Part I. Classifiers and Regressors

Shantanu Ghosh (4311-4360) Kalpak Seal (8241-7219)

October 8, 2020

1 Introduction

This project evaluates the performance of classification and regression algorithms on a game of Tic Tac Toe.

2 Program Execution

The code has been developed in **python 3.7**. Details of other packages have been mentioned in **./requirements.txt** file for reference. Also we encounter a future warning while invoking the GridSearchCV of sckit-learn package.

1. To train the models
   python3 buildModels.py
2. To play the game
   python3 tictactoe.py

2.1 Dataset path

All the dataset paths have been assigned to the variable **final_dataset_path** in the file **./buildModels.py**.

2.2 Saving the optimal model for the game-play and plots

The accuracy vs k value plots will be found at the folder ./Plots. The most optimal model has been saved as the file **./model_params.pkl** and will be used to play the game.

3 Implementation

We split each of the 3 datasets as 80% train and 20% test sets. Then we used 10-fold cross validation to the training set to determine the optimal parameters for the model and used the model on the test set to report the test accuracies.

4 Classifiers

4.1 Cross-validation

4.1.1 Linear SVM

Table 1 and 2 shows the different parameter combinations with the mean 10-fold train and validation accuracies of $tictac\_final$ (final board classification) and $tictac\_single$ (Intermediate boards optimal play) datasets respectively.

4.1.2 k-nearest neighbors

For the k-nearest neighbors, we take a range for the values of k from 1 to 100 incremented by two steps. Then we plotted the mean validation accuracies of the 10 fold cross validation vs k values to get the optimal k. Figure 1 shows the plots for $tictac\_final$ (final board classification) and $tictac\_single$ (Intermediate boards optimal play) datasets respectively (from left to right).

4.1.3 Multilayer perceptron

We used 10-fold cross validation to find out the best possible combination of hyper-parameters for Multilayer perceptron (MLP) classifier. Table 3 and 4 shows the different parameter combinations with the mean 10-fold validation accuracies of $tictac\_final$ (final board classification) and $tictac\_single$ (Intermediate boards optimal play) datasets respectively.

Parameters	Train Accuracy(%)	Validation Accuracy(%)
C: 10, gamma: 1	**97.8**	**98**
C: 10, gamma: 0.1	97.4	98
C: 10, gamma: 0.5	97.7	98
C: 12, gamma: 1	97.1	98
C: 12, gamma: 0.1	97.8	98
C: 12, gamma: 0.5	97.9	98

Table 1: Accuracy scores of SVM classifier on **tictac_final (final board classification)** dataset. All the accuracies are reported based on the mean score of 10 fold cross validation

Parameters	Train Accuracy(%)	Validation Accuracy(%)
C: 10, gamma: 1	**39.5**	**39.02**
C: 10, gamma: 0.1	39.5	39.26
C: 10, gamma: 0.5	39.5	39.02
C: 12, gamma: 1	39.5	39.26
C: 12, gamma: 0.1	39.5	39.17
C: 12, gamma: 0.5	39.5	39.028

Table 2: Accuracy scores of SVM classifier on **tictac_single (Intermediate boards optimal play)** dataset. All the accuracies are reported based on the mean score of 10 fold cross validation

Parameters	Validation Accuracy(%)
activation: relu, hidden_layer_sizes: (100, 100)	99.08
activation: relu, hidden_layer_sizes: (120, 120)	**99.2**
activation: relu, hidden_layer_sizes: (150, 150)	97.1
activation: relu, hidden_layer_sizes: (200, 200)	97.9

Table 3: Validation accuracy scores of MLP classifier on **tictac_final (final board classification)** dataset. All the accuracies are reported based on the mean score of 10 fold cross validation

Parameters	Validation Accuracy(%)
activation: relu, hidden_layer_sizes: (100, 100)	92.9
activation: relu, hidden_layer_sizes: (120, 120)	93.1
activation: relu, hidden_layer_sizes: (150, 150)	93.1
activation: relu, hidden_layer_sizes: (200, 200)	**93.2**

Table 4: Validation accuracy scores of MLP classifier on **tictac_single (Intermediate boards optimal play)** dataset. All the accuracies are reported based on the mean score of 10 fold cross validation

4.2 Performance on the test set

4.2.1 On the final board classification dataset

Table 5 shows the test accuracies for all the 3 classifiers on *tictac_final* (final board classification) dataset. Here, all the three algorithms perform really good, with K Nearest Neighbors getting the best of them all. The reason for KNN doing the best can be attributed to the smaller size of the dataset and the fact that the data is very well defined. Also, since no complicated classification is involved in this case, KNN gets to perform the best of them all.

