Classification of Poker Hands using Decision Tree, Random Forest, K Nearest Neighbors, and Support Vector Machine

George Zhou

[1] University of Washington Bothell, USA
[2] CSS 490 Machine Learning

Abstract. Ever since the introduction of video poker games, computers are explicitly programmed to recognize poker hands using Genetic Algorithms. Due to the simple nature of poker games, it's trivial for humans to validate poker hand rules. This paper discusses the possibility of recognizing poker hands using machine learning algorithms. The dataset includes a list of poker hands. Each data instance is a sample poker hand consisting of five poker cards from a uniform deck of 52 (Excluding 2 jokers). Each card is represented using two attributes, suit and rank, for a total of 10 predictive attributes. At the end of each instance, there is one Class label that represents the poker hand scores. There are 10 Class labels, ranging from *Nothing in hand* to *Royal flush*. The goal of my research is to predict class labels for each data instance using classification methods, including Decision Tree, Random Forest, K Nearest Neighbors, and Support Vector Machine. In addition, K Fold Validation is implemented in order to improve performance.

Keywords: Machine Learning, Classification, Decision Tree, Random Forest, K Nearest Neighbor, Support Vector Machine, K Fold Validation, Poker Hand.

1 Introduction

Texas Hold'em and many other online video poker games are becoming increasingly popular nowadays. Nevertheless, computers are still trained to recognize poker hands using Genetic Algorithms. Because of the way that the problem is represented it is difficult to discover rules that can correctly classify poker hands, however, the simple nature of the game makes it trivial for the human analyst to validate potential rules objectively.[1]

At every Texas Hold'em games, two hole cards are dealt face down to each player, and then five community cards are dealt face up in three stages. Each poker player seeks the best five card poker hand from any combination of the seven cards of the five community cards and their two hole cards. [1] This project implements several machine learning algorithms and validation technique to discover the most accurate, fine-tuned, optimized model, to recognize poker hands with five cards.

The poker hand dataset is contributed by Robert Cattral and Franz Oppacher at Carleton University, Canada. [1] More dataset information will be discussed in 1.1.

Robert and other researchers at Carleton University have used genetic algorithms (GA) on the poker hand dataset with slightly different ranking rules, which is considered more difficult. The GA has achieved an average accuracy of 57.6% overall test cases. [1]

1.1 Dataset overview

The dataset contains both training set and testing set. Training set has 25,010 instances; testing set has 1,000,000 instances. Each data instance is a sample poker hand consisting of five poker cards from a uniform deck of 52 (Excluding 2 jokers). Each card is represented using two attributes, suit ,and rank, for a total of 10 predictive attributes. There is one class label that describes the poker hand scores. The position of cards is critical to the uniqueness of poker hands, which is the reason why there are 480 possible Royal Flush hands as compared to 4 (One for each suit explained in more detail below).

Attribute Information

1. Suit of card: Ordinal (1 - 4) representing Hearts, Spades, Diamonds, Clubs.
2. Rank of card: Numerical (1 - 13) representing Ace, 2, 3, …, Jack, Queen, King.
3. Class labels (Poker Hand Scores): Ordinal (0 - 9) explained more in **Table 1**.

A quick example of individual data instance should look like this:

$$1, 10, 1, 11, 1, 13, 1, 12, 1, 1, 9 \quad (1)$$

The sample reads Ten of Hearts, Jack of Hearts, King of Hearts, Queen of Hearts, Ace of Hearts, Royal Flush. Since there are five cards in each poker hand, five pairs of suit and rank for each data instances.

Table 2. Poker Hand Scores

Score	Name	Description	Example [1]
9	Royal Flush	Ace to 10 Flush	AH KH QH JH 10H
8	Straight Flush	Five sequential cards (same suit)	4C 5C 6C 7C 8C
7	Four of A Kind	Four cards of the same rank	2H 2D 2S 2C 8S
6	Full House	Three of a kind plus one pair	3D 3C 7H 7S 7D
5	Flush	Five cards of the same suit	2C 3C 6C 9C AC
4	Straight	Five sequential cards	3C 4C 5D 6H 7D
3	Three of A Kind	Three equal cards (same rank)	5H 5S 5D 3C 7H
2	Two Pairs	Two pairs of equal cards	4H 4S 9D 9S 7C
1	One Pair	Two equal cards	5H 5C 7D JS 2D

0	Nothing in Hand	No useful cards in group	2S 3D 6H 9C 10H

[1] The suit and rank of cards are represented using the first letter of the names for efficiency reasons. A for Ace, K for King, Q for Queen, J for Jack, H for Hearts, S for Spades, D for Diamonds, C for Clubs. In the actual dataset, however, they represented by numbers as mentioned earlier.

1.2 Probability & Statistics

Entire Domain

From a standard deck of 52 (Excluding two Jokers), there are a total of 311,875,200 combinations for a poker hand of 5. However, the total number of unique hands are only 2,498,960, because the positions of cards might change. The probability decreases when going up the rank, thus resulting in a bigger payout in play.

