Poker Hand

EECSE6690.2019Fall.StatisticalLearning.Report
Prutha Parmar prp2126, Kavita Anant ka2744, Jeswanth Yadagani jy3012
Columbia University

Abstract

The objective of our project is to approach the area of automatic rule induction and data mining systems. Imagine coming up with rules for a game you have never played on the basis of history of thousands of games. It is quite a challenging problem. Our project intends to acknowledge this problem by reproducing the algorithms from the paper "Evolutionary Data Mining With Automatic Rule Generalization"[1]*. For this project we are using the "Poker Hand Data Set provided by UCI Machine Learning Repository"*[2]*. One of the major technical challenge was to design an algorithm that invalidates the importance of the position of each card in a poker record i.e. train the model to consider multiple permutations for each poker hand. Furthermore, we trained various models and got a test accuracy of 95.57% after feature engineering.*

Keywords- RAGA, Feature Engineering, Class, Suite, Rank

1. Introduction

Imagine you are given a set of observations/outcomes and based on that you are supposed to predict the rules. Rule Induction is an area of machine learning which approaches to deal with such problems. It is a type of data mining approach which sorts of detects local patterns in the data. RAGA, or Rule Acquisition with a Genetic Algorithm is one of the rule induction data mining techniques which attempts to solve our problem definition. It combines symbolic and evolutionary machine learning methods and extracts strong rules from a very challenging dataset. In our project, we are reproducing the paper "Evolutionary Data Mining With Automatic Rule Generalization"[1] and for that we use the "Poker Hand Data Set provided by UCI Machine Learning Repository"[2]. The paper uses the same dataset. The authors of the paper incorporate the RAGA algorithm and compare it with the traditional evolutionary search algorithm See-5. Systems such as See-5 cannot discover relationships between attributes and therefore are not capable of coming up with strong rules in many domains. They also state how RAGA is a more robust algorithm in terms of accurate predictions and memory size than the See-5.

2. Summary of original paper

The original paper uses the hard dataset: Poker Hand that the older version of RAGA couldn't handle it any better than See-5. In the paper, they tackle the dataset using the new RAGA which combines evolutionary and non-evolutionary machine learning models. Most part of the paper is a description of RAGA and its comparison with See-5. We will learn more about it in the following sections.[1]

2.1 Methodology of original paper

The authors of the paper discuss RAGA, a data mining system that uses Genetic Programming based engine to extract knowledge in the form of predictive rules. It points out the drawback of the more traditional and restrictive data mining systems such as See-5 and how evolutionary algorithms have a more global style of search. The below sections points out the difference between RAGA and new RAGA as stated by the paper.

2.2 RAGA

RAGA is an evolutionary data mining system that uses a hybrid GA and GP engine. The quality of the rule relies on various statistical and subjective factors which include the confidence, the coverage and how useful and interesting the rule is within the given domain. Since the latter two terms are subjective, there is no universally accepted method for determining them, and no common ground/scale to rate/compare them.

Confidence/Rule accuracy is the percentage of times that the consequent is true given the antecedent is true. If at all the consequent is false when the antecedent is true, the confidence for the given rule drops. Moreover, if the antecedent is not matched by data item, then that item doesn't contribute to the determination of the confidence of the rule.

Support/Rule coverage is the number of data elements that are correctly answered using the rule, divided by the total number of elements in the set.

The goal of RAGA is to find rules of the form:
If X1 ∧ X2 ∧ ... ∧ Xn Then Y1 ∧ Y2 ∧ ... ∧ Ym.

The symbols X1…Xn and Y1 ...Ym each defines terms within the rule, where term is a function that either shows the existence of a value, or perform an operation on two or more features/attributes. In a

classification task the value of m is always 1 and the value of n is unbounded. However, ours is a regression task!

The genetic engine of RAGA is a hybrid of GA and GP, with many modifications and additions to the standard models. One of which being the non-evolutionary component called Intergenerational Processing.

The intergenerational processing refers to several tasks that are run between generations. These tasks can vary depending on the user specifications, but it can include operations like trimming the dataset of rules that are logical tautologies or contradictions. It feature has been greatly expanded in the new RAGA after realizing its benefits.

2.2.1 New in RAGA

The previous methods like See-5 uses an evolutionary search and this is enhanced by intergenerational processing. Since intergenerational processing facilitates faster discovery of stronger rules the authors of the paper expanded its role to include machine learning techniques, in particular, rule generalization techniques that can be used to abstract higher level rules from groups of related and more specific rules.

The recent advances in RAGA was the design and implementation of a system to examine and generalize sets of rules. Generalization is known to have improved comprehensibility as can be seen from the poker hand data set below(scores below are from the paper).

Score	Name	Description	Example
16*	Royal Flush	Ace → Ten of same suit	**AH KH QH JH 10H**
15*	Straight Flush	Five sequential cards, same suit	**4C 5C 6C 7C 8C**
14*	Four of a kind	Four of the same card	**2H 2D 2S 2C** 8S
13*	Full house	Three of a kind plus one pair	**3D 3C 7H 7S 7D**
12*	Flush	Five cards of the same suit	**2C 3C 6C 9C AC**
11*	Straight	Five sequential cards	**3C 4C 5D 6H 7D**
10	Four to RF	Four cards towards Royal Flush	**AD KD QD JD** 4S
9*	Three of a kind	Three equal cards	**5H 5S 5D** 3C 7H
8*	Two pairs	Two pairs of equal cards	**4H 4S 9D 9S** 7C
7*	One high pair	Two equal cards (Jacks or better)	**QH QD** 2C KH 9S
6	Four to Flush	Four cards of the same suit	**2C 5C 7C 9C** 4H
5	Four to Straight	Four sequential cards (2 < c < A)	**3H 4D 5D 6C** AS
4	Three to RF	Three cards towards Royal Flush	**KD QD JD** 9S 8H
3	One low pair	Two equal cards (Ten or less)	**5H 5C** 7D JS 2D
2	Two high cards	Two cards (Jacks or better)	**JD QC** 2H 9C 5D
1	One high card	One card (Jack or better)	**AS** 4H 8D 3S 10H
0	Nothing	No useful cards in the group	2S 3D 6H 9C 10H

