

# Project Report

Ethan Prihar, Odell Dotson

## Description of the Features and Design Strategies

The first feature returns the player whose piece is in the bottom left corner of the board. This feature is not very useful because having a piece in the bottom left corner of the board is not something that should determine if a player wins. This feature was implemented because it was required by the project guidelines.

The second feature returns which player has more pieces that aren't at the edge of the board. This feature should be useful because when a piece is at the edge of the board it has a lot less potential to be connected to. In general a piece toward the center would have a higher chance of being used to win than a piece on the side. This feature predicts that the player with the least pieces at the edge will win.

The third feature calculates a heuristic value based on how close all of a players pieces are to the center. This feature is similar to feature two because it also values pieces that are away from the edge of the board. This feature goes beyond feature two because each piece closer to the center has more weight in the heuristic calculation. The heuristic is calculated by adding one for every piece in an edge column. For every column closer to the center, a piece in that column is worth twice as much as a piece in the adjacent column closer to the edge of the board. Therefore, the number of pieces in a column is multiplied by 1, 2, 4, 8, 4, 2, or 1 respectively. The total value of all of the columns is evaluated and the player with the highest total is returned by the feature.

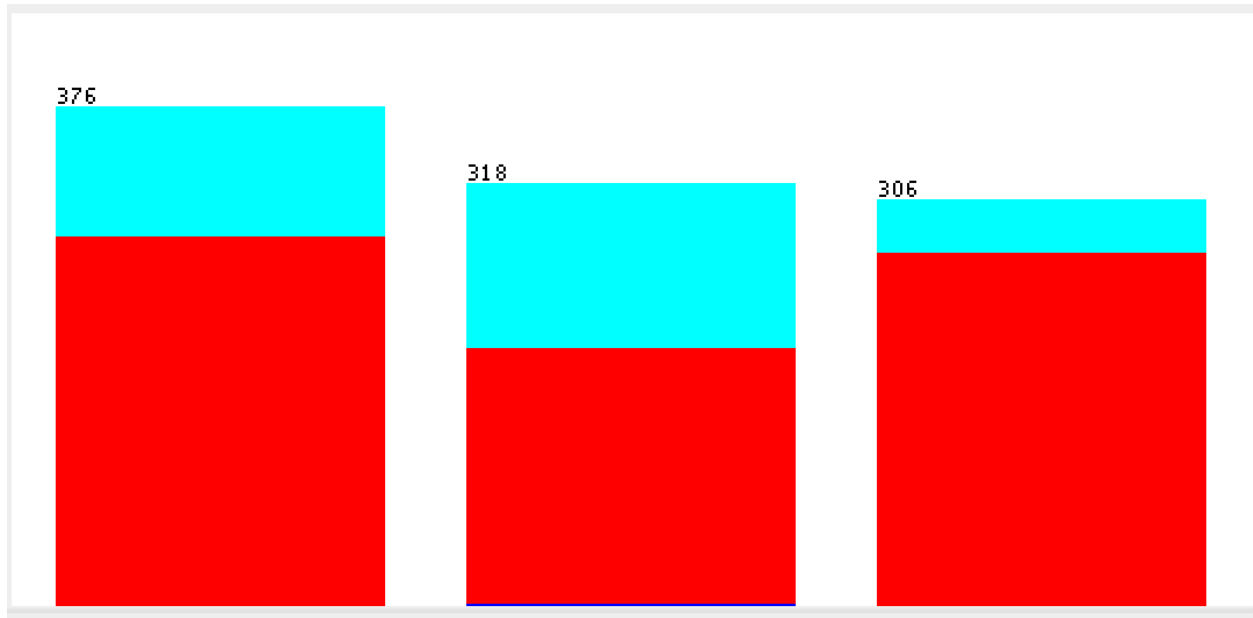
The fourth feature uses the same heuristic as feature three, but instead of returning whichever player has a higher heuristic, the feature returns the value of player one's heuristic minus player two's heuristic. This feature therefore returns a continuous function that represents how much of an advantage one player has over another because of a more centered positioning. This feature's usefulness will be compared to the previous feature's usefulness to determine whether it is better for this data set to use a continuous value to represent positioning advantage, or a categorized set of values.

