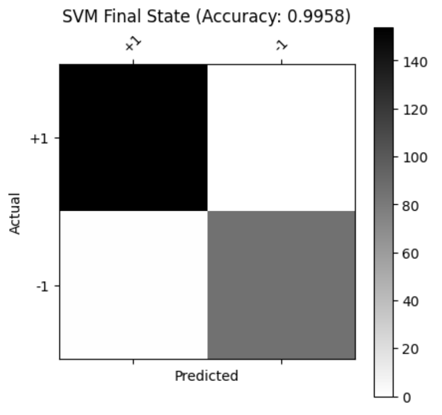
Samuel Maley
3/3/2024
Intro to ML

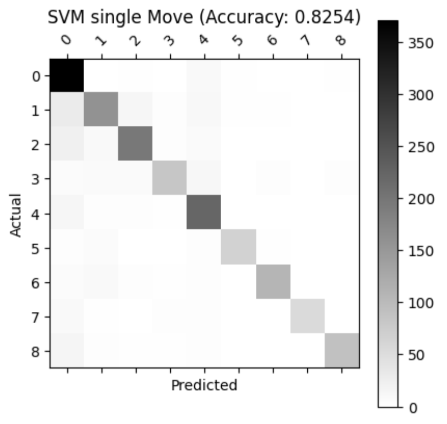
Explanation of implementation

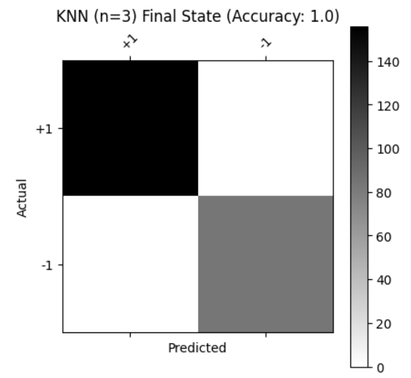
My code was implemented in a jupyter notebook. I used the pandas, numpy, and scikit-learn libraries to import, process, and build models on the given tic-tac-toe data. For the classifiers, I first created functions to implement a pipeline for easy comparison. I compared the Linear SVM, K-nearest-neighbors (KNN), multilayer perceptron models (MLP). For the regressors, I used KNN, MLP, and linear regression. These models were tested on the data provided to me, which was a dataset containing final board states, optimal single moves, and optimal multiple moves. The classifiers were trained and tested on the former two while the regressors were tested on the third. The scikit-learn library was used to implement these models, and my code displays a few metrics for measuring the performance of the models. After training the regression models, they were saved using the “joblib” library for implementation in the command line tic-tac-toe game.

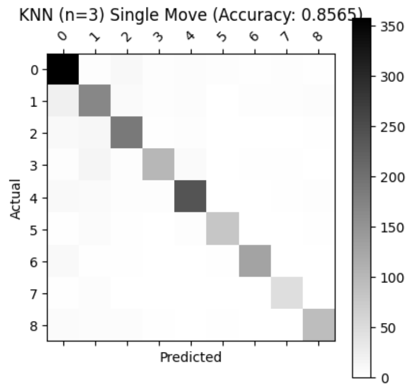
Evaluation results

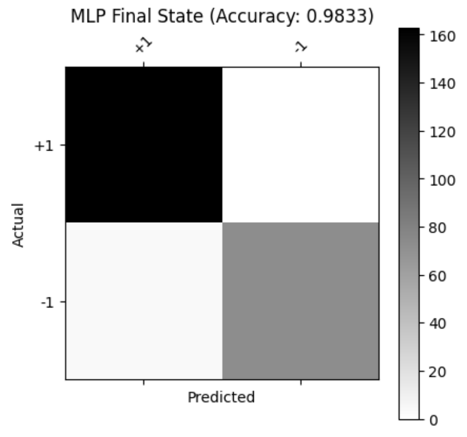
For the classifiers, I found that the MLP and SVM performed the best. With MLP getting an accuracy of 98% on the final board state data, and 92% on the single move data. KNN was close behind with an accuracy of ~100% and 85%, respectively. Additionally, SVM had an accuracy of 99% and 82% on the same datasets. Due to the nature of the datasets, I only considered their performance on the single move datasets as a benchmark for their accuracies. All models performed very well, all scoring above 80% accuracy on the single move datasets. Due to computing limitations on my machine, MLP was limited to a max number of iterations at 1000. It is assumed that the model would perform better at a higher number of iterations. Furthermore, I use a “n” value of 3 for the KNN model.

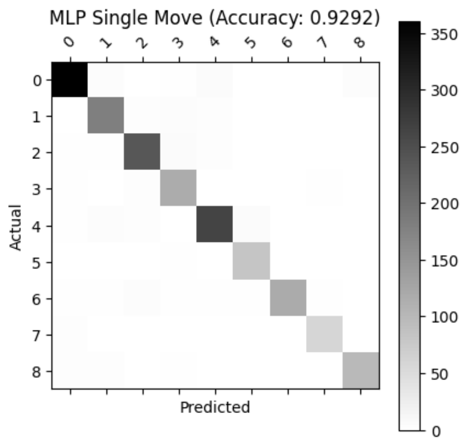












For the regression models, I found similar results. The outputs of the regression models were rounded to match the binary nature of the tic-tac-toe dataset. An output of 0.5 or above was rounded to 1, while a value lower was defaulted to 0. This allowed me to concisely compare the accuracies of each of the models. MLP and KNN were the highest performers and linear regression fell far behind. On the optimal multiple moves dataset, MLP boasted an accuracy of 83% and KNN of 71%. As mentioned, linear regression fell far behind with an accuracy of ~0%. As seen by the tables, the root mean square error is positively correlated to the accuracy.

Model:	Accuracy	RMSE
KNN	0.7118437118437119	0.2699
Linear Regression	0.0	0.4587
MLP	0.8376068376068376	0.1654

These results make sense, as the linear regression and linear SVM models are the simplest of all the techniques utilized. MLP is the most complex method of calculating these results, and as a consequence, takes the longest to train. KNN sits in the middle, and is less accurate than MLP, but takes significantly less time and resources to train.

My implementation of tic-tac-toe in the command line exemplifies these differences and provides a hands-on opportunity to grasp the predictive powers of each model. As expected, MLP and KNN prove near impossible to beat, while linear regression can sometimes make mistakes.

Instructions on how to run your code

My code can be run with any version of python that meets the requirements of the given libraries: pandas, numpy, matplotlib, and jupyter. After these libraries are installed and all the source files are in the same folder. There should be no issue running my provided code.

Any bugs, difficulties you would like us to know

I had difficulty getting the multi output linear regression model to work, and it still gives an accuracy of zero when run. I experimented with different ways of tackling this issue, but no methods proved to fix it.