Table 2. Poker Hand Probability

Name	No. of Hands	No. of Combinations	Probability
Royal Flush	4	480	0.00000154
Straight Flush	36	4320	0.00001385
Four of A Kind	624	74880	0.0002401
Full House	3744	449280	0.00144058
Flush	5108	612960	0.0019654
Straight	10200	1224000	0.00392464
Three of A Kind	54912	6589440	0.02112845
Two Pairs	123552	14826240	0.04753902
One Pair	1098240	131788800	0.42256903
Nothing in Hand	1302540	156304800	0.50117739
Total	2598960	311875200	1.0

Training Set

In this training set, Straight Flush and Royal Flush are over-sampled. The chance of getting a Straight Flush in hand is 14.43 times higher than reality; the chance of getting a Royal flush is 129.32 times higher than the entire domain.

Table 2. Training Set Probability

Name	No. of instances	Probability
Royal Flush	5	0.0001999
Straight Flush	5	0.0001999
Four of A Kind	6	0.0002399
Full House	36	0.0014394
Flush	54	0.0021591
Straight	93	0.0037185
Three of A Kind	513	0.0205118

Two Pairs	1206	0.0482207
One Pair	10599	0.4237905
Nothing in Hand	12493	0.4995202
Total	25010	1.0

Testing Set

The testing set has a total number of 1,000,000 instances from the entire domain of 311,875,200 Poker hands. Again, taking Royal Flush as an example, with a total of 3 instances out of 1,000,000 instances, the chance of getting Royal Flush in the testing set is still much higher than the entire domain.

Table 3. Testing Set Probability

Name	No. of instances	Probability
Royal Flush	3	0.0000030
Straight Flush	12	0.0000120
Four of A Kind	230	0.0002300
Full House	1424	0.0014240
Flush	1996	0.0019960
Straight	3885	0.0038850
Three of A Kind	21121	0.0211210
Two Pairs	47622	0.0476220
One Pair	422498	0.4224980
Nothing in Hand	501209	0.5012090
Total	1000000	1.0

2 Methods

This section covers the genetic algorithms used by previous researchers as well as the machine learning used in this project

2.1 Genetic Algorithms

A quick example of the genetic algorithm used by Robert is displayed below. Followed by a refined version of the algorithm.

Brute Force

Here's an example rule below classify three subsets of the poker hands as a Three of A Kind, Score no. 3. These three rules are not confident and accurate throughout all combinations, because they do not consider the condition of the possibility of a Full House, Rank no. 6.

If (R2 == R5) && (R5 == R1) && (R5 != R3) && (R4 != R5) (2)

If (R1 == R2) && (R2 == R3) && (R4 != R1) && (R5 != R1) (3)

If (R2 == R3) && (R3 == R4) && (R5 != R3) && (R1 != R3) (4)

These three rules indicate that they are sensitive to the order/position of the cards, which means that a huge amount of work is required to completely classify even one single score of the poker hands.

Generalized rule

A more generalized example of classifying Three of A Kind is shown below:

If (NumberEqualRank == 3) {return SCORE == 3} (5)

The equation reads: "In a set of poker hand, if number of cards with equal ranks equals to 3, then we have a Three of A Kind"

This generalized rule is not position sensitive, because it's calculating the total number of cards with same rank. However, the rule does not consider the possibility of getting a Full House in hand.

2.2 Decision Tree

Decision Trees are a non-parametric supervised learning method used for classification and regression. The goal is to create a model that predicts the value of a target variable by learning simple decision rules inferred from the data features. [3]

Entropy and Information Gain are calculated for each attribute at each tree level. Decision Tree chooses the attribute with the highest information gain to split the tree at the current level.

2.3 Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size, but the samples are drawn with replacement if bootstrap=True (default). [4]

The steps of the random forest algorithm are stated as follows:

1. Draw n-tree samples from the original data set.
2. For each data instance, grow a classification tree, with the following modification: at each node, rather than choosing the best split among all predictors, randomly sample of the predictors and choose the best split from among those variables.
3. Predict new data by aggregating the predictions of the n-tree trees (i.e., majority votes for classification, the average for regression). [4]

2.4 K Nearest Neighbors

The principle behind nearest neighbor methods is to find a predefined number of training samples closest in distance to the new point and predict the label from these. The number of samples can be a user-defined constant (k-nearest neighbor learning) or vary based on the local density of points (radius-based neighbor learning). [5]

The steps of K Nearest Neighbors algorithm are stated as follows:

1. Determine parameter K = number of nearest neighbors
2. Calculate the distance between the query-instance and all the training samples
3. Sort the distance and determine nearest neighbors based on the Kth minimum distance
4. Gather the category of the nearest neighbors
5. Use the simple majority of the category of nearest neighbors as the prediction value of the query instance [5]

2.5 Support Vector Machine

Given a set of training examples, each marked as belonging to one or the other of two categories, an SVM training algorithm builds a model that assigns new examples to one category or the other, making it a non-probabilistic binary linear classifier. Although methods such as Platt scaling exist to use SVM in a probabilistic classification setting. [6]

2.6 Support Vector Machine (Linear Kernel)

LinearSVC is another implementation of Support Vector Classification for the case of a linear kernel. Note that LinearSVC does not accept keyword kernel, as this is assumed to be linear. [6]

2.7 K Fold Validation

Learning the parameters of a prediction function and testing it on the same data is a methodological mistake: a model that would just repeat the labels of the samples that it has just seen would have a perfect score but would fail to predict anything useful on yet-unseen data. This situation is called overfitting. To avoid it, it is common practice when performing a (supervised) machine learning experiment to hold out part of the available data as a test set X_test, y_test. Note that the word “experiment” is not intended to denote academic use only, because even in commercial settings machine learning usually starts out experimentally.