Algorithms	Optimal Parameters	Test Accuracy(%)
Linear SVM	C – 10, gamma - 1	98.95
K Nearest Neighbors	Optimal K value - 3	100
Multi-Layer Perceptron	Activation function - relu Hidden Layer configuraiton: (120, 120)	99.4

Table 5: Test accuracy scores of all 3 classifiers on **tictac_final (final board classification)** dataset

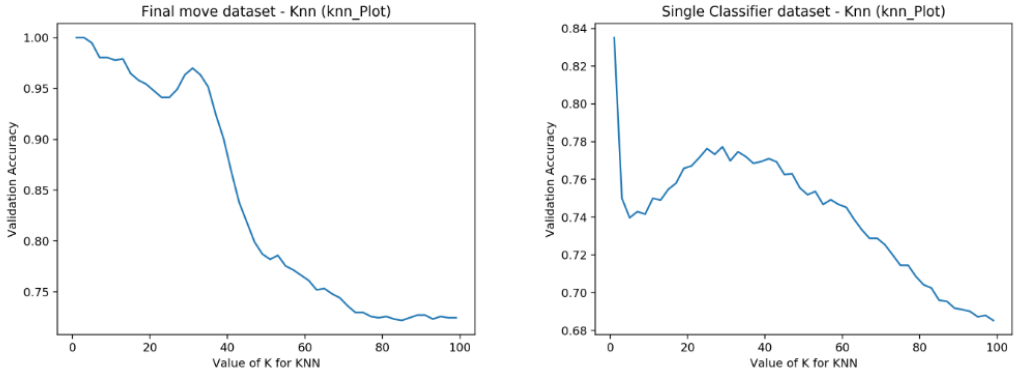

Figure 1: (Left to Right) K values vs validation accuracy plot of the knn classifiers of **tictac_final** and **tictac_single datasets**

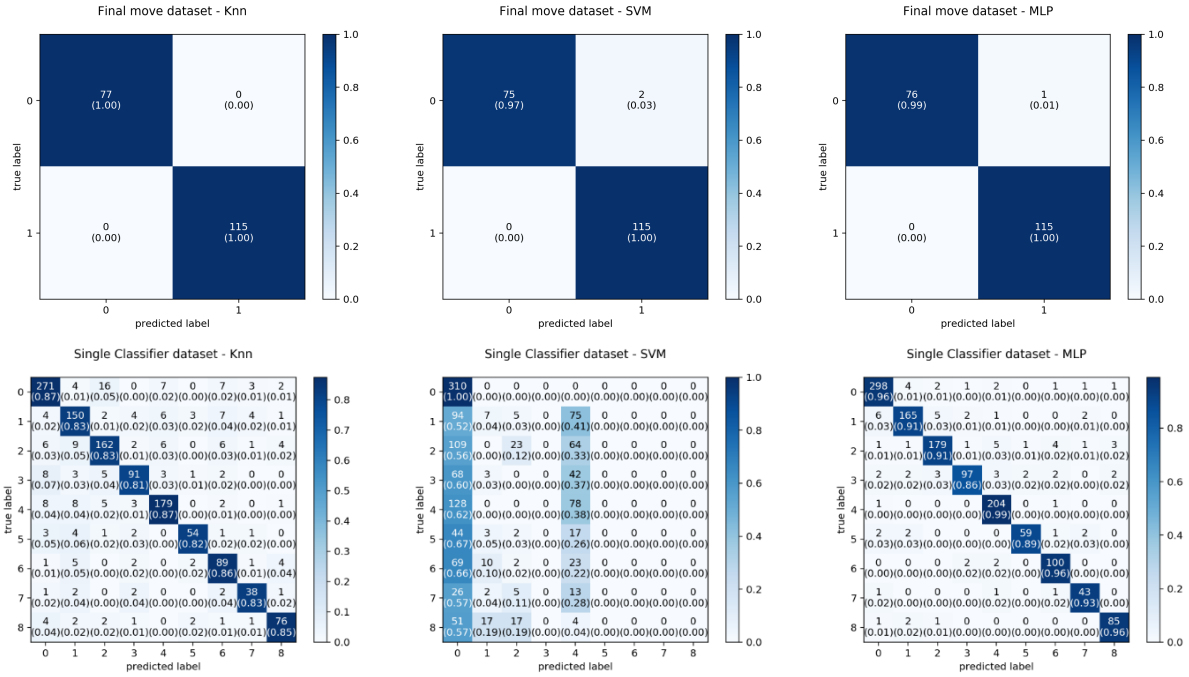

Figure 2: From top row to down, confusion Matrices of the **tictac_final** and **tictac_single** datasets

