Course Project Part I Report
Simple Learning Models: Classifiers and Regressors

Pranath Reddy Kumbam
University of Florida, Gainesville, FL 32611, USA

UFID: 8512-0977

In this project, we will be exploring the application of machine learning to develop an AI-based player for a two-player game environment. Specifically, we will be developing and evaluating the performance of an ML model for a game of Connect Four. We begin by testing the performance of various classification and regression models in the context of Tic-Tac-Toe. While a computer player for a game of Tic-Tac-Toe can be developed based on the minimax algorithm, testing machine learning models for this task could be an interesting exercise and also provide a baseline for the performance and contextualize the results of all the tested models, which can help us select a model to test on the Connect Four game. This will help us understand how and whether simple machine learning models can be used to develop an AI-based player for zero-sum games.

I. INTRODUCTION

Zero-sum games are increasingly used to develop and benchmark intelligent algorithms. A popular example of this is DeepMind's AlphaGo which is the first computer program to defeat a professional human Go player. While Go is an incredibly complex game with more possible positions than there are atoms in the universe and the AlphaGo algorithm uses a specific machine learning technique called Reinforcement learning, we will study a simpler game that is Connect Four and explore various simple supervised machine learning models to investigate the possibility of developing a competent AI-based computer player. Firstly, we will test our models on the Tic-Tac-Toe data which will help us benchmark and establish a baseline.

Tic-Tac-Toe and Connect Four are zero-sum games that can be solved using the minimax algorithm. We will be using synthetically generated data to train our models. Two players take turns placing the markings X and O in one of the nine spots on a three-by-three grid as they play tic-tac-toe. Tic-tac-toe is frequently used in the area of artificial intelligence that deals with the searching of game trees because of its simplicity. It is simple to train a computer to play tic-tac-toe flawlessly or to enumerate all 26,830 potential games in this space up to rotations and reflections (game tree difficulty) or the 765 fundamentally distinct locations (state space complexity). There are no secrets in the two-player game of Connect Four because both players have complete information. Given that a player's advantage is a disadvantage to an opponent, Connect Four can also be categorized as an adversarial, zero-sum game. The number of game board positions is one way to gauge how complicated a game of Connect Four is. There are 4,531,985,219,092 positions for all game boards filled with 0 to 42 pieces in conventional Connect Four played on a 7-column by 6-row grid.

II. METHODOLOGY AND RESULTS

We will be training several supervised classification and regression models to solve the task of binary and multi-class classification. We have written Python scripts for training and testing classification and regression models on the Tic-Tac-Toe data for solving the tasks of binary classification, multi-class classification, and multi-label classification. For Connect Four, we have chosen to go with classifiers considering that they achieved the highest performance and performed better than regression models on Tic-Tac-Toe data. Furthermore, we have also developed a command line game for testing the trained model in a real-world practical use-case. For Tic-Tac-Toe, we have trained and tested three models: k nearest neighbours, SVM, and a multi-layer perceptron. First, we have split the dataset into training and validation datasets first, and the training data will be further split while performing 10-fold and K-fold cross validation. The validation data, which we have split above, will be used to perform grid search and find the best model hyper parameters. For MLP in particular, we have tested one to five hidden layers and found that three hidden layers work best. For the number of neurons (width) of the layers, the configuration of (200, 400, 200) performed the best, and the scikit-learn implementation of

the MLP classifier uses cross entropy as the loss function. For SVM, we tested both the linear and RBF kernels. We have also tested regression models on a multi-class multi-label dataset of Tic Tac Toe . The linear regression implementation is based on the normal equation.

$$W = ((X \times X^{\intercal})^{-1} \times X^{\intercal}) \times Y, \tag{1}$$

We have recorded both the multi-label accuracy and also the score based on the number of optimal moves the model predicts. The parameters for the regression MLP and KNN models are pretty similar to their classification counterparts, but here we have used four hidden layers for MLP considering the increased complexity of the task. The metrics used for evaluating the models are accuracy and normalized confusion matrices. The accuracy scores for the Tic Tac Toe datasets are compiled in the following tables.

TABLE I: Binary Classification Accuracy Results for Tic-Tac-Toe data (tictac final)

Model	Mean	Standard Deviation
MLP	0.9945	0.00737
KNN	1	0
SVM (Linear)	0.98241	0.01407
SVM (RBF)	0.98681	0.00957

TABLE II: Multi-Class Classification Accuracy Results for Tic-Tac-Toe data (tictac single)

Model	Mean	Standard Deviation
MLP	0.94552	0.01839
KNN	0.8711	0.00936
SVM (Linear)	0.23734	0.0153
SVM (RBF)	0.82195	0.02041

TABLE III: Regression Accuracy Results for Tic-Tac-Toe data (tictac multi)

Model	Mean	Standard Deviation
MLP	0.91644	0.0124
KNN	0.80026	0.02191
Linear Regression	0.09529	0.0125
Linear Regression (Optimal Movies)	0.23445	0.01793

All models perform equally well for the binary classification task, with a score close to unity, but we see a differentiation when it comes to multi-class performance. MLP performs the best, which showcases the ability of neural networks to learn and model non-linear and complex relationships. On the opposite end of the spectrum, we see the SVM with a linear kernel performing very poorly, which suggests that the data is not linearly separable. We see a similar trend in the regression results while the linear regression model is struggling to produce good results which again highlights the non-linearity of the data.

