature that represents the rank of the card with values (1-13). Values 1-13 represent: Ace, 2, 3, …, 10, Jack, Queen, and King. Our classifier is the column labeled ‘hand’, which holds values (0-9) that represents: Nothing(0), One Pair(1), Two Pairs(2), Three of a kind(3), Straight(4), Flush(5), Full House(6), Four of a Kind(7), Straight Flush(8), and Royal Flush(9). This is what each row in the data consists of. We sourced our data from the Poker Rule Induction challenge on Kaggle.

To provide greater accuracy, additional features had to be implemented. Since all poker hands consist of some combination of pairs, triples, four-of-a-kinds, straights, and flushes, these were the first features added. Then to further improve the sensitivity of some of the higher ranking hands, we created features specific to them: top end straights (a straight consisting of 10 through Ace) and straight flushes (a combination of straights and flushes). These features were added by running the original data through algorithms that detected these particular traits.

Another aspect of data that we took advantage of to further improve our model was adding data entries for higher ranking hands. Since the rules of poker were set in stone, it was only a matter of making sure the correct values were put in the S# and C# columns and making sure that none of the data was repeated. We added entries for hands ranging from straights to royal flushes, and this proved especially critical in detecting four-of-a-kinds, straight flushes, and royal flushes.

**Our process**

The models used were KNN, logistic regression, and random forest. The process of cross validation repeatedly splits the data K times, so that all data samples will be used (K-1) times in the training set and one time in the testing set. To evaluate each model fairly, cross validation was used with cv = 10 and scoring = accuracy to ensure fair splitting of the data. KNN performed best when the value of k was around 30.

The group settled on random forest because it was the best model. Logistic regression was not the best model because it is commonly used to classify data with binary labels, but the data had many labels for poker hands. Random forest is a very accurate classification algorithm that can be used to improve the model’s accuracy. The n-estimators value used before features were added was 10. Then it was increased to 50 when used for cross validation with additional features in the dataset. The highest ranking hands did not have much data samples, so additional data was created to test the sensitivity of them.

**Our results**

**Initial Results**

Our initial results were as follows:

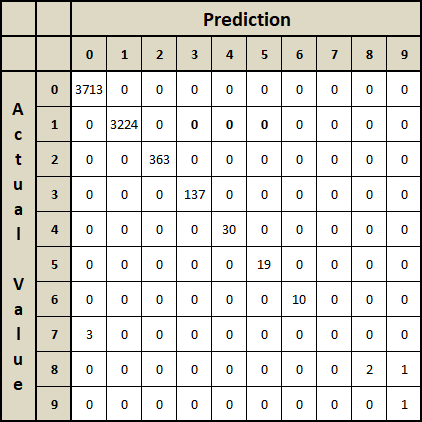
|  |  |  |
| --- | --- | --- |
| **Accuracy with default data set and no additional features** | | |
| **KNN** | **Logistic Regression** | **Random Forest** |
| 58.15% | 49.96% | 56.49% |

These results were not particularly exciting nor surprising since we expected low accuracy scores with features that only listed the rank and suit of the five cards in a player’s hand. The first step we took to improve accuracy was to add features unique to the entire hand that have meaning in poker. Initially, only pair, triple, four of a kind, and flushes were added. These inclusions were enough to boost the accuracy of our Random Forest model to over 99%, which we decided as our model of focus.

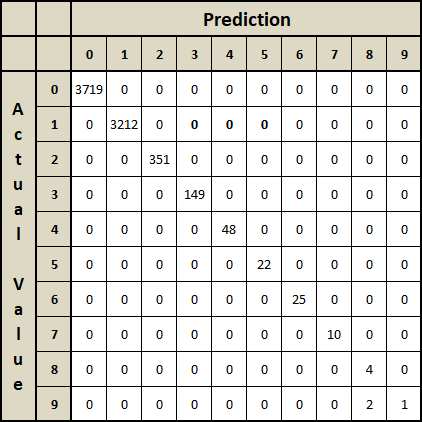
**Improved results**

However, there was still a critical issue of sensitivity. Upon looking at the confusion matrix of our random forest model, the sensitivity for straights, full houses, four-of-a-kinds, straight flushes, and royal flushes were below 70%. To help combat this issue, the value of n-estimators was increased to 50, and the features for straight, top straight (a straight that runs from 10 to Ace), and straight flush were added. After this implementation, the accuracy and confusion matrix were as follows (next page):

|  |  |  |
| --- | --- | --- |
| **Accuracy with additional features** | | |
| **KNN** | **Logistic Regression** | **Random Forest** |
| 69.64% | 97.50% | 99.94% |



In the confusion matrix, the values 0-9 at the top and far left represented each poker hand from ‘‘nothing’’ (0) all the way to “Royal Flush” (9). In each row, the values where the prediction value was equal to the actual value (i.e. (0, 0), (1, 1), etc.) were the true positives for that hand, with the other squares being false positives. Although we hit very high levels of accuracy, we were still uncertain how sensitive the highest ranking few hands were due to there being too few samples. Fortunately, poker hands were very easy to create new data for. We added additional samples for all hands equal to or greater than a straight to get the following confusion matrix from our random forest (next page):



The above confusion matrix revealed there might still be a problem with our methodology, so we tried trimming down the number of features to only focus on card sets that mattered (pairs, flushes, etc.) and removing all individual card values. The results we got were as follows, with only a single false positive for royal flushes in our random forest model:

|  |  |  |
| --- | --- | --- |
| **Accuracy with card suits and faces removed using the original data set** | | |
| **KNN** | **Logistic Regression** | **Random Forest** |
| 99.93% | 95.14% | 99.98% |

**Perfect Accuracy**

When the finalized random forest model of features that excluded all individual card values was ran with our customized data set, we reached a perfect accuracy of 100%.