Project report
on
Poker Prediction using Machine Learning

By,

Arish Balasubramani (304455611)
Cheryl Maria Jose (305052753)
Rajaputri Maharaja (305059344)
Prasanth Kannan (305059149)
Prasanth Reddy Yerradoddi (304387751)

ABSTRACT

In this project, we are asked to predict the best hand we can play based on the cards we have been dealt.
The order of the cards is important, which means there are 480 possible Royal Flush hand instead of just four. The intent of this project is automatic rules induction i.e, to learn the rules using machine learning, without hand coding heuristics.

We will be predicting the outcome of the poker hand. We take each hand in the testing set and find out the features for each hand. Following are few features we have included in this project:

1. Number of same suit cards.
2. Highest card.
3. Number of same value cards.
4. Number of cards in sequence.

1.INTRODUCTION

Poker is a family of gambling card games, but is often considered a skill based game. All poker variants involve betting as an intrinsic part of play, and determine the winner of each hand according to the combinations of players' cards, at least some of which remain hidden until the end of the hand. Poker games vary in the number of cards dealt, the number of shared or "community" cards, the number of cards that remain hidden, and the betting procedures.

2.IMPLEMENTATION

In this project, we will be predicting the outcome of the poker hand. From testing set we take each hand and find the features for each hand.

1. Number of same suit cards.
2. Highest card.
3. Number of same value cards.
4. Number of cards in sequence.

Number of same suit cards:

This function will be able to find out the number of same suit (Spades, Hearts, Clubs, Diamonds) cards. Each hand has exact 5 cards, of those 5 cards there can be any number of same suit cards. Each suit is given a number. To find the number of same suit cards we used the **set()** in python. **set()** stores the unique collections of values. For example, if a hand has 3 cards of same suit and other 2 cards are of different suit, then the set will store only 3 values. Based on the size of set, we will be able to tell the number of same suit cards.

```
def number_of_same_suit_cards(b):
    if len(set(b)) == 1:
        return 5
    elif len(set(b)) == 2:
        return 4
    elif len(set(b)) == 3:
        return 3
    elif len(set(b)) == 4:
        return 2
    else:
        return 0
```

Highest card:

Highest card function will find the highest card among the 5 cards in hand.**max()** inbuilt function is used to find the highest card. Ace, King, Queen, Jack are given numbers 14, 13, 12, 11 respectively.

```
def highest_card(a):
    return max(a)
```

Number of same value cards:

This function will return the count of same numbered cards. It is similar to number of same suit cards. Here too, we use the **set()** function to find the number of same numbered cards.

```
def number_of_same_value_cards(a):
    if len(set(a)) == 2:
        return 4
    elif len(set(a)) == 3:
        return 3
    elif len(set(a)) == 4:
        return 1
    else:
        return 0
```

Number of cards in sequence:

This function will find the number of cards in sequence. When the cards are dealt, they are not in order. So, the sorting function is used to sort the cards, which makes it easier to find the sequence of any 5 cards cards. After the cards are sorted, we compare a card to its next card to see if the cards are in sequence. We use the counter variable to keep the count of cards in sequence. The cards are compared from the first to the last in ascending order. It is a very rare occasion that all the 5 cards are in sequence. Most of the time only two or three of these cards are found to be in sequence.

```
def number_of_cards_in_sequence(a):
    counter=0
    if(a[1] == a[0]+1):
        counter = counter+1
    else:
        counter = 0
    if(a[2] == a[1]+1):
        counter = counter +1
    else:
        counter = 0
    if(a[3] == a[2]+1):
        counter = counter+1
    else:
        counter = 0
    if(a[4] == a[3]+1):
        counter = counter+1
    else:
        counter = 0

    return counter
```

4.EXPERIMENT:

All the 4 features were made into a table using pandas DataFrame. To this dataframe we also added the hand column which was available in training data set. After the extraction of these features we used knn algorithm to train the relationship between extracted features and 'hand' from training set. The feature table is randomly split into 60% and 40% for training and testing respectively. The accuracy was predicted using the 40% of randomly split data from feature table. We were able to achieve an accuracy of more than 95%.

We also created a new pandas dataframe for testing set. Testing set does not contain the hand which is available in the training set. So we followed the same procedure to extract the features and put it into a new feature table. Since we had used the knn algorithm to train, we used same algorithm to predict the hand for each row in the testing set. We also calculated the accuracy of testing set after training the data set using knn and decision tree. Our accuracy came up to 98% also predicting using decision tree and validating accuracy between predictions achieved using knn and decision tree.

5.CONCLUSION:

Based on the features extraction method that we implemented we were able to achieve an accuracy of more than 95%.

DATA SET

1. https://raw.githubusercontent.com/PrasanthReddyYerradoddi/Machine-Learning/master/train.csv
2. https://raw.githubusercontent.com/PrasanthReddyYerradoddi/Machine-Learning/master/test.csv

References

1. https://www.kaggle.com/c/poker-rule-induction
2. https://www.tutorialspoint.com/python/