This project implements 3-Fold validation on Decision Tree and Random Forest algorithms. As stated in Table 3, the minimum number of instances among all poker

ranks is 3 (Royal Flush) in the testing set. Thus, 3-Fold validation is the maximum of K Fold Validation on this particular data set.

3 Results

3.1 Decision Tree

As we can see in Fig 1, Decision Tree is making predictions for poker hands with all ranks. Decision Tree is not able to correctly predict any poker hands who has a rank score higher or equal to 7. Moreover, the algorithm is better at predicting poker hands with lower scores and high possibilities. In other words, Decision Tree is more sensitive to data instances of high occurrences.

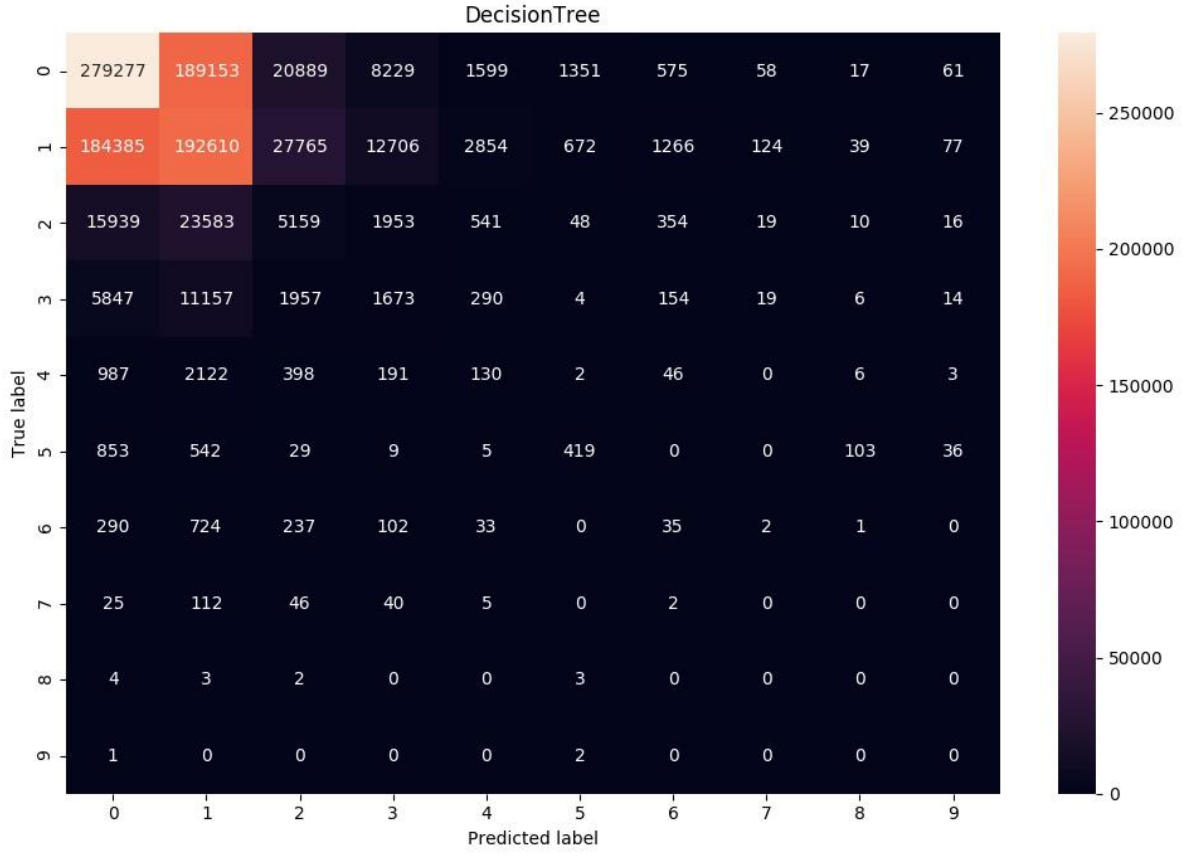

Fig 1. Decision Tree Confusion Matrix

3.2 Decision Tree (Three-Fold Validation)

After three-fold validation, the accuracy of Decision Tree increased 11.88%. Decision Tree is making predictions for poker hands with rank score 0 to 7. Nevertheless, Decision Tree is still unable to correctly predict any poker hands who has a rank score higher or equal to 8. Overall, the model sees improvements in performance as well as data sensitivity.

