William Schimitsch and Eli Werstler

Professor Mohsin

Data Mining

8 December 2023

Performing Classification on a Poker Hand Dataset

Our project utilized the Poker Hand Dataset, created by Franz Oppacher of Carleton University. The dataset consists of 11 attributes, including 10 predictive attributes, and 1 goal attribute. The predictive attributes represented the rank and suit of 5 cards, with a rank attribute and suit attribute for each card. The goal attribute was the class of the poker hand, from 0 or 1 meaning ‘nothing in hand’ and 1,’one pair’, respectively, up to a 9, which is a ‘royal flush’. While the dataset provided a testing set of 1,000,000 instances and a training set of 25,010 instances, we combined these two sets into one to split the data ourselves. When we first started looking at the data, we wanted to see the distribution of the poker hands. We achieved this using the pandas function “value_counts” to count how many of each class were in the data set. We found that over half of the hands resulted in ‘nothing’, while only a few hands were royal flushes. The vast majority of hands were classified as either ‘nothing in hand’ or ‘one pair’. This meant that we were going to have to later split our data in a stratified manner, to ensure that we would train and test on all types of classes. Our mean, median, mean absolute deviation, standard deviation, and interquartile range calculations yielded unsurprising results. The means and medians were around the center values of each attribute range, and the deviation values indicated similar variability across comparable attributes. Furthermore, when detecting outliers using the “.quantile()” function, the calculated bounds yielded no outliers, which was expected from the data. When calculating the covariances between attributes, we found that no two unique

attributes had significantly positive or negative covariances, and thus none were dependent. The multivariate gaussians of the cards in each hand, however, showed a correlation between the rank and suit of each card. This follows from the natural correlation between the two features, as they are shared by each card. To conclude our findings from our numeric analysis, the poker data set provided a comprehensive representation of poker hands. There was no missing data and no outliers that would skew the results.

There were three different algorithms we were looking at to perform our classification task on the poker hand dataset: k-nearest neighbors, decision trees, and neural networks. K-nearest neighbors was our original plan, as it typically works well in classification tasks, could understand the dataset's multidimensionality, and would be easy to implement. Further, in order to ensure all classes were represented in the training data, we used k-fold cross validation for k-nearest neighbors and averaged the accuracies and F1 scores to assess its performance. After initially performing classification using this algorithm, we were getting slow runtimes and low accuracy and F1 scores: ~60% for each. Thus, we tried using decision trees instead, as we expected a lower computational cost and perhaps a more accurate model that could better understand the relationship between the poker hands and their value in the game. While the classification task executed much quicker using decision trees, we were left with no different performance than that of KNN. Further, changing the hyperparameters of the models did not help their performances: enforcing a maximum depth did not change the performance of the decision tree model, and implementing a minimum impurity decrease drastically brought down the accuracy. Similarly, trying different values of k for KNN changed practically nothing about the model's performance. Now, knowing the ability of neural networks to recognize patterns and have incredible performance rates, we tried implementing them here. While we saw great

improvements in performance—the accuracy and F1 scores were now in the high 90s, pushing above 99%—the runtime was incredibly slow, taking up to five times longer than KNN. Testing with smaller train splits (as small as 5%) made the algorithm run faster, however it sacrificed the performance of the model.

We discussed these results with Professor Mohsin, who suggested a way to reduce the dimensionality of the dataset by considering all permutations of a hand, or different ways to order a given hand, as the same hand. Since there are $5! = 120$ ways to order a hand of 5 cards, this would reduce the dimensionality of our set by exactly 120, which meant k-nearest neighbors could run both faster and more accurately. In order to implement this, we sorted the data set by card, first by each card's suit, then by its rank. In python, this meant putting each card's suit and rank value into a tuple, so that when we sorted the dataset, the respective values for these features were not scrambled. Then, we sorted these arrays of tuples, and finally reverted the tuples back to two, independent values. After this, each permutation of a given hand in the dataset looked exactly the same.

Sorting our data improved our results drastically: our k-nearest neighbors implementation executed quicker (which could, in part, be due to using the scikit learn KFold library for cross validation instead of the more costly StratifiedKFold) and with an average accuracy and F1 score both above 80%. The model could now better understand the relationship between the input and output features because we were now treating hand permutations equally, which meant hands of similar class would now be closer in distance to each other. Similarly, decision trees saw improvements of these values to 70%. Thus, we could now get reasonable predictions, but in a fraction of the time. To further examine the capacities of KNN, we did further testing on hyperparameters, and in particular on the optimal k value to use. The different k values did not

alter the model's performance drastically, though $k = 4$ and $k = 6$ posted the best performances, with accuracies of approximately 82%.

In performing classification tasks on this Poker Hand Dataset, it became clear how important data preprocessing would be, especially for a dataset with over a million instances. Now, normalization actually did not help in this case—since our rank features are ordinal and its range is not drastically different from that of the suit features, normalization actually negatively impacted our model's ability to predict the correct classes. Instead, reducing the dimensionality was key, as we do not care about the order of the cards here. That is, in poker, a two pair is a two pair, no matter which way you order the cards. With this, then, our models, particularly k-nearest neighbors, could better understand the data and the relationship between the hands and their respective value in the game of poker.

Collaboration Log

We both worked fairly evenly on each of the tasks. For the numeric analysis, Eli did the first half, up to and including the covariance graphs, and Will did everything else. We then worked together on our classification of the data. In the end, Eli handled the Neural Network classification and Will sorted the data, which led to faster and more accurate KNN and decision tree classification. For the essay, Eli wrote about numerical analysis and Will wrote about our high level tasks.

Works Cited

Throughout the project, we used a lot of the code and direction from homeworks 2, 3, and 4. We also used the numpy, pandas, pyplot, and scikit learn documentation, linked below.

https://pandas.pydata.org/docs/

https://numpy.org/doc/stable/

https://scikit-learn.org/stable/index.html

https://matplotlib.org/3.5.3/api/_as_gen/matplotlib.pyplot.html