Figure 2.2.1 Scoring as per original paper

From the table, observe that the rules are not 100% confident over all possible hands because they do not exclude the case of a Full House, however the addition of new term is not needed to achieve a perfect accuracy to see the benefits of generalization.

If (R3 = R5) ∧ (R5 = R4) ∧ (R5 != R1) ∧ (R2 != R4) then (SCORE = 9)

If (R2 != R1) ∧ (R2 = R5) ∧ (R2 != R3) ∧ (R2 = R4) then (SCORE = 9)

If one examines the rules above, it will reveal that these rules are very specific to the positions of certain cards, which means that a large number of rules will be needed to fully describe the score tabulated above.

When RAGA comes across these rules which have the same consequent it will automatically attempt to generalize it. This is not done for every rule as the operation is computationally expensive, however the authors choose to rely on the evolution to choose the subsets that warrant further examination. The entire group of rules can be replaced by a single rule as follows:

If (NumEqualValues(class = rank) = 3) then (SCORE = 9)

When the technique formulated above is applied then a subset of rules will be compressed into a single, more general and more comprehensible rule. Additionally, it also increases the rule coverage.

2.3 Key results of original paper

The paper compares the results of See-5 and RAGA based on a number of experiments. They produced a set of classifiers: a decision tree for See-5 and a rule hierarchy for RAGA from a sample set of 10000 data. Thereafter, many test sets of size 1000 were used to test the classifiers. Below are the results for the them.

	Original Paper Results	
	Evolutionary search(See-5)	RAGA
Training Accuracy	64.25%	90.39%
Test Accuracy	36.16%	57.6%

Table 2.3 Original paper results

From the table above, the paper suggest that RAGA doesn't rely on the default hierarchy to boost the predictive accuracy while the poor quality of rules See-5 found id because the correctness of the class default values came from the assumptions made based on the more popular scores within the data. Additionally, tree it produced had 3946 nodes of which 3430 were leaves.

3. Methodology for report

For our project, we are incorporating the generalized RAGA method to our Poker Hand dataset. Moreover, unlike the paper we are also focusing on exploratory data analysis, feature engineering and implementing various models that gives the best prediction accuracy.

Later sections in the report elaborates more upon EDA, feature engineering and models. Before that lets take a look at the technical challenges, problem definition and dataset.

3.1 Objective and technical challenges

The aim of our project is to apply rule induction data mining technique, more specifically the improvised RAGA to our challenging Poker Hand dataset. Moreover, we will also be performing exploratory data analysis, feature engineering and apply several models and select the one with the best prediction accuracy.

4. Data set

Generating a set of random data that conforms to pre-specified set of known rules can be one way to address the problem of estimating results from a rule induction algorithm. The data could also contain corrupted(noise), incorrect, missing values, and trivial attributes. The quality of the algorithm can be evaluated by the correlation between the original and newly discovered rules. However, the major disadvantage of such method is that since the dataset is fabricated it holds no real-world meaning.[1][2]

Secondly, in the analysis phase it is meaningless to search for rules that are an identical match, but in cases where the newly discovered rules are different from the actual ones are sometimes difficult to determine its value. For instance, let's say that the new rule is an optimized version of the original one than it may be deemed incorrect since the human analyst might not really see the relationship.

Another challenge which the synthetic dataset poses is that it can represent a search space that is unbounded and unfair to learning algorithms. Similarly, some fabricated datasets cannot be correctly classified by the algorithms that are not capable of relational learning. One of the reasons for selecting the Poker Hand dataset is that it is difficult to find solutions but easy to analyze the same. Hence, we selected this dataset for our project.

The Poker Hand dataset was taken from the UCI Machine Learning Repository. The purpose of designing this dataset was to approach automatic rules induction i.e. to use machine learning to extract the rules. In

the dataset, each record implies a hand consisting of five playing cards which are not sorted by suit or rank from a standard deck of 52. The two attributes, suit and rank describe each card and there is a class attribute that describes the 'Poker Hand'. Moreover, the order of cards is very important, meaning there are 480 possible Royal Flush hands as opposed to just 4. Additionally, the dataset consists of 25010 training and 1,000,000 testing instances. As stated above, since the poker cards are not sorted by suit or rank, the total dataset has approximately 311.8 million instances, which is a pretty large number in terms of data. Furthermore, there are 10 predictive attributes and 1 goal attribute.

```
Attribute Information:
1) S1 ìSuit of card #1î
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

2) C1 ìRank of card #1î
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

3) S2 ìSuit of card #2î
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

4) C2 ìRank of card #2î
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

5) S3 ìSuit of card #3î
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

6) C3 ìRank of card #3î
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

7) S4 ìSuit of card #4î
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

8) C4 ìRank of card #4î
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

9) S5 ìSuit of card #5î
   Ordinal (1-4) representing {Hearts, Spades, Diamonds, Clubs}

10) C5 ìRank of card 5î
   Numerical (1-13) representing (Ace, 2, 3, ... , Queen, King)

11) CLASS ìPoker Handî
   Ordinal (0-9)

   0: Nothing in hand; not a recognized poker hand
   1: One pair; one pair of equal ranks within five cards
   2: Two pairs; two pairs of equal ranks within five cards
   3: Three of a kind; three equal ranks within five cards
   4: Straight; five cards, sequentially ranked with no gaps
   5: Flush; five cards with the same suit
   6: Full house; pair + different rank three of a kind
   7: Four of a kind; four equal ranks within five cards
   8: Straight flush; straight + flush
   9: Royal flush; {Ace, King, Queen, Jack, Ten} + flush
```