4.2.2 Intermediate boards optimal play (single label)

Table 5 shows the test accuracies for all the 3 classifiers on *tictac_single* (final board classification) dataset. In this scenario, Multi-layer Perceptron gives the best accuracy amongst all the algorithm. The dataset is substantially large and with a hidden layer configuration of (200, 200) the MLP algorithms learns the best out of all the algorithms. Linear SVM performs the worst because the data is not linearly separable. We have also tried the RBF Kernel which gives an accuracy of 93%, and this is because RBF is a non-linear (Gaussian) kernel.

Algorithms	**Optimal Parameters**	**Test Accuracy(%)**
Linear SVM	C – 10, gamma - 1	32.7
K Nearest Neighbors	Optimal K value - 3	84.6
Multi-Layer Perceptron	Activation function - relu	93.82
	Hidden Layer configuraiton: (120, 120)	

Table 6: Test accuracy scores of all 3 classifiers on **tictac_single (Intermediate boards optimal play)** dataset

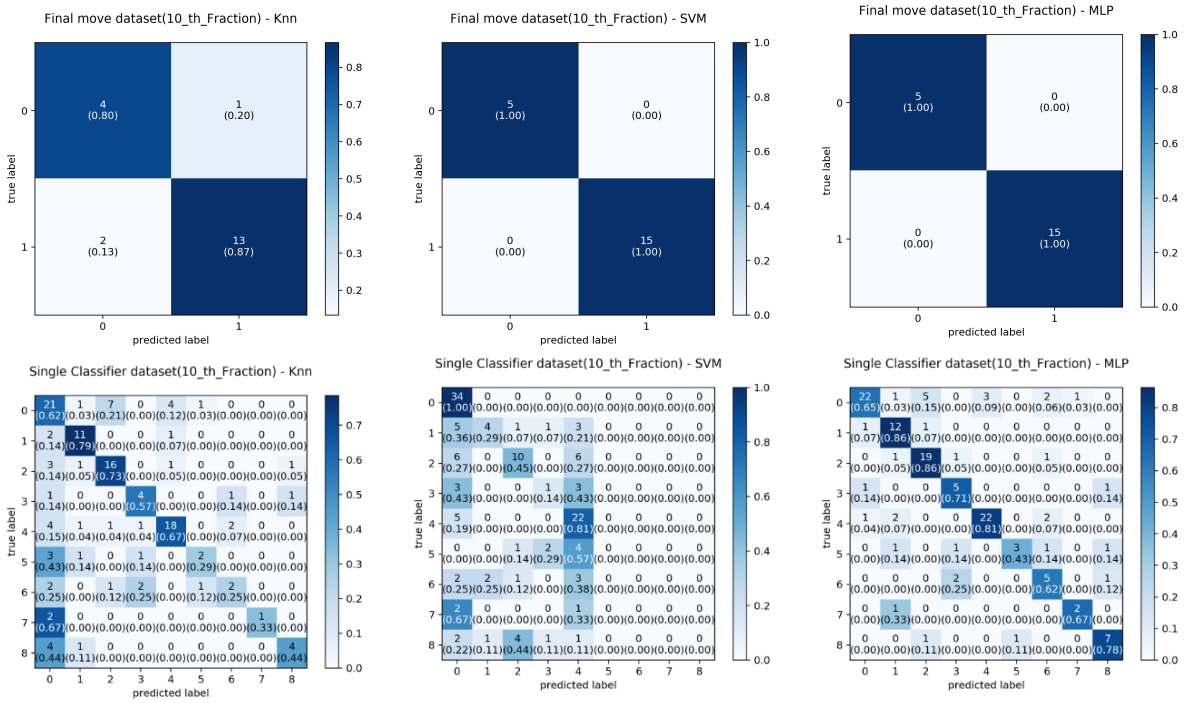

Figure 3: From top row to down, confusion Matrices of $1/10^{th}$ the **tictac_final**(1^{st} row) and **tictac_single**($2^{nd} row$) datasets for knn

4.3 Classifiers trained on 10% of the data

Table 7 and 8 show the performance of the classifiers when trained on $1/10^{th}$ of the whole dataset. In the tictac_final dataset, which by itself is small, there is no substantial change in the prediction accuracies of the models, but for the tictac_single dataset, the accuracy percentage drops significantly, and especially for MLP, due to the lack of available datapoints for the algorithm to learn from.