The fifth feature tests who has "top control" of the columns of the board. "Top control" is defined as whoever has the highest piece on any given column. For every column where player 1 has top control, 1 is added to the top control variable and for every column that player 2 has top control, 1 is subtracted from this variable. Columns with no pieces in them do not count for top control at all and do not affect the top control score. When doing the first project, we found that often that your ability to make advantageous moves was dependent on how much control you had over the top edge of the board because you're only able to make plays that will become new edges on the top, interacting with the old top of the board.

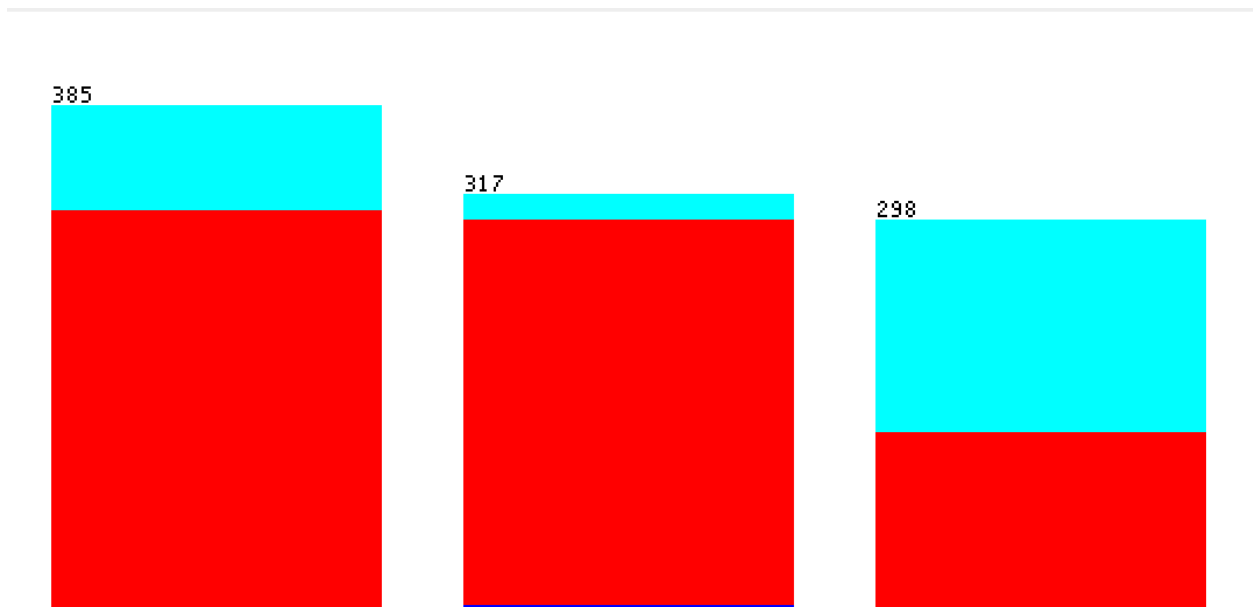
## Analysis of the Features

To determine which features are better than others we can look at how well each feature predicts the winner of a board. To do this we show a histogram of every value the

