Assignment 1 Final Report

Pieter Alley, Pranet Allu, Oliver Peralta

Tic-Tac-Toe

Implementation:

The implementation begins with initializing a blank game board and loading training data from both single and multi-label datasets. The classifiers and regressors are then trained on their respective datasets, enabling them to learn patterns and strategies from the historical game data. Upon starting the game, the user is prompted to select a model to play against. During gameplay, the user inputs their moves. Due to the models being trained on extensive data, the user is unlikely to win.

Classifier Performance:

- Linear SVM:
  - Accuracy: 32.189
  - Cross-Validated Accuracy: 35.019
  - Game Test: 9 ties, 1 loss against the model
  - Confusion Matrix:

```
[[1.         0.         0.         0.         0.         0.
 0.         0.         0.        ]
[0.45454545 0.03030303 0.06060606 0.         0.45454545 0.
 0.         0.         0.        ]
[0.57352941 0.         0.10784314 0.         0.31862745 0.
 0.         0.         0.        ]
[0.50980392 0.02941176 0.05882353 0.05882353 0.34313725 0.
 0.         0.         0.        ]
[0.63414634 0.         0.         0.         0.36585366 0.
 0.         0.         0.        ]
[0.60869565 0.         0.08695652 0.         0.30434783 0.
 0.         0.         0.        ]
[0.61538462 0.05982906 0.01709402 0.05982906 0.24786325 0.
 0.         0.         0.        ]
[0.7037037  0.03703704 0.03703704 0.         0.22222222 0.
 0.         0.         0.        ]
[0.55555556 0.08641975 0.24691358 0.         0.11111111 0.
 0.         0.         0.        ]]
```

- K-Nearest Neighbors
  - Accuracy: 77.956
  - Cross-Validated Accuracy: 73.588
  - Game Test: 10 ties
  - Confusion Matrix:

```
Confusion Matrix
 [[0.87579618 0.02547771 0.03821656 0.00955414 0.01592357 0.00318471
  0.00955414 0.00318471 0.01910828]
 [0.04242424 0.82424242 0.03030303 0.01212121 0.03636364 0.01212121
  0.01818182 0.01212121 0.01212121]
 [0.11764706 0.06372549 0.76960784 0.01960784 0.01960784 0.00980392
  0.         0.         0.        ]
 [0.11764706 0.04901961 0.03921569 0.7254902  0.02941176 0.
  0.02941176 0.         0.00980392]
 [0.10243902 0.03414634 0.02439024 0.01463415 0.81463415 0.
  0.00487805 0.         0.00487805]
 [0.05797101 0.05797101 0.07246377 0.02898551 0.01449275 0.68115942
  0.         0.         0.08695652]
 [0.11965812 0.05128205 0.08547009 0.04273504 0.03418803 0.
  0.65811966 0.00854701 0.        ]
 [0.09259259 0.05555556 0.01851852 0.09259259 0.01851852 0.03703704
  0.         0.66666667 0.01851852]
 [0.12345679 0.07407407 0.04938272 0.01234568 0.0617284  0.01234568
  0.         0.01234568 0.65432099]]
```

- Multilayer Perceptron
    - Accuracy: 95.423
    - Cross-Validated Accuracy: 91.756
    - Game Test: 10 tie
    - Confusion Matrix:

```
[[0.95541401 0.00318471 0.01273885 0.         0.         0.01273885
 0.01273885 0.00318471 0.        ]
[0.00606061 0.92727273 0.03030303 0.01818182 0.01818182 0.
 0.         0.         0.        ]
[0.         0.00490196 0.98039216 0.00980392 0.         0.00490196
 0.         0.         0.        ]
[0.00980392 0.00980392 0.         0.95098039 0.00980392 0.
 0.         0.         0.01960784]
[0.01463415 0.         0.00487805 0.00487805 0.96585366 0.
 0.         0.         0.0097561 ]
[0.         0.         0.         0.01449275 0.01449275 0.95652174
 0.         0.         0.01449275]
[0.03418803 0.         0.         0.         0.         0.00854701
 0.95726496 0.         0.        ]
[0.         0.         0.         0.01851852 0.         0.
 0.01851852 0.96296296 0.        ]
[0.03703704 0.02469136 0.         0.         0.         0.03703704
 0.         0.         0.90123457]]
```

Regressor Performance:

- KNN
    - Accuracy: 72.998
    - R2 Score: 0.561
    - Mean Absolute Error: 0.131
    - Mean Square Error: 0.071
    - Game Test: 10 ties
    - Confusion Matrix:

```
Confusion Matrix
 [[212  17  24  11  23   6  14   4   9]
 [  6 113  21  11  16   4   7   6   5]
 [  6   3 132   6  10   0   7   5   8]
 [  2   1   0  61  12   2   5   5   2]
 [  2   7   4   5 186   3   2   6   4]
 [  1   2   1   1   1  50   2   2   2]
 [  3   1   5   6   2   2  83   2   1]
 [  0   0   0   0   0   1   3  32   5]
 [  8   3   0   0   4   2   2   1  88]]
```

- Linear Regression
    - Accuracy: 18.459
    - R2 Score: 0.0016
    - Mean Absolute Error: 0.335
    - Mean Square Error: 0.166
    - Game Test: 10 wins against the model
    - Confusion Matrix:

```
Confusion Matrix
 [[ 20   0  37   2 209   0  36   0   0]
 [ 21   0  13   0 121   0  19   0   3]
 [ 16   0  21   0 114   0   4   0   4]
 [ 14   0   9   0  67   0  25   0   0]
 [ 24   0   6   0 180   0   5   0   3]
 [ 11   0   9   0  38   0   5   0   0]
 [ 13   0   8   0  78   0  21   0   0]
 [ 10   0   8   0  28   0  11   0   0]
 [  7   0  13   0  66   0  12   0   0]]
```

- Multilayer Perceptron
    - Accuracy: 69.794
    - R2 Score: 0.628
    - Mean Absolute Error: 0.175
    - Mean Square Error: 0.060
    - Game Test: 10 ties against model
    - Confusion Matrix:

```
Confusion Matrix
 [[182   3  20   7  34   9  32  14  17]
 [  0  75  14  11  30   7  19   2  23]
 [  2   0 135   3  19   3   8   1  15]
 [  0   0   0  50  17   5   7   4   6]
 [  0   0   0   0 180   5   3   5  12]
 [  0   0   0   0   0  60   5   4   2]
 [  0   0   0   3   7   0  86   1   2]
 [  0   0   0   0   2   0   1  47   7]
 [  1   1   0   0   2   1   0   0 100]]
```

How to run code

- Unzip our given file into a directory of your choice.

To play game

- Run “python TicTacToeGame.py” in the terminal.
- Follow directions, you will be asked to enter a number 1-6 to pick the ML model to play against and then you will be asked for a row and column index.

To run ML code

- Depending on how you will run the code, either open Jupyter Notebook or run “jupyter notebook Evaluation.ipynb”.
- In Jupyter notebook, simply run all cells and look at evaluation metrics and code that was used to make the implementation.

Bugs (if any)

- Out of bounds error on entering a number not between or including 0 and 2 when playing tic tac toe game.