# Part 1 Classifier and Regressors - Tic Tac Toe

# Implementation details

We implemented all algorithms in Python using scikit learn libraries for the machine learning models. For the multi-label regressors, we regress on each class independently and round the predicted value to be either 0 or 1, thus leading to 9 total regressors per model.

# Evaluation Results

Below are the accuracy results for each model, where the accuracy is calculated using *accuracy\_score(y\_true, y\_predicted)* from scikit learn. This function calculates exact matches for y\_true and y\_predicted, thus it’s calculating the total correct divided by the total number of samples.

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifiers** | KNN | SVM | MLP |
| Final Boards Classification | **100%** | 99% | 99% |
| Intermediate Play (Single Label/Multi-class) | 76% | 84% | **96%** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Regressors** | KNN | Linear Regression | MLP |
| Intermediate Play (Multi Label) | **91%** | 78% | **91%** |

A picture containing square

Description automatically generatedA picture containing chart

Description automatically generatedA picture containing chart

Description automatically generated

Graphical user interface, application

Description automatically generatedTimeline

Description automatically generatedGraphical user interface, application

Description automatically generated

KNN was best for the Final Boards dataset at 100% accuracy, but the other models achieved 99% accuracy, thus a negligible difference. The MLP performed the best for the multiclass single label dataset at 96% accuracy. For the regressors, MLP and KNN achieved equal accuracy of 91%. MLP achieves high accuracy across all datasets due to their ability to model non-linear relationships between inputs and outputs. KNN is able to perform well since it uses a small neighborhood (k = 5) for classification and regression.

#### Training with 1/10th of the data

|  |  |  |  |
| --- | --- | --- | --- |
| **Classifiers** | KNN | SVM | MLP |
| Final Boards Classification | 72% | 77% | **85%** |
| Intermediate Play (Single Label/Multi-class) | 57% | 66% | **72%** |

Chart, treemap chart

Description automatically generatedChart, treemap chart

Description automatically generatedChart, treemap chart

Description automatically generated

KNN performed the worst with less data, this may be due to KNN’s instance-based approach for classification. KNN looks at a neighborhood of points to make predictions, but if the current point isn’t surrounded by similar samples (due to a lack of data) then it will not perform well. MLP are great for modeling non-linear relationships between the input and output, and thus performed best out of the two datasets, across all methods. Similarly, for the multiclass data.

We shuffled the data and took the first 10% of samples, therefore our KNN didn’t get above 80%, but it might have for a different random seed shuffle.

Graphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generatedGraphical user interface, application

Description automatically generated

#### Can we beat the computer? How hard is this for the three regressors?

We are using the multi-class MLP, which achieved a 96% accuracy, therefore it almost always achieves the correct answer. We ran 5 experiments for the MLP classifier and each of the three regressors. All games resulted in draws except for 1 game against KNN Regressor, where the algorithm won, and we won all 5 games against linear regression. Therefore, it’s easy to win with the Linear Regressor but the other regressors and the MLP classifier did as good as a human and all this considering that the human plays first.

# How to run programs

To evaluate the 3 classifiers on two datasets and regressors on the multilabel dataset, you can run the following code after changing to the directory where Part1\_TicTacToe.py resides.

#### Classifiers and Regressor Evaluations

*$ python Part1\_TicTacToe.py*

*> Would you like to train (t) or evaluate (e)?*

Type **e**, to **evaluate accuracy for the already trained** models (training takes a long time and is not feasible for grading, unless you would like to train them, then press **t**).

#### Tic-Tac-Toe AI Game

*$ python TicTacToe\_AIGame.py*

Then enter a number between 0 and 8 to play against the best ML classifier, the multiclass MLP.

# Other comments

Everything works and there are no known bugs.