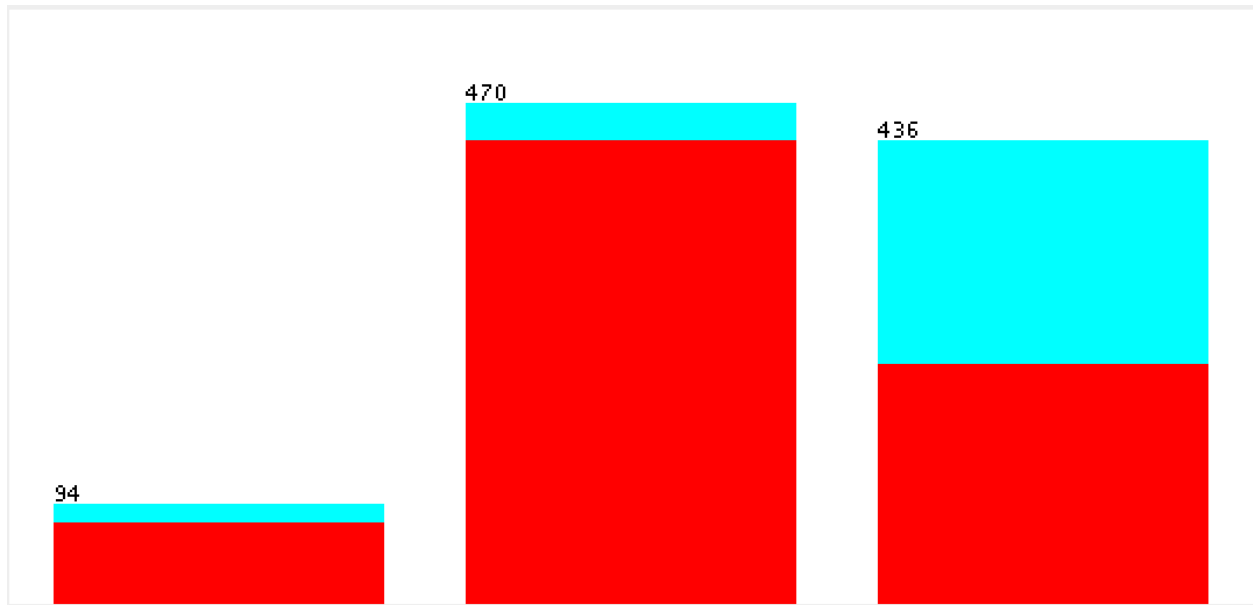
feature returns with how many times player 1 or player 2 won a game at each value of the feature. Player 1 won about 75% of its games, so if the histogram shows that at each value of the feature, about 75% of the time player 1 won, then we know that feature is useless. Speaking of useless features, below is a graph of feature 1.



As you can see from the data, every value of feature one leads to only a slight change in the win rate of player one, ranging from around 65% to 85% instead of 75%. There is only around a 10% change in the win rate of player 1 based on the value of feature 1 so this heuristic is not very good at predicting the winner. Feature 2 on the other hand has a larger change in the win rate of player one based on its value. The graph of Feature 2 is shown below.