Additionally, we also test the models on 1/10th of the data to see how the models scale down. As expected, the models perform worse on the smaller dataset. In the case of binary classifcation, the arguably simplest model among all KNN performs the best and the other two models scale down accordingly, which shows the relation between model complexity and generalization. In the case of multi-class, KNN displays the worst performance, with the other two models scaling proportionally, which could be due to the model failing to converge. The results of training on 1/10th of the data are shown below.

TABLE IV: Binary Classification Accuracy Results for 1/10th Tic-Tac-Toe data (tictac final)

Model	Mean	Standard Deviation
MLP	0.78999	0.10592
KNN	0.83444	0.13429
SVM (RBF)	0.70111	0.15888

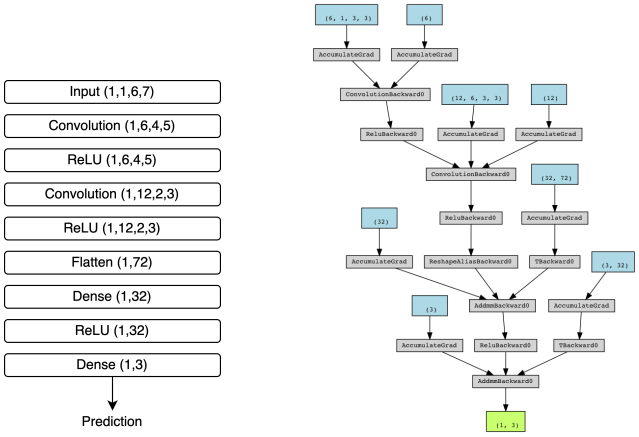

FIG. 1: Architecture of the CNN Model (a) and the block diagram of the graph visualized using torchviz(b)

TABLE V: Multi-Class Classification Accuracy Results for 1/10th Tic-Tac-Toe data (tictac single)

Model	Mean	Standard Deviation
MLP	0.73658	0.05076
KNN	0.56653	0.04064
SVM (RBF)	0.66436	0.04208

We tested classifiers, specifically MLP, decision trees, random forests, and a convolutional neural network (CNN), for Connect Four. We have used a similar approach for training as the one used for Tic Tac Toe. Since the Connect Four dataset is highly unbalanced, we have tested the models on the unbalanced data and also the balanced data where each class has the same number of samples. In this case, the "draw" class. All the models are trained on our local computer using scikit-learn except CNN, which has been implemented using Pytorch and trained on Google Colab. The model is shown in Figure 1 . We used the Adam optimizer and Cross entropy loss and trained the model for 500 epochs. The results for Connect Four have been presented in the following tables.

TABLE VI: Multi-Class Classification Accuracy Results for Balanced Connect Four Data

Model	Mean	Standard Deviation
CNN	0.86479	0.01439
MLP	0.88824	0.00926
Random Forest	0.87175	0.00597
Decision Trees	0.83845	0.00897
KNN	0.74775	0.01027
SVM	0.72707	0.00825

TABLE VII: Multi-Class Classification Accuracy Results for Unbalanced Connect Four Data

Model	Mean	Standard Deviation
CNN	0.82731	0.00684
MLP	0.85733	0.00458
Random Forest	0.8341	0.00392
Decision Trees	0.77712	0.00487
KNN	0.69533	0.00648
SVM	0.75348	0.00527

We have also implemented a command line python connect four game using the trained ML models. The ML-based model is relatively easy to defeat. It's not entirely surprising considering that the accuracy of the models is not very high. Another observation is that the model performs slightly better as the board gets populated, but at the beginning of the game it's fairly trivial to defeat. For example, if we

just keep selecting a single column, the game gets completed in just four moves. This could mean that the model is weak against sparsity. CNNs are generally good at analyzing sparse data, so it could mean that the training data doesn't have enough sparse samples for the model to learn and generalize. It's something worth studying further. And then we have also tested the model against a naive random generator, which randomly picks a column, and the CNN model performs respectably well against it, with an average win rate of around 75 percent, which shows that the model is not completely ineffective. We have run 50 experiments per model, and each experiment had 20 game runs. We compiled the scores (average win rates) in the following tables.