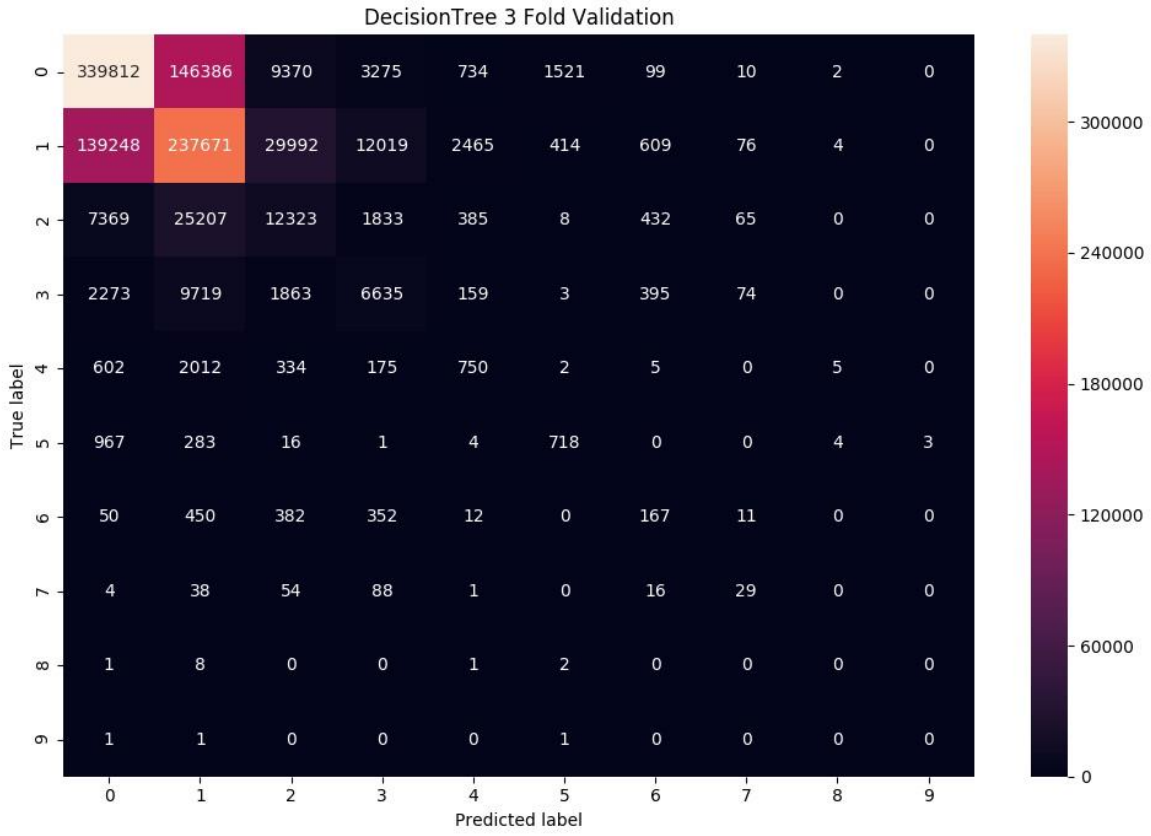

Fig 2. Decision Tree (Three-Fold Validation) Confusion Matrix

3.3 Random Forest

Random Forest Algorithm is only making predictions for poker hands with rank score 0 to 3. Random Forest is unable to recognize poker hand rules for the hands with rank score higher or equal to 4. Comparing to Decision Tree, Random Forest is more sensitive to Poker Hand Rank Score 0—Nothing in Hand, while suffering performance issue for all other poker hand ranks.