Figure 4. (a)Attributes and scoring information according to Poker Hand dataset

```
Statistics

  Poker Hand       # of hands       Probability     # of combinations
  Royal Flush      4                0.00000154      480
  Straight Flush   36               0.00001385      4320
  Four of a kind   624              0.0002401       74880
  Full house       3744             0.00144058      449280
  Flush            5108             0.0019654       612960
  Straight         10200            0.00392464      1224000
  Three of a kind  54912            0.02112845      6589440
  Two pairs        123552           0.04753902      14826240
  One pair         1098240  0.42256903      131788800
  Nothing          1302540  0.50117739      156304800

  Total            2598960  1.0             311875200

  The number of combinations represents the number of instances in the entire domain.
```

Figure 4. (b)Statistics as per Poker Hand Dataset

One important thing about Poker Hand dataset is that it can be readily mapped onto many other real world problem domains like resource allocation in a network, and it eases out rule evaluation even in unsupervised learning tasks.

5. Implementation

5.1 Exploratory Data Analysis

The points mainly belong to the classes 0,1,2 and 3 since in the actual poker game it more common to have instances where the five cards either are not recognised as poker hand or have 1 or 2 pairs. The scatter plots below show the distribution of the train and test datasets.

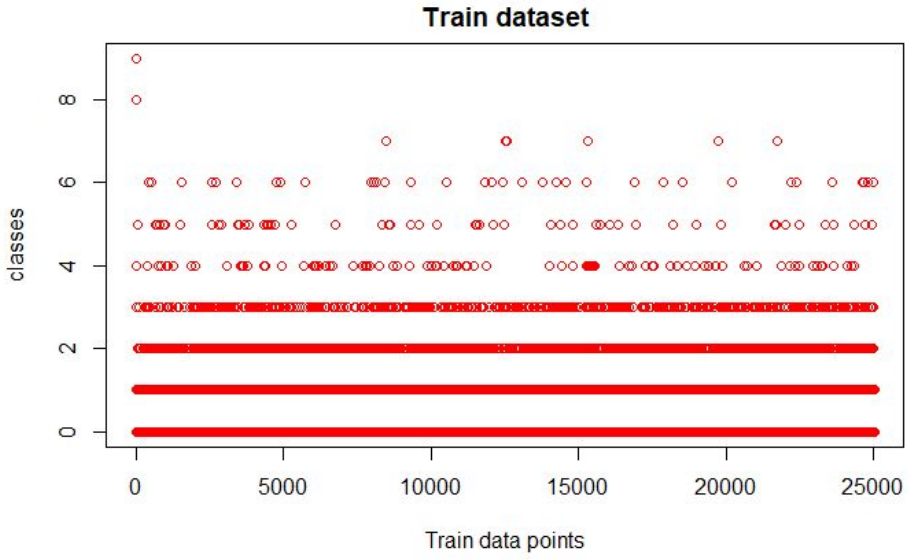

Figure 5.1(a) Distribution of train dataset

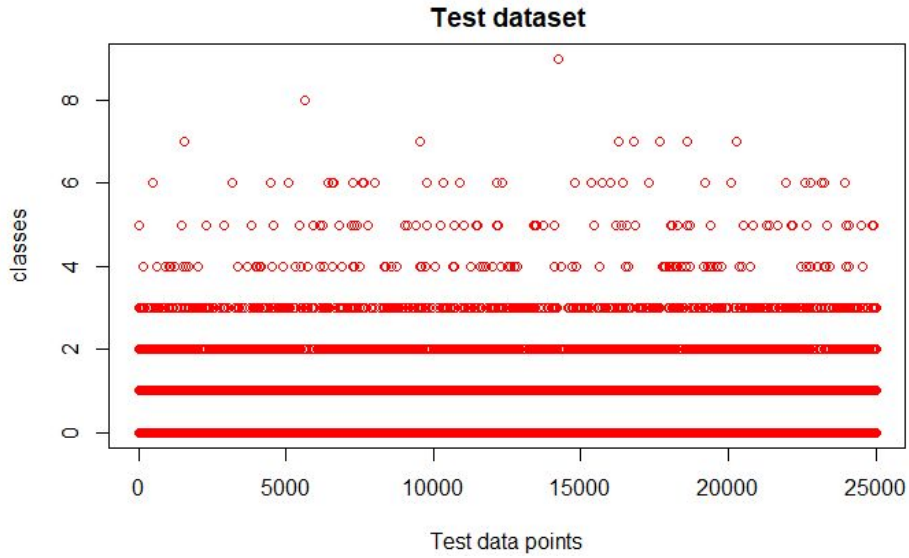

Figure 5.1(b) Distribution of test dataset

5.2 Feature Engineering

Consider the scenario below-

The training dataset consists of cards with classes and suites as given in the table below:

	Card 1		Card 2		Card 3		Card 4		Card 5		
Instance#	S1	C1	S2	C2	S3	C3	S4	C4	S5	C5	Class
1	Heart	10	Heart	K	Heart	Q	Heart	A	Heart	J	Royal Flush(9)
2	Heart	A	Spade	Q	Club	K	Heart	J	Diamond	10	Straight (4)

Table 5.2(a) Cards with suites and classes in training dataset

Now if the test data set consists of an instance as below-

Heart	A	Heart	Q	Heart	K	Heart	J	Heart	10

Table 5.2(b) Test dataset instance 1

When the above test instance is fed into a machine learning model it is most likely to be classified as a Straight instead of a Royal Flush. This happens because the positional component is impacting the decision making. But in the game of poker the position of the cards does not matter. Hence this positional component has to be removed.