Algorithms	**Optimal Parameters**	**Test Accuracy(%)**
Linear SVM	C – 10, gamma - 1	85
K Nearest Neighbors	Optimal K value - 3	100
Multi-Layer Perceptron	Activation function - relu	100
	Hidden Layer configuraiton: (120, 120)	

Table 7: Test accuracy scores of all 3 classifiers on $1/10^{th}$ of **tictac_final (final board classification)** dataset

Algorithms	**Optimal Parameters**	**Test Accuracy(%)**
Linear SVM	C – 10, gamma - 1	60
K Nearest Neighbors	Optimal K value - 3	54.19
Multi-Layer Perceptron	Activation function - relu	74.04
	Hidden Layer configuraiton: (120, 120)	

Table 8: Test accuracy scores of all 3 classifiers on $1/10^{th}$ of **tictac_single (intermediate boards optimal play)** dataset

4.4 Confusion Matrix

Figure 2 and 3 show the confusion matrices for the tictac_final and tictac_single datasets trained on the entire datasets and on the $1/10^{th}$ of the datasets respectively.

5 Regressors

5.1 Cross validation

Table 9 shows the different parameters and their values used for the 10-fold cross validation of the MLP-Regressor. For knn-Regressor, we used a range for the values of k from 3 to 100 with a increment of 1 step.

Parameters	**Values**
max_iter	1000, 500
hidden_layer_sizes	(200, 100, 100, 50), (200, 100, 50)
activation	tanh, relu
solver	adam, lbfgs, sgd
alpha	0.0001
learning_rate	constant

Table 9: Grid search parameters for MLP regressors for the **tictac_multi** dataset. All the accuracies are reported based on the mean score of 10 fold cross validation

5.2 Performance on the test set

In the regression problem, the Multi-Layer Perceptron algorithm performs the best out of all the three. The reason for this can be attributed to the availability of large dataset and the depth of the network. Also, the activation function that gets selected is tanh and this fits our dataset perfectly since our data ranges from [-1, 1]. The KNN algorithms performs well, but not the best as KNN being a lazy learning model with local approximation having commonly the distance function and k value as its hyperparameters.

Linear Regression on other hand, predicts a continuous value ranging from -ve infinity to infinity. The algorithm tries to fit a straight line in this problem which demands a classification. Also, Linear Regression is highly sensitive to imbalance data and such can be seen from a comparison between the R2 and RMSE errors between the two.

Table 10 shows the performance of the regressor models on the held-out test set.

Algorithms	**Optimal Parameters**	**Test Accuracy(%)**
Linear Regression	NA	72.62
K Nearest Neighbors	Optimal K value - 3	90.64
Multi-Layer Perceptron	Activation function - tanh	97.61
	Hidden Layer configuraiton: (200, 100, 100, 50)	
	alpha: 0.0001, solver: lbfgs, max_iter: 1000, learning_rate: constant	

Table 10: Test accuracy scores of all 3 regressors on **tictac_multi** dataset

6 Optimal model used to play the game

The AI uses the MLP regressor to predict it's moves when it plays the game against the human player as the MLP regressor has the highest test accuracy than the other two regressors.

It was quite difficult to use the linear regressor model to the predict the moves as the output for a given board is a one hot encoding vector where it is one for the most optimal position and zero otherwise. So if the position predicted by the linear regressor is already occupied by the human, then there will be a problem to find out the next move. For MLP and Knn regressor, the output vector may contain multiple ones. So if the position is occupied by the human, the AI will have the choice of choosing another move.

7 Conclusion

In this project, we implemented a tictactoe game between a human player and an AI where the AI has been trained on multiple classification and regression algorithms. A comparitive study has been conducted on the accuracies and feasibility of the same on multiple datasets.

Going forward, there remains a scope to improve the game by incorporating reinforcement learning where by the AI can be trained to learn on the go from the previous moves by the human player and make decisions accordingly.