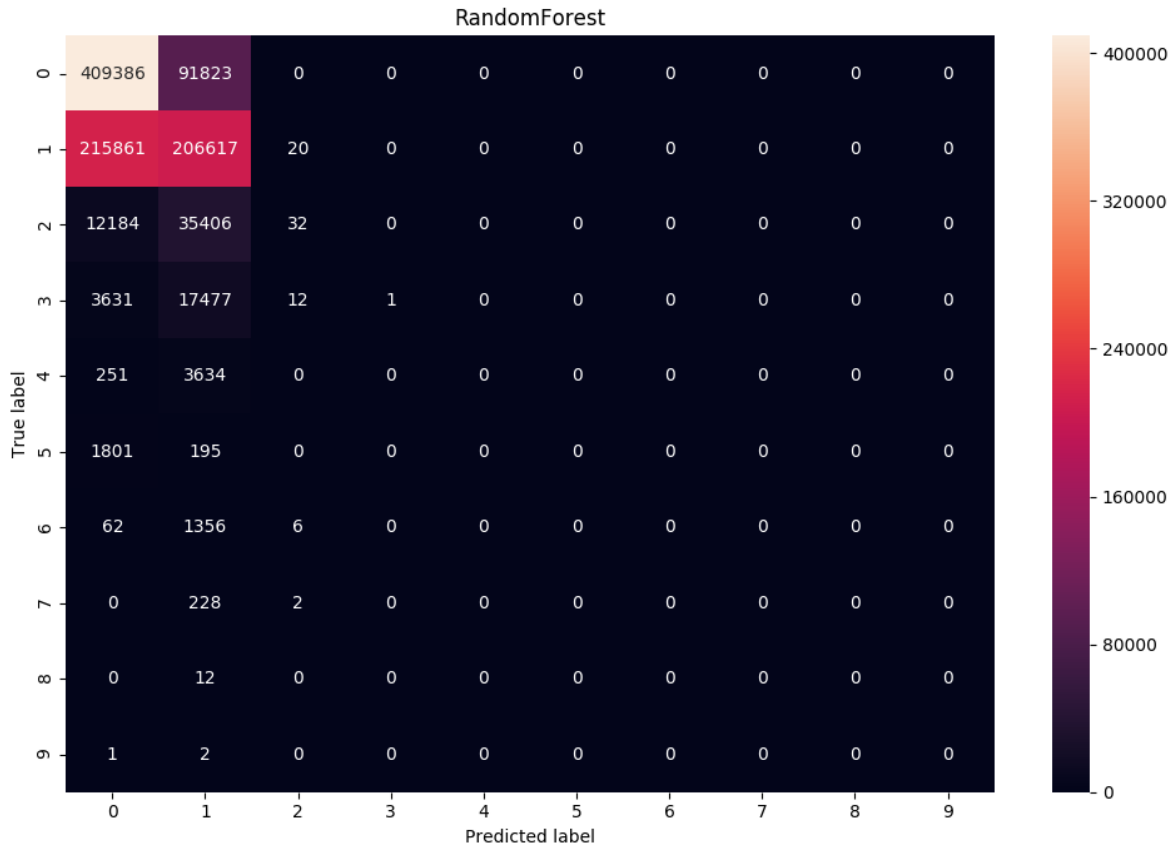

Fig 3. Random Forest Confusion Matrix

3.4 Random Forest (Three Fold Validation)

After three-fold validation, the accuracy of Random Forest increased 11.76%. Random Forest is making predictions for poker hands with rank score 0 to 3 and 5. The model is still unable to learn the pattern for poker hands with higher rank scores.

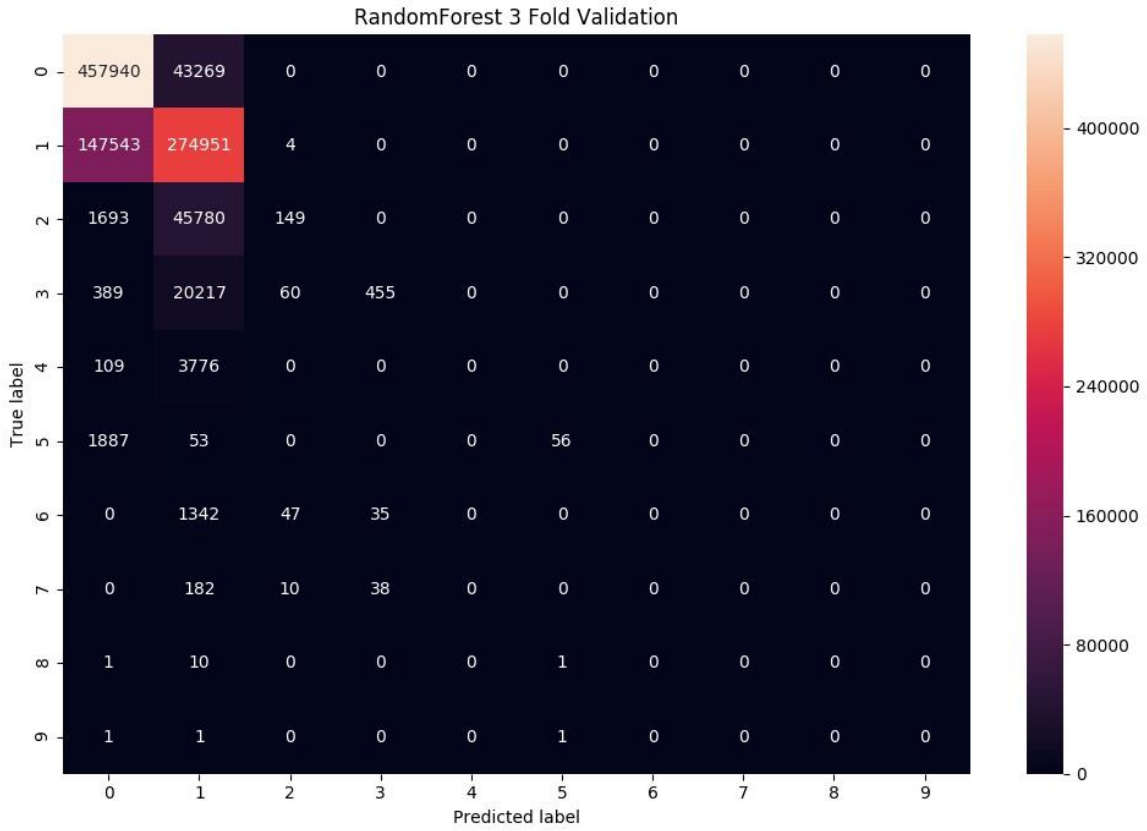

Fig 4. Random Forest (Three Fold Validation) Confusion Matrix

3.5 K Nearest Neighbors

In this project, the number of nearest neighbors is set to 3, which is considered more precise. (Potential over-fitting of the dataset) K Nearest Neighbors is making prediction for all poker hand ranks. However, K Nearest Neighbors could only correctly predict the poker hands with rank score from 0 to 6.