5.2.1 Proposed Solution:

The best way to mitigate the effect of positional component is by restricting the order of the cards to make any influence on the prediction of rank. We propose to this by sorting the cards either by suite or by classes. In the above example, if we sort the cards in instance 1 then it will look like this-

	Card 1		Card 2		Card 3		Card 4		Card 5	
Instance#	S1	C1	S2	C2	S3	C3	S4	C4	S5	C5
1	Heart	10	Heart	J	Heart	Q	Heart	K	Heart	A
2	Heart	10	Spade	J	Club	Q	Heart	K	Diamond	A

Table 5.2.1 Sorting cards in instance 1

Now, when a test data is fed, the decision will be made on the sorted data and hence, the effect of position of the cards on prediction gets reduced.

5.2.2 Adding new features:

The features in the current dataset are mainly the class and suite of each of the five cards. To implement rule generalization which is described in RAGA, we investigated for features on which many of the poker hands might depend. Two of them are the maximum number of cards that belong to the same suite and the maximum number of cards that belong to the same class. Thus, we additionally added those features to our dataset.

We later performed feature importance to understand whether or not these additional features are influencing the decision. The results are shown below-

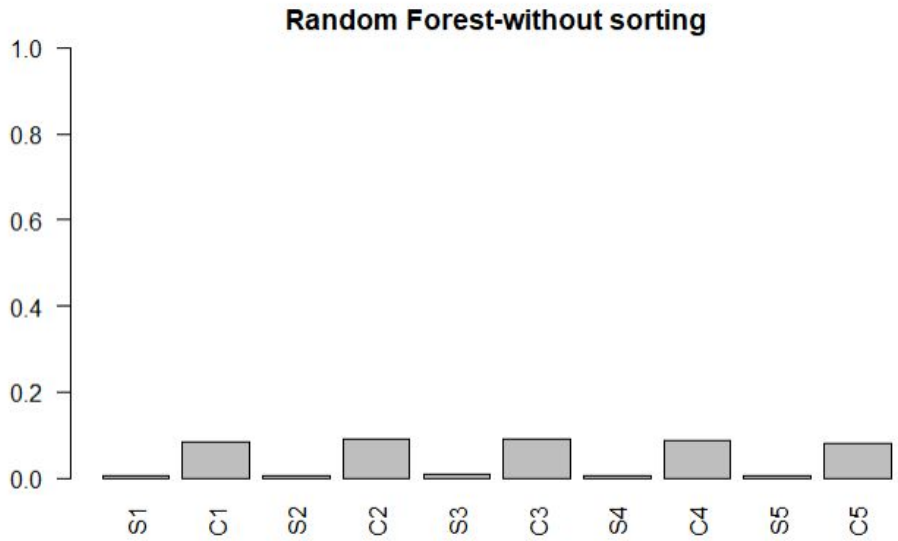

Figure 5.2.2(a) Random Forest-without sorting

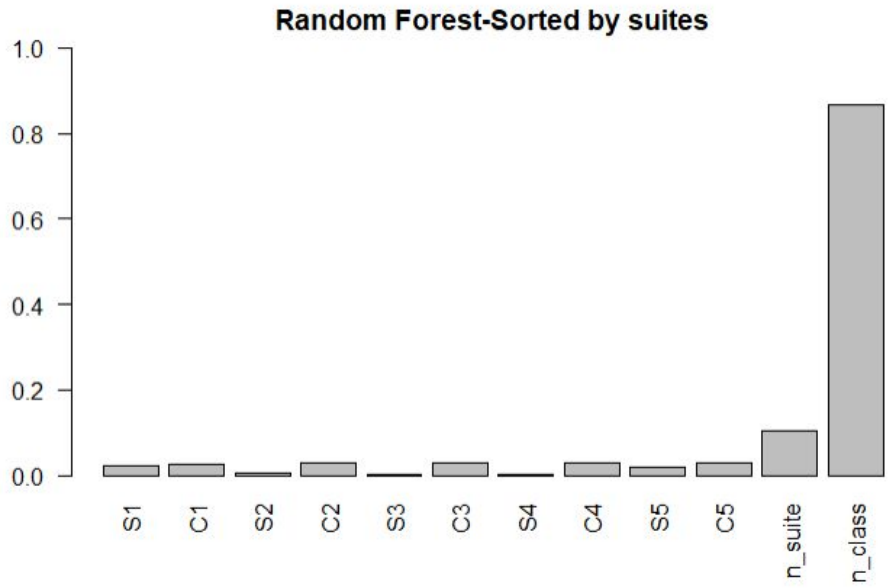

Figure 5.2.2(b) Random Forest-sorted by suites

As we can see in the above plot, the features 'n_suite' (maximum number of suites) and 'n_class'(maximum number of classes) have more importance than the suite and class of a card that was given in the dataset. This is because in the game of Poker the combination of cards is more important than a single card. Since the randomness (permutations) that classes can have is greater than the suite, n_classes has greater importance than n_suites but both of them are more important than individual card features.