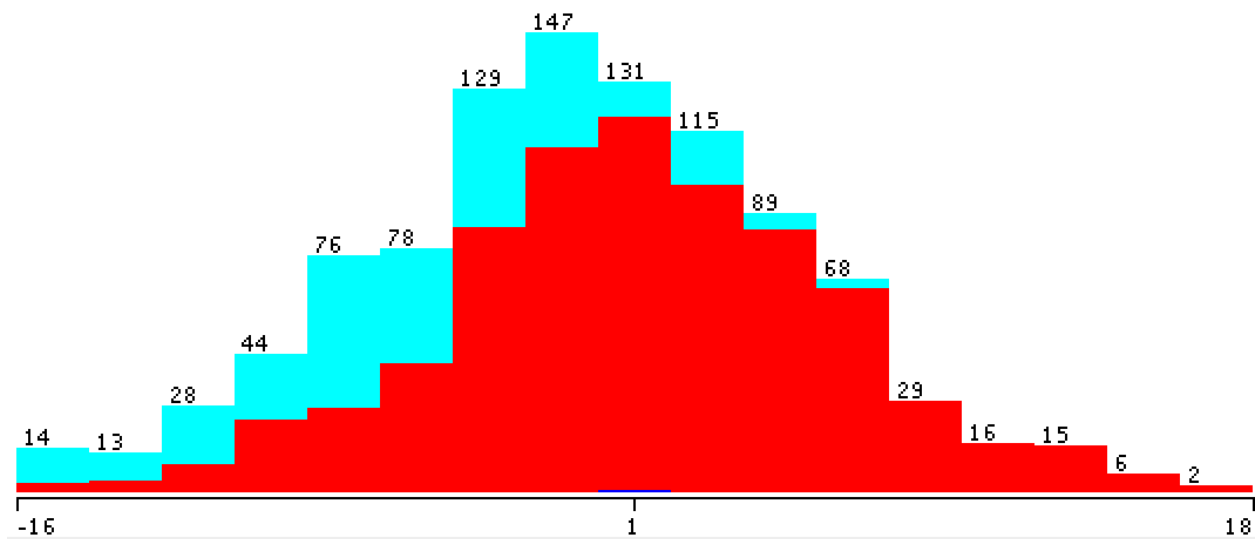


As shown by the graph, when feature 2 puts a data point into one category, player 1 wins 95% of the time, and when feature 2 puts a data point in a different category, player 1 wins 45% of the time. Both of these changes in the win rate of player 1 show that feature 2 can predict the winner much more accurately than feature 1. The issue with feature 2 is that it has one category that it cannot predict the outcome from accurately and it places over a third of the data into this category. Feature 2 is good, but only for two thirds of the data. Feature 3 on the other hand can almost as accurately predict the outcome of many more data points than feature 2 which is shown by the graph of feature 3 below.

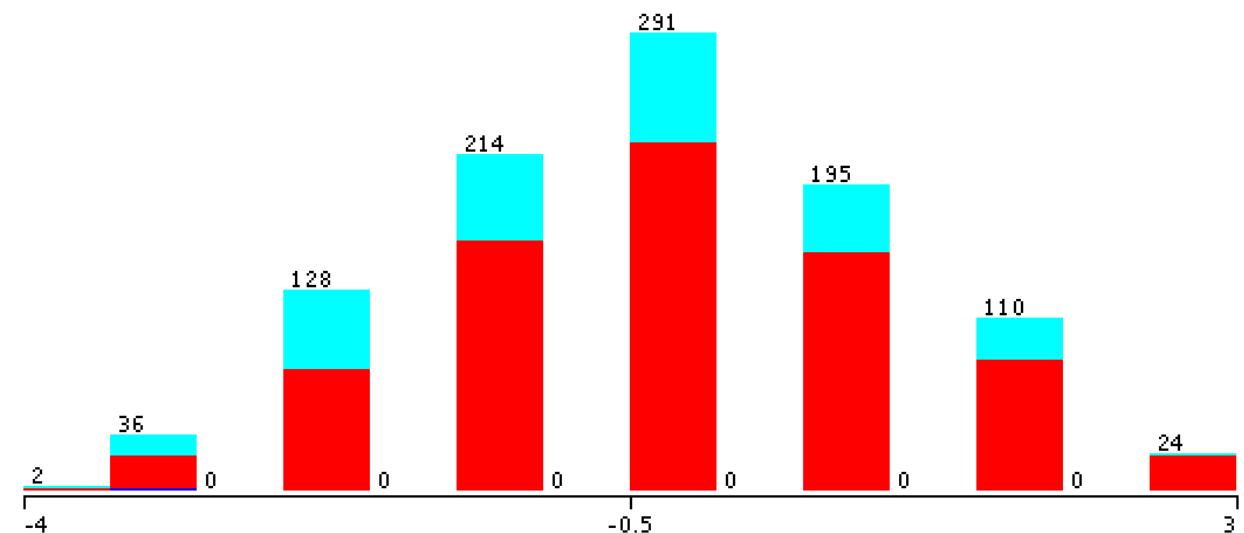


For each category in feature 3, there is almost a 25% change in the win rate of player 1 than if the predictions were random. Feature 3 can predict all of the data with this accuracy, whereas feature 2 could only predict two thirds of the data with this much accuracy.