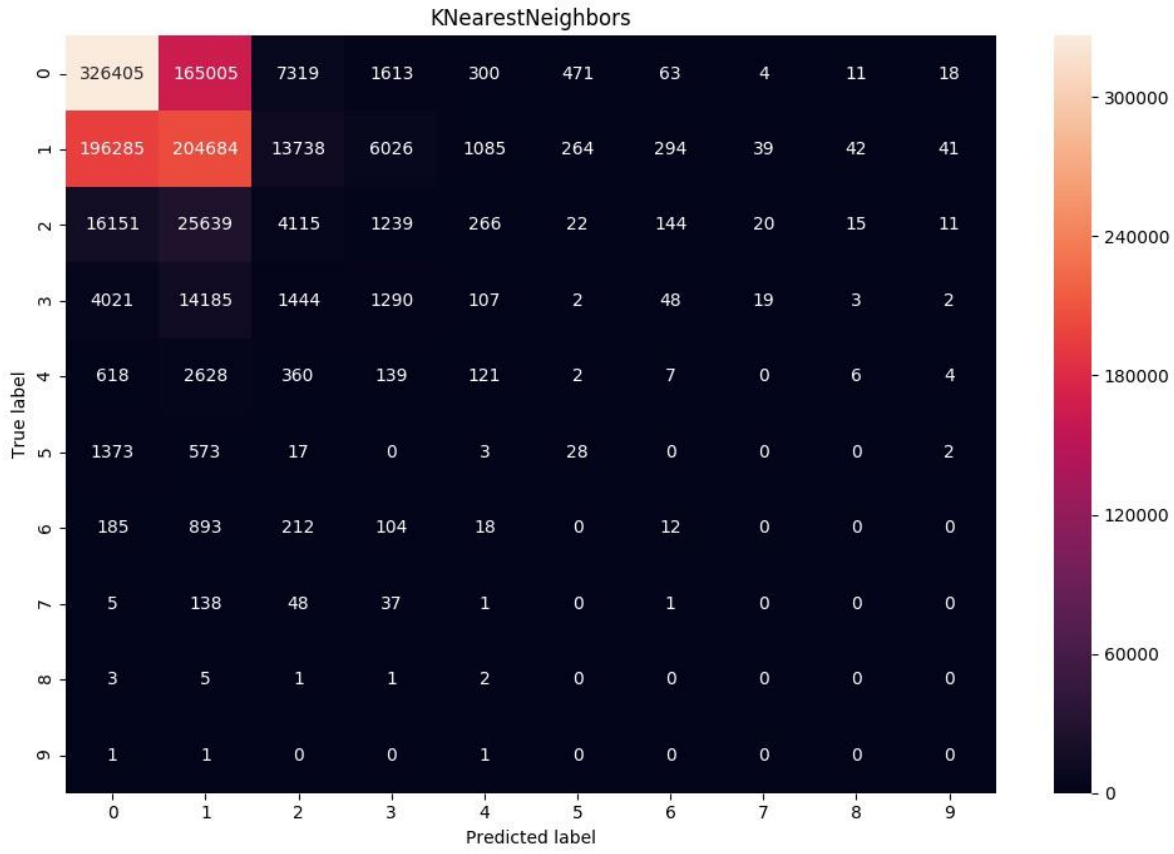

Fig 5. K Nearest Neighbors Confusion Matrix

3.6 Support Vector Machine

Same as Random Forest, Support Vector Machine is only making predictions for poker hands with rank from 0 to 3. The model is unable to predict poker hands with higher rank scores.