6. Models

We implemented 3 models Multiclass Linear Regression, Random Forest (RF) and Support Vector Machine (SVM) in addition to the Decision Trees that was implemented in the original paper.

6.1 Scenario before feature importance and feature engineering-

a. Multiclass Linear Regression:

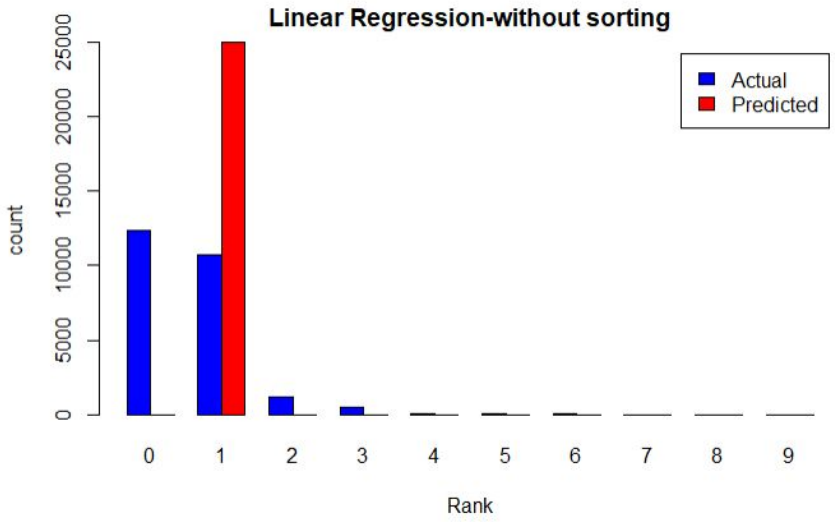

Figure 6.1(a) Multiclass Linear Regression-without sorting

b. Decision Trees:

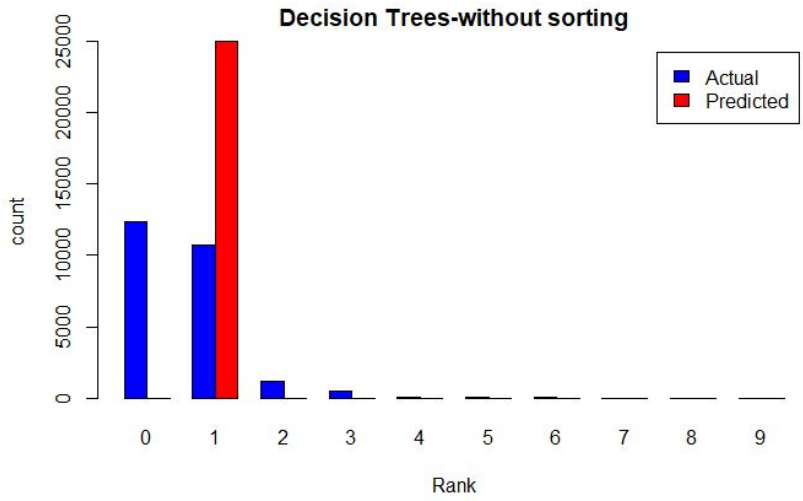

Figure 6.1(b) Decision Trees-without sorting

c. Random forest

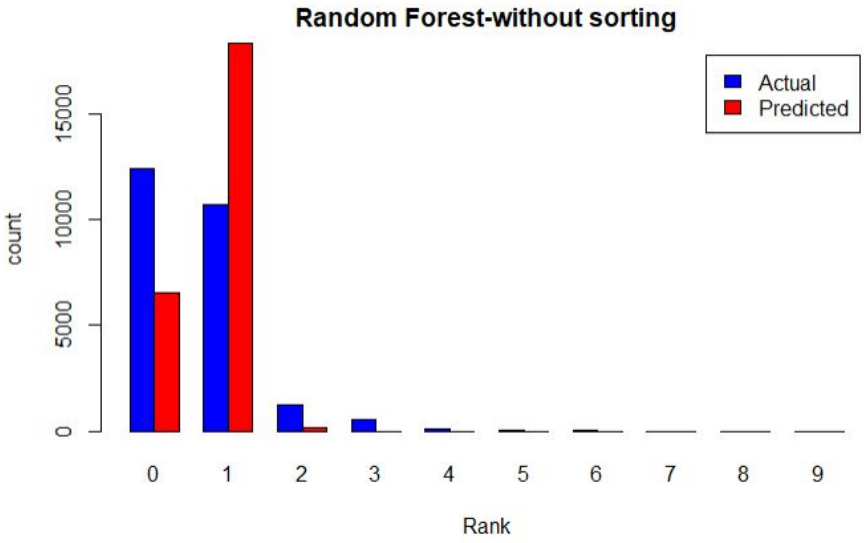

Figure 6.1(c) Random Forest-without sorting

d. SVM

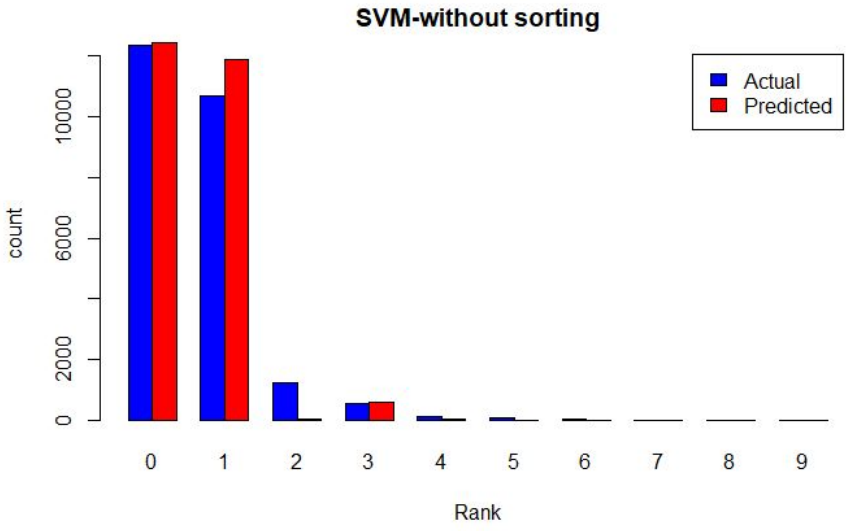

Figure 6.1(d) SVM-without sorting

As we can see from the plots above, none of the models were able to predict the rank of the card sets correctly, especially the ranks from 2-9. Most of the sets were getting classified as either rank1 or rank zero.