Feature 4 creates a continuous output of values. The decision tree treats each value as a different category. Below is the graph of feature 4.



As shown by the graph, feature 4 has the potential to predict values more accurately than any of the previous graphs. For some values feature 4 can 100% accurately predict that player 1 will win, and that player 1 will win 15% of the time. There is only 30% of the data that feature 1 cannot predict accurately but it can more accurately predict the other 70% of the data than any of the other features. Feature 5 was not very good at predicting the winner and for every value of feature 5, there was not much change in the win rate of player 1 than if the winner was predicted randomly.



Based on the analysis of every feature, feature 4 is by far the most useful feature because it can predict whether or not player 1 will win with more accuracy than any of the other features. The only place feature 4 falls short is it's inability to accurately predict 30% of the data. This is where feature 3 is useful because it can predict the winner of all of the data almost as accurately as feature 4 can predict the winner of 70% of the data. So the 30% of

data that feature 4 cannot predict well can be predicted by feature 3, this will ensure all of the data is predicted with good accuracy.

## The Value of Cross Validation:

Cross-validation is used to prevent overfitting. Holdout percents will train a system on a portion of the data, and then test that trained model against the holdout percent, testing the validity of the model. This helps prevent overfitting because it is observable how badly off the prediction is on data that it has never seen before. This prevents it from becoming overfit on that data.

K-fold cross validation is essentially holdout cross-validation that runs multiple times, on the data split up in different ways. This prevents overfitting because across a full k-folded run, all the data will be run upon multiple times but each datapoint will also have been left out at some point in the testing. This helps to prevent overfitting because the data that is left out is not overfit for.

## Decision Tree Testing

To create the optimal decision tree we used two methods of cross validation. First we held out a certain percent of the data and created a tree with the remaining data, then tested on the held out data. We started with a hold out of 10% and increased this percent until the accuracy of the tree began to decrease. Below is the screenshot of the 10% holdout test.

The screenshot shows the Weka Classifier window. The 'Classifier' dropdown is set to 'J48 -C 0.25 -M 2'. Under 'Test options', 'Percentage split' is selected with a value of 10%. The '(Nom) winner' dropdown is set to '(Nom)'. The 'Start' button is visible. The 'Result list (right-click for options)' shows a list of test runs, with the most recent being '14:51:53 - trees.J48'. The 'Classifier output' pane displays the following summary and detailed accuracy by class:

```
==== Summary ====
Correctly Classified Instances      666           74 %
Incorrectly Classified Instances    234           26 %
Kappa statistic                    0.3545
Mean absolute error                 0.1792
Root mean squared error             0.3701
Relative absolute error             64.5625 %
Root relative squared error         102.5278 %
Total Number of Instances          900

==== Detailed Accuracy By Class ====
          TP Rate   FP Rate   Precision   Recall   F-Measure   ROC Area   Class
          0         0         0           0         0           0.5        0
          0.799     0.425     0.843       0.799     0.821       0.723      1
          0.573     0.202     0.496       0.573     0.532       0.721      2
Weighted Avg.   0.74     0.367     0.753       0.74     0.745       0.722

==== Confusion Matrix ====
a  b  c  <-- classified as
0  0  1  |  a = 0
0 533 134 |  b = 1
0  99 133 |  c = 2
```

As shown, holding out 10% of the data lead to 74% accuracy. When increasing to 20% hold out the accuracy increased. Below is the 20% holdout test.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 22

☒ Percentage split % 20

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:55:15 - trees.J48

13:55:21 - trees.J48

13:55:27 - trees.J48

13:55:30 - trees.J48

13:55:33 - trees.J48

13:55:36 - trees.J48

13:59:45 - functions.MultilayerPerceptron

14:49:09 - trees.J48

14:51:31 - trees.J48

14:51:53 - trees.J48

14:52:26 - trees.J48

14:52:26 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	594	74.25 %
Incorrectly Classified Instances	206	25.75 %
Kappa statistic	0.3086	
Mean absolute error	0.1844	
Root mean squared error	0.3779	
Relative absolute error	71.2602 %	
Root relative squared error	104.4295 %	
Total Number of Instances	800	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.5	0
	0.853	0.561	0.806	0.853	0.829	0.661	1
	0.441	0.147	0.522	0.441	0.478	0.664	2
Weighted Avg.	0.743	0.45	0.73	0.743	0.735	0.662	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	500	86	b = 1
0	119	94	c = 2