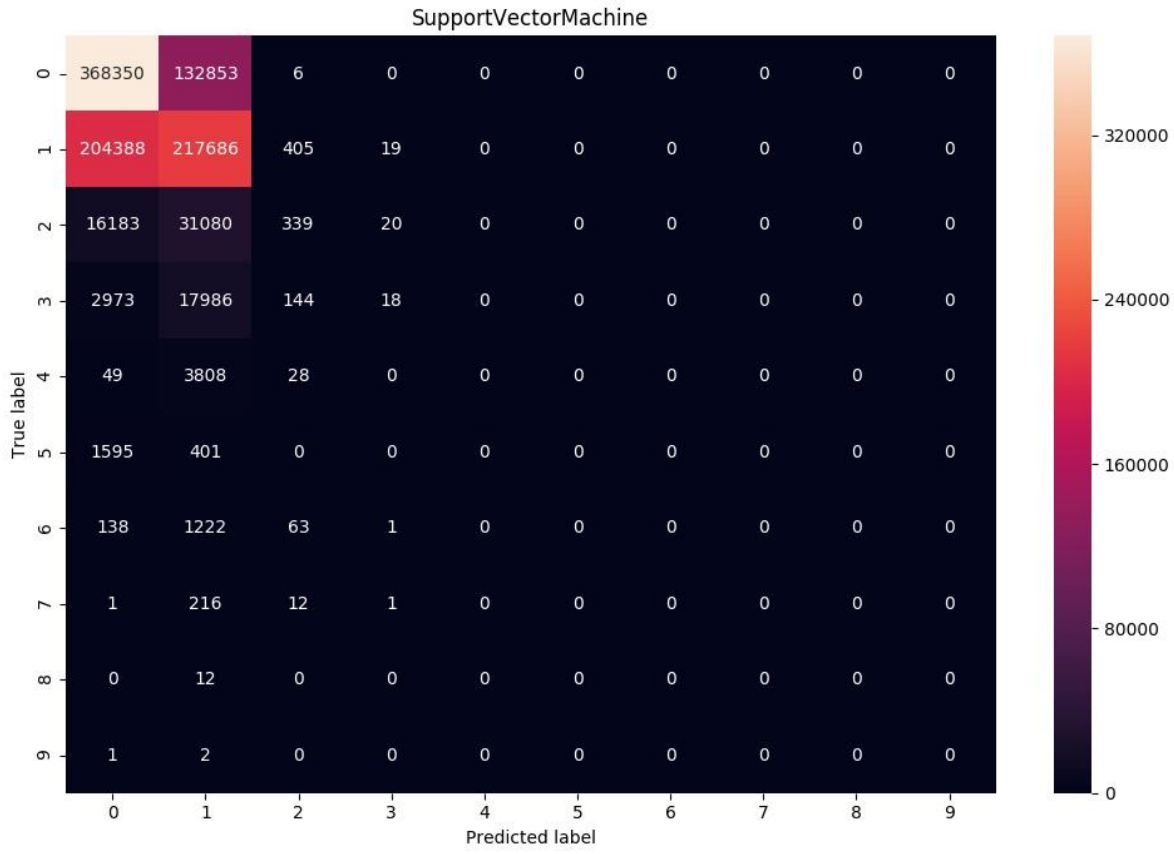

Fig 6. Support Vector Machine Confusion Matrix

3.7 Support Vector Machine (Linear Kernel)

Linear Support Vector Machine is making predictions of poker rank 0 and 1. It is not making any predictions other than that. Despite the fact that linear support vector machine is more sensitive to Nothing in Hand and One Pair, support vector machine in 3.6 has better average accuracy.