TABLE VIII: Connect Four vs Random Generator results for Unbalanced Connect Four Data

Model	Mean	Standard Deviation
CNN	75.2	6.8818
MLP	62.5	7.0391
Random Forest	70.3	5.8403

TABLE IX: Connect Four vs Random Generator results for Balanced Connect Four Data

Model	Mean	Standard Deviation
CNN	64.1	6.9419
MLP	66	7.0142
Random Forest	61.8	8.1461

Although the accuracy value of the MLP model is higher than the CNN model, in practical use, we can see that the CNN model outperforms the MLP model when tested against a naive player. This can be explained by the fact that convolutional neural networks have translational invariance as opposed to MLP models. Models trained on unbalanced datasets seem to be performing better than those trained on balanced datasets, which could be due to the difference in the size of training samples . While the results are definitely decent, we can see that simple ML models are not suitable to create AI-based players to challenge humans in a relatively complex zero-sum game like Connect Four. Perhaps reinforcement learning based models are better suited for this task. Figure 2 shows a simple block diagram of the game play implementation.

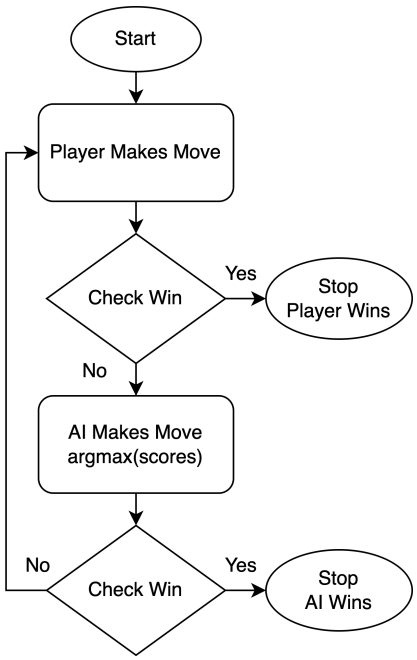

FIG. 2: A simple block diagram of the connect four gameplay

III. SUBMISSION DETAILS

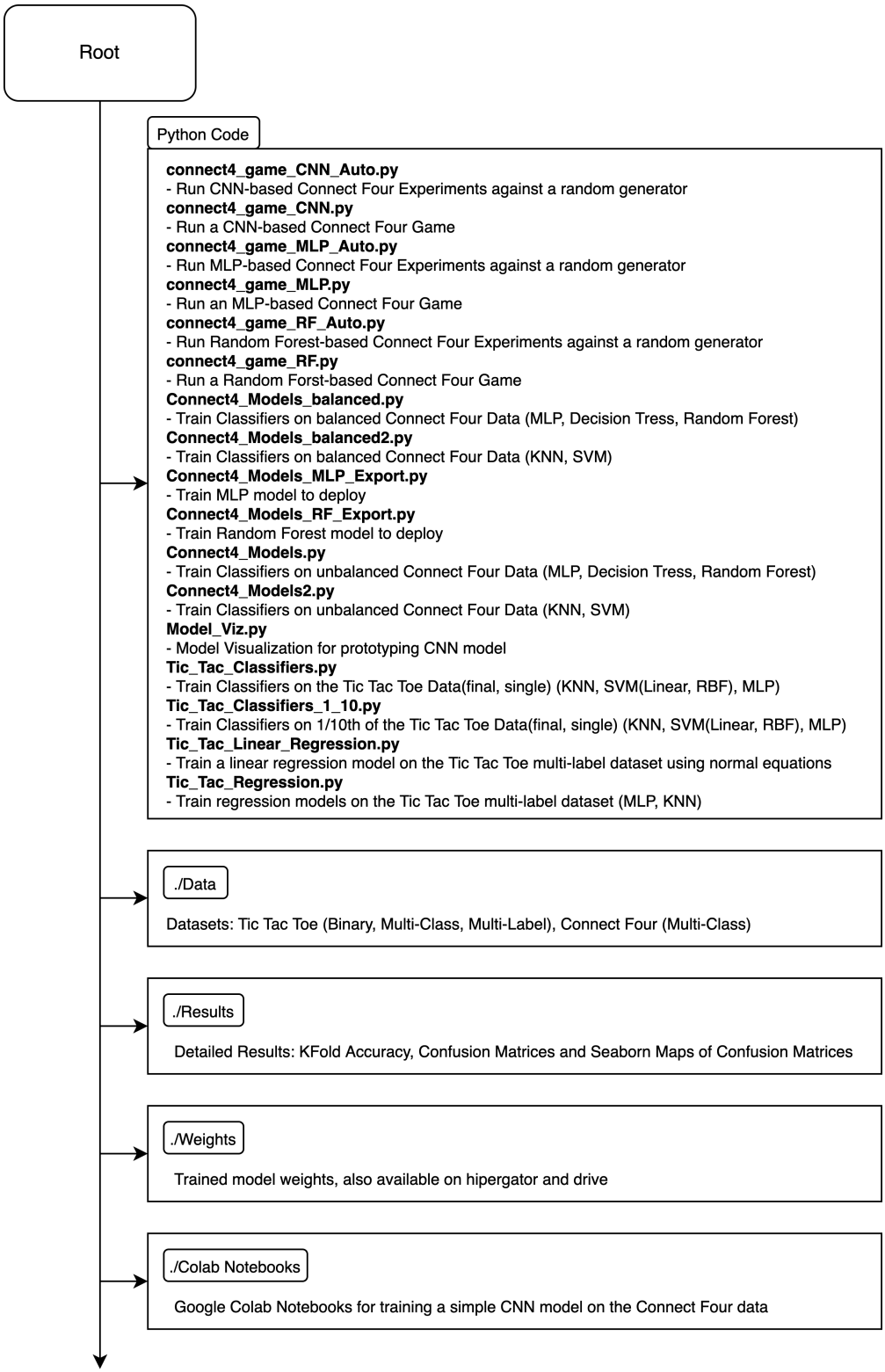

FIG. 3: File structure of the submission

Figure 3 shows the file structure of the submission and the details of each Python script. The scripts are divided based on the types of tasks (binary classification, multi-class classification, and multi-label classification) and models. As shown in the file structure, the model weights and data are stored in separate folders . The results are compiled into text files which include the accuracy scores and normalized confusion matrices. Additionally, we have also plotted the seaborn heatmaps for Tic-Tac-Toe results. The zip file of the folder containing the project folder, file structure and the project walkthrough video can be found in the following Google drive link:

Google Drive Submission

Additionally, Figure 4 shows the details of the files in the results folder.

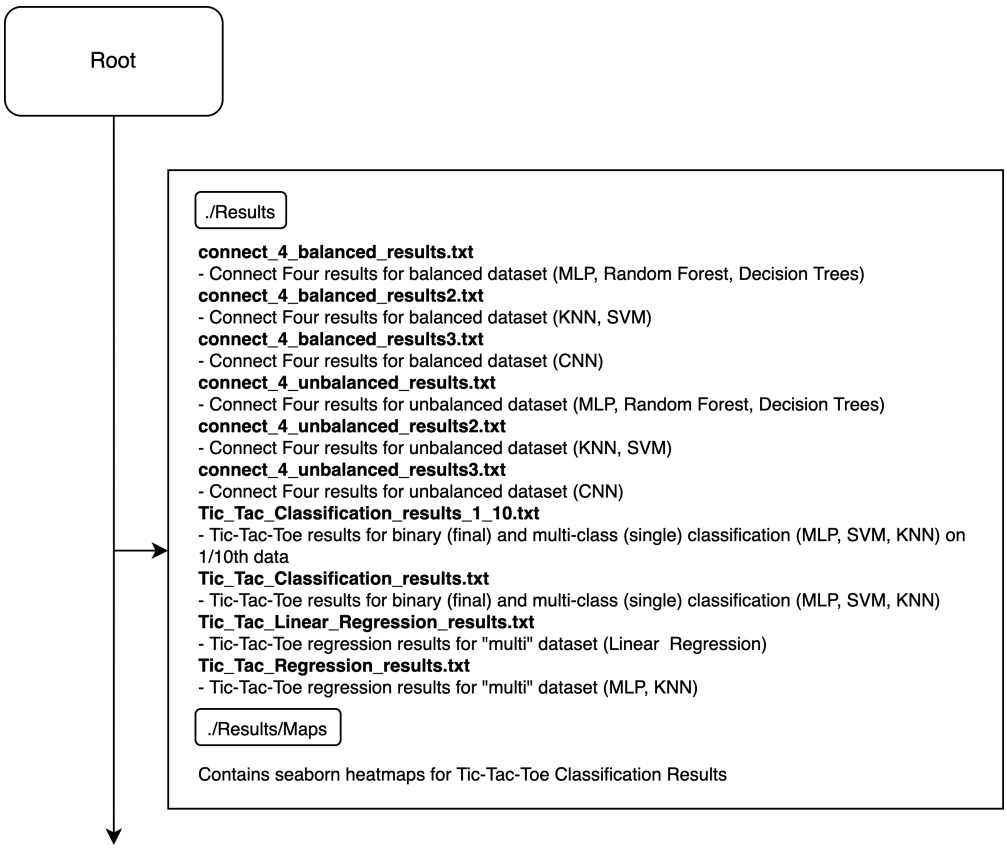

FIG. 4: File structure and details of the Results