The accuracy of the prediction with 20% hold out had increased from 10% by 0.25%. The accuracy of the prediction continued to increase as the hold out percent was increased to 30%. The screenshot of the 30% hold out test is shown below.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 22

☒ Percentage split % 30

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:55:21 - trees.J48

13:55:27 - trees.J48

13:55:30 - trees.J48

13:55:33 - trees.J48

13:55:36 - trees.J48

13:59:45 - functions.MultilayerPerceptron

14:49:09 - trees.J48

14:51:31 - trees.J48

14:51:53 - trees.J48

14:52:26 - trees.J48

14:53:15 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	540	77.1429 %
Incorrectly Classified Instances	160	22.8571 %
Kappa statistic	0.3659	
Mean absolute error	0.1822	
Root mean squared error	0.3625	
Relative absolute error	71.2526 %	
Root relative squared error	98.4511 %	
Total Number of Instances	700	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.5	0
	0.907	0.579	0.802	0.907	0.851	0.718	1
	0.423	0.093	0.636	0.423	0.508	0.72	2
Weighted Avg.	0.771	0.444	0.755	0.771	0.755	0.718	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	458	47	b = 1
0	112	82	c = 2

When increasing the hold out percentage to 30% the accuracy jumped to 77.1%. Again the hold out percentage was increased by 10% to see if the accuracy would continue to increase. Shown below is the screenshot of the 40% hold out test.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 22

☒ Percentage split % 40

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

- 13:55:27 - trees.J48
- 13:55:30 - trees.J48
- 13:55:33 - trees.J48
- 13:55:36 - trees.J48
- 13:59:45 - functions.MultilayerPerceptron
- 14:49:09 - trees.J48
- 14:51:31 - trees.J48
- 14:51:53 - trees.J48
- 14:52:26 - trees.J48
- 14:53:15 - trees.J48
- 14:53:58 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	455	75.8333 %
Incorrectly Classified Instances	145	24.1667 %
Kappa statistic	0.325	
Mean absolute error	0.1982	
Root mean squared error	0.3525	
Relative absolute error	76.7075 %	
Root relative squared error	95.2232 %	
Total Number of Instances	600	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.5	0
	0.909	0.624	0.787	0.909	0.844	0.73	1
	0.379	0.09	0.621	0.379	0.471	0.733	2
Weighted Avg.	0.758	0.472	0.739	0.758	0.737	0.73	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	391	39	b = 1
0	105	64	c = 2

As shown by the data, the accuracy of the decision tree at 40% hold out decreases from the accuracy at 30%. This means that the optimal amount to hold out is somewhere between 30% and 40%. The highest accuracy was achieved at 32% holdout with an accuracy of 77.4%. Below is the screenshot of the 32% hold out test.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☐ Cross-validation Folds 22

☒ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

- 13:55:30 - trees.J48
- 13:55:33 - trees.J48
- 13:55:36 - trees.J48
- 13:59:45 - functions.MultilayerPerceptron
- 14:49:09 - trees.J48
- 14:51:31 - trees.J48
- 14:51:53 - trees.J48
- 14:52:26 - trees.J48
- 14:53:15 - trees.J48
- 14:53:58 - trees.J48
- 14:54:39 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	526	77.3529 %
Incorrectly Classified Instances	154	22.6471 %
Kappa statistic	0.3815	
Mean absolute error	0.1854	
Root mean squared error	0.351	
Relative absolute error	72.1314 %	
Root relative squared error	95.7063 %	
Total Number of Instances	680	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.5	0
	0.896	0.548	0.811	0.896	0.851	0.744	1
	0.455	0.103	0.625	0.455	0.526	0.747	2
Weighted Avg.	0.774	0.425	0.758	0.774	0.761	0.745	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	441	51	b = 1
0	102	85	c = 2

After the best hold out percentage was determined the next step was to find the most accurate model using K-fold cross validation. To determine the best number of folds. First, a low number of folds was used. Then, the number of folds was increased until the accuracy began to decrease. Below is the screenshot of the summary of a three fold decision tree.



Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 3

☐ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:51:52 - trees.J48

13:51:58 - trees.J48

13:52:00 - trees.J48

13:52:05 - trees.J48

13:52:09 - trees.J48

13:52:12 - trees.J48

13:52:16 - trees.J48

13:52:19 - trees.J48

13:52:23 - trees.J48

13:52:26 - trees.J48

13:52:30 - trees.J48

13:52:36 - trees.J48

13:52:39 - trees.J48

Classifier output

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	776	77.6	%
Incorrectly Classified Instances	224	22.4	%
Kappa statistic	0.3898		
Mean absolute error	0.1813		
Root mean squared error	0.346		
Relative absolute error	69.6623	%	
Root relative squared error	96.0772	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.329	0
	0.879	0.511	0.827	0.879	0.852	0.707	1
	0.49	0.121	0.592	0.49	0.536	0.708	2
Weighted Avg.	0.776	0.408	0.765	0.776	0.768	0.707	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	647	89	b = 1
0	134	129	c = 2

With three folds the decision tree was 77.6% accurate. When the number of folds was increased to 5, the model became even more accurate. Below is the screenshot of the model with 5 folds.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 5

☐ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:55:00 - trees.J48

13:55:02 - trees.J48

13:55:05 - trees.J48

13:55:08 - trees.J48

13:55:11 - trees.J48

13:55:15 - trees.J48

13:55:21 - trees.J48

13:55:27 - trees.J48

13:55:30 - trees.J48

13:55:33 - trees.J48

13:55:36 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	785	78.5	%
Incorrectly Classified Instances	215	21.5	%
Kappa statistic	0.4121		
Mean absolute error	0.1752		
Root mean squared error	0.3423		
Relative absolute error	67.364	%	
Root relative squared error	95.0538	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.304	0
	0.887	0.5	0.832	0.887	0.859	0.726	1
	0.502	0.113	0.614	0.502	0.552	0.729	2
Weighted Avg.	0.785	0.398	0.774	0.785	0.777	0.726	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	653	83	b = 1
0	131	132	c = 2

As shown, the accuracy of the decision tree's predictions increased to 78.5%. This showed that as the number of folds increased, so did the accuracy of the model. The next step was increasing the number of folds to 10. Below is the screenshot of the 10 fold decision tree analysis.



Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 10

☐ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:55:02 - trees.J48

13:55:05 - trees.J48

13:55:08 - trees.J48

13:55:11 - trees.J48

13:55:15 - trees.J48

13:55:21 - trees.J48

13:55:27 - trees.J48

13:55:30 - trees.J48

13:55:33 - trees.J48

13:55:36 - trees.J48

13:59:45 - functions.MultilayerPerceptron

Classifier output

=== Summary ===

Correctly Classified Instances	797	79.7 %
Incorrectly Classified Instances	203	20.3 %
Kappa statistic	0.4492	
Mean absolute error	0.1729	
Root mean squared error	0.3332	
Relative absolute error	66.4783 %	
Root relative squared error	92.5106 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.275	0
	0.891	0.466	0.842	0.891	0.866	0.726	1
	0.536	0.109	0.638	0.536	0.583	0.729	2
Weighted Avg.	0.797	0.371	0.788	0.797	0.791	0.726	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	656	80	b = 1
0	122	141	c = 2

The accuracy of the model increased again to 79.7%, The number of folds was again increased, this time to 25, the plan was to increase the number of folds until the accuracy of the model started to decrease. Shown below is the screenshot of the 25 fold decision tree analysis.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 25

