There are three stages in my classification algorithm (all is my own idea).

The first stage deals with hands that have identical cards as hands in the training data. In the training phase, it is determined whether the order of a particular hand matters. Specifically, if there is another hand with the same cards, but with different order, and their rank is different, then the fact that order matters is stored along with the hand data in the database, as well as their corresponding rank. Although generalizations can be made to the rank of hand about order dependence, it is possible for other card games to have specific rules concerning order. There is also the issue of how machine learning can be used to interpolate rules describing order dependencies among cards of the same rank. Consequently, generalizations were not made and order is handled on a case-by-case basis. It is expected that the loss of scrutiny in order dependent rules at this point of the algorithm will not impact the accuracy of the classification algorithm significantly.

The second stage carries almost the entire bulk of the classification burden. Nearly the entirety of hands is classified in the second stage. The idea of this stage is using pattern matching. A hand is stripped of its numerical value, but maintains its numerical boundaries and suit information. Firstly, a hand is transformed into a bitmap. 4 bits represent the 4suits of a card rank, and 52 bits are used to represent the entire card space. Each hand contains 5 cards of which 5 bits in the 52-bit bitmap are set. The first bit, which is set, is then moved to the first card rank position (in this case, the ace position), and all the other bits also receive the same relative translation. This is then stored in a corresponding hash table with its order significance. Order dependency is determined in a similar fashion as described in the first stage. During classification, the exact pattern match as well as order match is queried. If the query fails, the algorithm finds the next best match according to the order of the cards. If a match is not found even when the order does not matter, the query fails. It should be noted that the strictness of the order match, that is when the algorithm ends, can be adjusted. Additional classification algorithms can be built on top of this; however, this already achieves acceptable performance. As a result, no further novel algorithms were designed.

Finally in the third stage, the remaining hands are characterized by descriptive statistics. In the training phase, each rank’s descriptive statistic is computed, and the best match is found with the descriptive statistic match of the test data. Although optimizations can be made to improve the performance of this stage, the result will not improve the accuracy by a significant amount. If a match is not found, then the rank of the test data is assigned as the highest occurring rank in the training data. The third stage typically accounted for less than 0.1% of the classification. The no match case accounted for 0.3% of the classification. Overall, the accuracy could be improved by 0.5% through optimizations. To improve the performance significantly, the added algorithm must outperform the pattern matching described in stage 2.

Program usage: ./main <train.csv> <test.csv> <output.csv>

<> denotes the name of the file