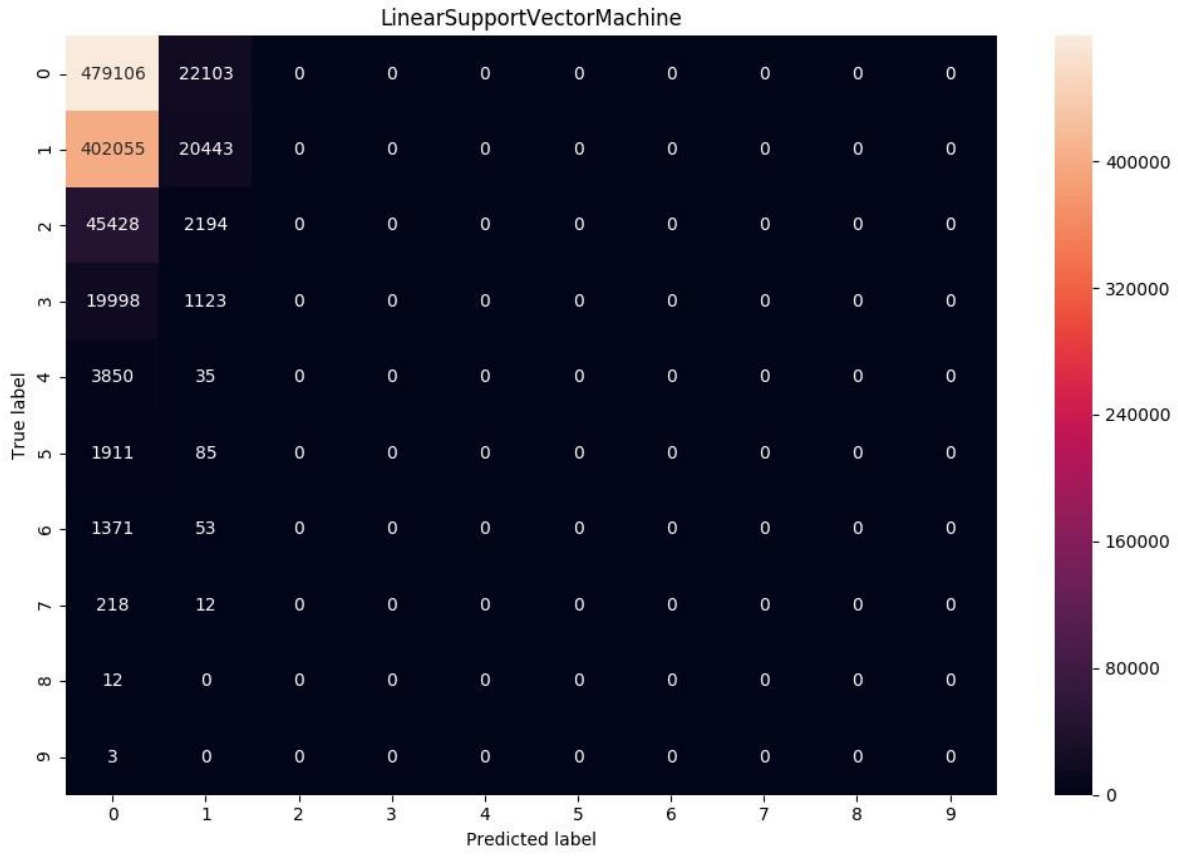

Fig 7. Support Vector Machine (Linear Kernel) Confusion Matrix

3.8 Model Performance Comparison

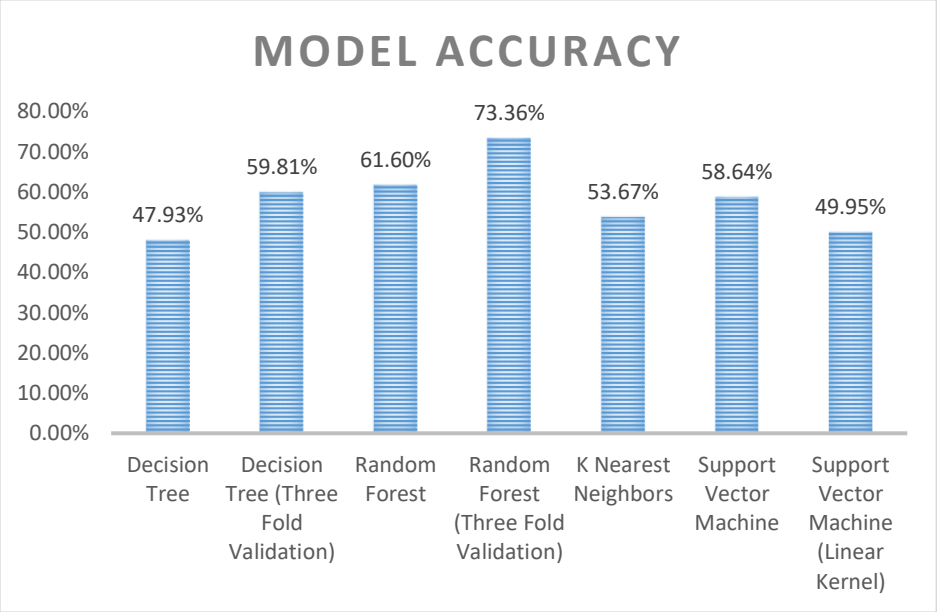

Fig 8. Model Performance Chart

In the term of Accuracy, the model is listed as follow, from best to worst, Random Forest (3-fold validation), Random Forest, Decision Tree (3-fold Validation), Support Vector Machine, K Nearest Neighbors, Support Vector Machine (Linear Kernel), Decision Tree. As we can see in Fig 8, Random Forest (three-fold validation) has the highest accuracy—73.36%. After three-fold validation, Decision Tree and Random Forest are able to boost their accuracy by approximately 11%. Support Vector Machine lost 8% accuracy after switching to linear kernel.

Random Forest has better accuracy than Decision Tree in general. However, Decision Tree is able to recognize more poker hands ranks.

Similarly, K Nearest Neighbors has worse accuracy than Support Vector Machine, while predicting more poker hands in the terms of number of rank categories.

4 Conclusion

Due to the nature of the dataset, the outcome values are highly dependent on each individual attributes of the data set. Thus, it's not feasible to reduce the dimensionality of the dataset while maintaining the overall integrity at the same time. Algorithms such as Principal Component Analysis, etc. are not a good match for the poker hand dataset.

On the other hand, Genetic Algorithms and dictionary models require a huge amount of work to explicitly specify genetic rules for every condition possible.

Classification algorithms like tree-structured models (Random Forest and Decision Tree) offer alternatives with decent accuracy rate, which is more accurate than the genetic model implemented by Robert. [1] Robert and other researchers at Carleton University have used genetic algorithms (GA) on the poker hand dataset with slightly different ranking rules, which is considered more difficult. The GA has achieved an average accuracy of 57.6% overall test cases.

Overall, machine learning models implemented in this project managed to either tie or out-perform Genetic Algorithms. Future work is required to further tune the classification models, which should focus on improving the sensitivity to poker hands with lower occurrences.

References

1. R. Cattral, F. Oppacher, D. Deugo. Evolutionary Data Mining with Automatic Rule Generalization. Recent Advances in Computers, Computing and Communications, pp.296-300, WSEAS Press, 2002.
2. Lemaître, G. Nogueira, F. Aridas, Ch.K. (2017) Imbalanced-learn: A Python Toolbox to Tackle the Curse of Imbalanced Datasets in Machine Learning, Journal of Machine Learning Research, vol. 18, no. 17, 2017, pp. 1-5.
3. Rokach, Lior; Maimon, O. (2008). Data mining with decision trees: theory and applications. World Scientific Pub Co Inc. ISBN 978-9812771711.
4. A Liaw, M Wiener (2002). Classification and regression by randomForest. Vol. 2/3, December 2002.
5. Piotr Indyk, Rajeev Motwani (1998). Approximate nearest neighbors: towards removing the curse of dimensionality. STOC '98 Proceedings of the thirtieth annual ACM symposium on Theory of computing, Pages 604-613.
6. Terrence S. (2000) Support vector machine classification and validation of cancer tissue samples using microarray expression data. Bioinformatics, Volume 16, Issue 10, 1 October 2000, Pages 906–914.