6.2 Scenario after feature importance and feature engineering-

On adding the new features, we get 2 kinds of datasets-

1. Dataset with new features sorted by class
2. Dataset with new features sorted by suite

6.2.1 Dataset with new features sorted by class

a. Multiclass Linear Regression

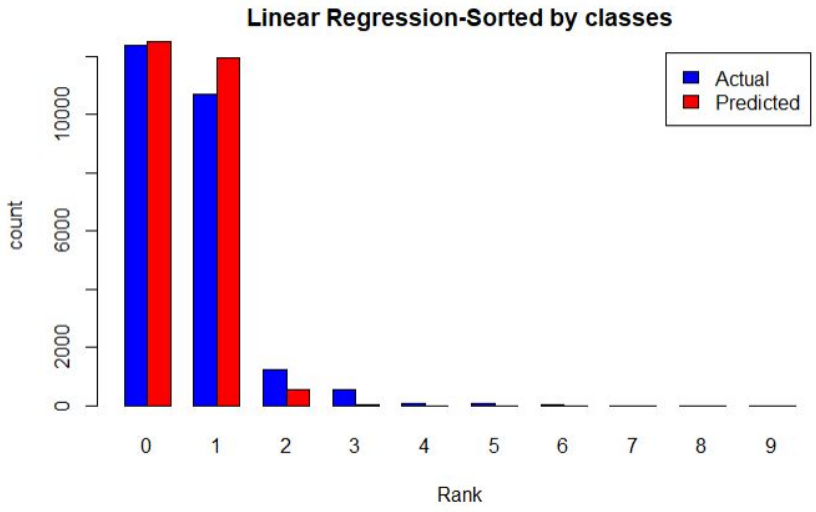

Figure 6.2.1(a) Multiclass Linear Regression-sorted by classes

b. Decision Trees

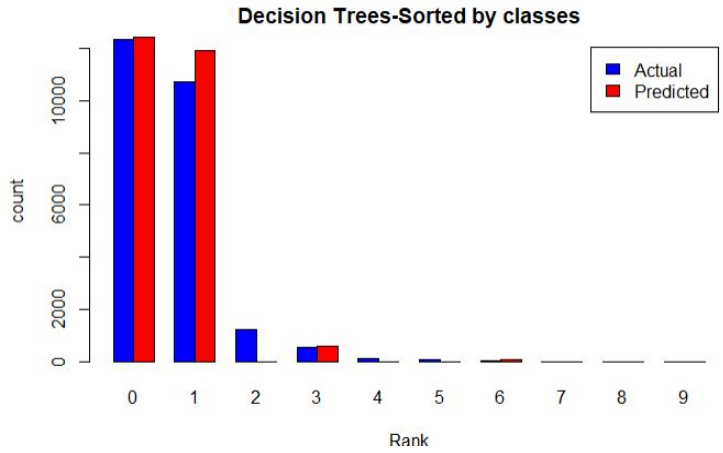

Figure 6.2.1(b) Decision Trees-sorted by classes

c. Random forest

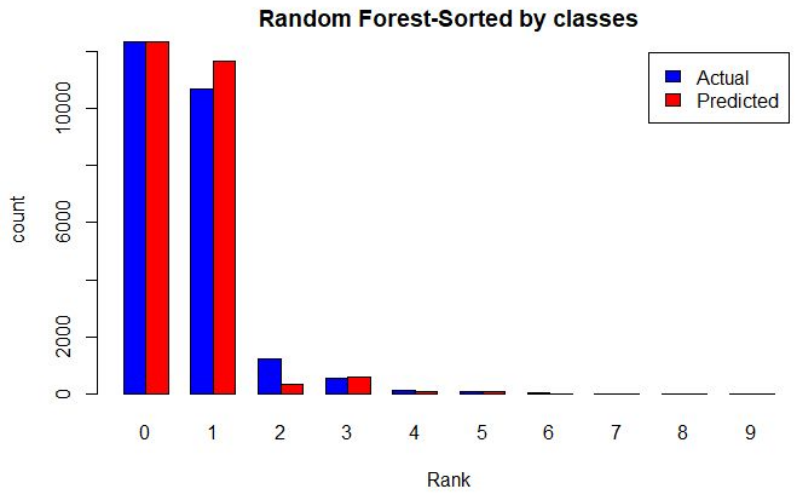

Figure 6.2.1(c) Random Forest-sorted by classes

d. SVM

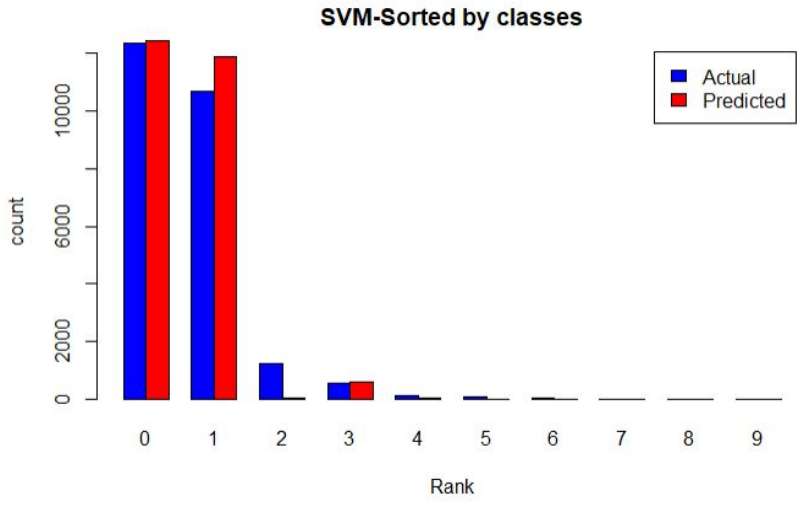

Figure 6.2.1(d) SVM-sorted by classes

Sorting the cards by class improved the prediction for the ranks 2-9. This is clear from the above plots. The random forest predicted the rank of the set of cards most accurately

6.2.2. Dataset with new features sorted by suite

a. Multiclass Linear Regression:

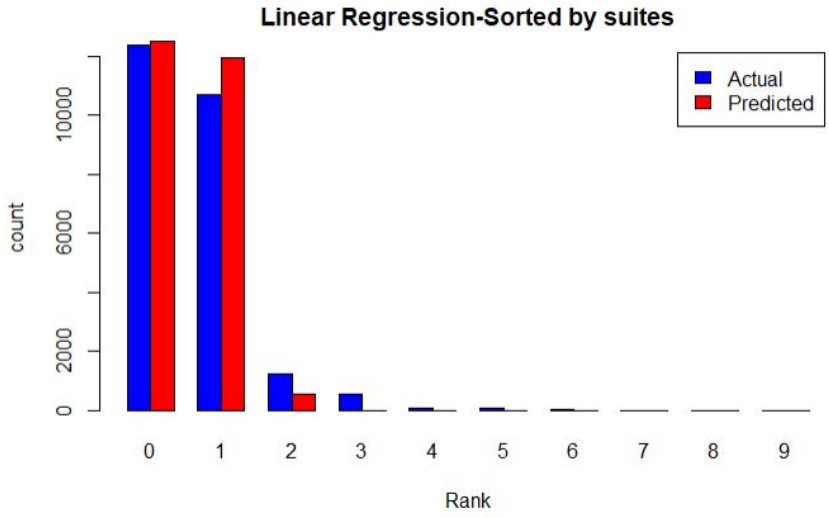

Figure 6.2.2(a) Multiclass Linear Regression-sorted by suites

b. Decision Trees:

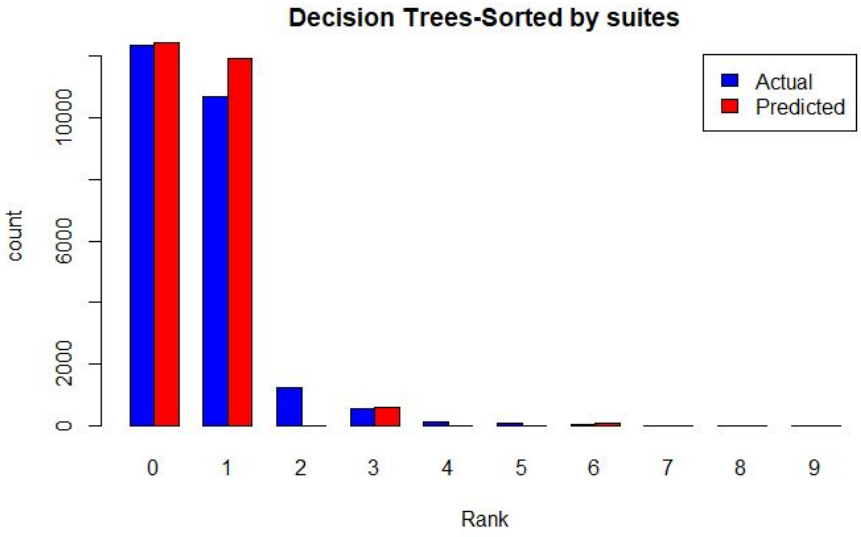

Figure 6.2.2(b) Decision Trees-sorted by suites

c. Random forest

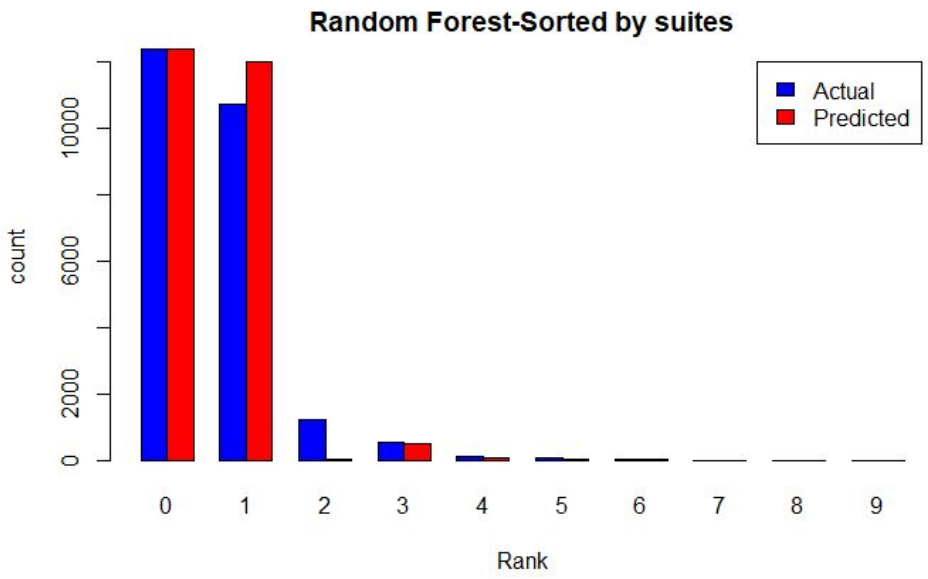

Figure 6.2.2(c) Random Forest-sorted by suites

d. SVM

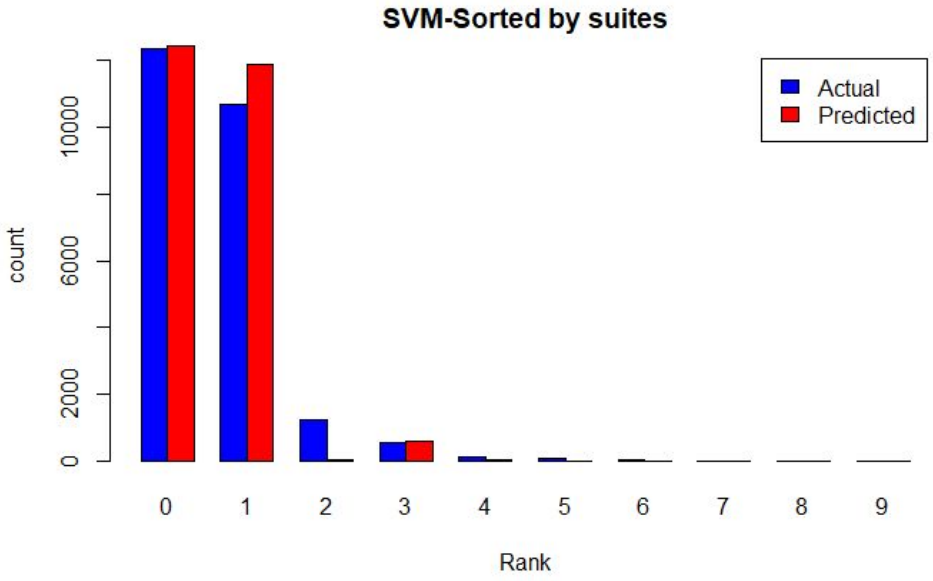

Figure 6.2.2(d) SVM-sorted by suites

7. Results

7.1 Before feature engineering and feature importance:

Since most of the sets were getting classified as either rank1 or rank zero, the test accuracy was as low as 42.83% using the Decision Tree classifier and the maximum test accuracy achieved was 55.67% using SVM classifier.

Model	Train classification Accuracy	Test Classification Accuracy
Multi Class Linear Regression	42.38	42.83
Decision Trees	42.38	42.83
Random Forest	58.20	58.39
SVM	60.22	55.67

Table 7.1 Before feature engineering and feature importance

7.2 After feature engineering and feature importance:

The highest accuracy achieved after sorting was 95.41% using random forest and the lowest was 92.36% using multiclass logistic regression. Comparing the models before and after sorting, we can observe a clear difference between the maximum accuracy before sorting(55.67%) and the minimum accuracy after sorting (92.36%).

7.2.1 Sort by class-

Model	Train classification accuracy	Test Classification accuracy
Multi Class Linear Regression	92.44	92.36
Tree Regression	94.45	94.38
Random Forest	**95.56**	**95.41**

SVM	94.50	94.42

Table 7.2.1 After feature engineering and feature importance-sort by class

7.2.2 Sort by suite-

Model	Train classification accuracy	Test Classification accuracy
Multi Class Linear Regression	92.33	92.26
Tree Regression	94.38	94.36
Random Forest	94.21	94.17
SVM	**94.5**	**94.42**

Table 7.2.1 After feature engineering and feature importance-sort by suites

8. Comparison of our results with the original paper

	Original Paper results		Our project results(best classifier)	
	Evolutionary search(See-5)	RAGA	Baseline	After feature engineering
Training Accuracy	64.25%	90.39%	60.22%	95.56%
Test Accuracy	36.16%	57.6%	55.67%	95.41%

Table 8 Comparison of our results with the original paper

9. Discussion and insights gained

1. Though the number of records went down with increase in poker hand points, we decided not to deal with imbalance as we didn't want to interfere with the game feature(occurance of higher poker hand is relatively very less probable).

2. Sorting with respect to class is better than with respect to suite. This is because of higher variance in class compared to suite.
3. Induction of rule as mentioned in RAGA is done by extracting appropriate features which resulted in drastic improvement of model's accuracy as observed above.

10. Conclusion

It is very easy for humans to instantaneously ignore positional component. However, for a machine to learn this, we have to train it with all possible permutations of a particular combination. This in case of games like poker or chess where there are humongous number of possible permutations, would not be practically feasible to implement. Thus RAGA based rule induction helps to overcome such issue.

11. 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] https://archive.ics.uci.edu/ml/datasets/Poker+Hand