☐ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

13:55:05 - trees.J48

13:55:08 - trees.J48

13:55:11 - trees.J48

13:55:15 - trees.J48

13:55:21 - trees.J48

13:55:27 - trees.J48

13:55:30 - trees.J48

13:55:33 - trees.J48

13:55:36 - trees.J48

13:59:45 - functions.MultilayerPerceptron

14:49:09 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	784	78.4 %
Incorrectly Classified Instances	216	21.6 %
Kappa statistic	0.4055	
Mean absolute error	0.1747	
Root mean squared error	0.3375	
Relative absolute error	67.1988 %	
Root relative squared error	93.7136 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.262	0
	0.89	0.511	0.829	0.89	0.858	0.738	1
	0.49	0.11	0.614	0.49	0.545	0.741	2
Weighted Avg.	0.784	0.405	0.772	0.784	0.775	0.738	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	655	81	b = 1
0	134	129	c = 2

When the number of folds was increased to 25 the accuracy decreased to 78.4%. This meant that the ideal number of folds was between 10 and 25. After experimenting the ideal amount of folds was determined to be 22. Below is the screenshot of the 22 fold decision tree analysis.

Classifier

Choose J48 -C 0.25 -M 2

Test options

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds 22

☐ Percentage split % 32

More options...

(Nom) winner

Start Stop

Result list (right-click for options)

- 13:55:08 - trees.J48
- 13:55:11 - trees.J48
- 13:55:15 - trees.J48
- 13:55:21 - trees.J48
- 13:55:27 - trees.J48
- 13:55:30 - trees.J48
- 13:55:33 - trees.J48
- 13:55:36 - trees.J48
- 13:59:45 - functions.MultilayerPerceptron
- 14:49:09 - trees.J48
- 14:51:31 - trees.J48

Classifier output

=== Summary ===

Correctly Classified Instances	805	80.5 %
Incorrectly Classified Instances	195	19.5 %
Kappa statistic	0.4644	
Mean absolute error	0.1677	
Root mean squared error	0.3299	
Relative absolute error	64.5103 %	
Root relative squared error	91.6023 %	
Total Number of Instances	1000	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0.001	0	0	0	0.256	0
	0.904	0.47	0.843	0.904	0.872	0.742	1
	0.532	0.095	0.667	0.532	0.592	0.744	2
Weighted Avg.	0.805	0.371	0.796	0.805	0.798	0.742	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
1	665	70	b = 1
0	123	140	c = 2

The maximum accuracy from holding out a percent of the data was only 77.4% whereas the maximum accuracy from K-fold cross validation was 80.5%. Based on the data K-fold cross validation is the best method of cross validation to use for creating a decision tree. The best decision tree we could create was cross validated with 22 folds and we achieved an accuracy of 80.5%

## Neural Net Testing

For our neural network testing, it was first planned to run percentage split tests on 10%, 20%, 30% and 40% holdouts. Not finding a conclusive “best” holdout percentage from these tests, we continued and tested on 50% through 90% holdout at 10% intervals. Some of these tests are shown here to provide the reader with a good understanding of the data. The rest are shown in Appendix A.

## 10% Holdout:

**Classifier**

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

**Test options**

☐ Use training set

☐ Supplied test set

☐ Cross-validation Folds

☒ Percentage split %

(Nom) winner

**Result list (right-click for options)**

- 14:03:33 - functions.MultilayerPerceptron
- 14:06:11 - functions.MultilayerPerceptron
- 14:08:35 - functions.MultilayerPerceptron
- 14:10:09 - functions.MultilayerPerceptron
- 14:12:10 - functions.MultilayerPerceptron
- 14:17:13 - functions.MultilayerPerceptron
- 14:19:51 - functions.MultilayerPerceptron
- 14:22:31 - functions.MultilayerPerceptron
- 14:25:44 - functions.MultilayerPerceptron

**Classifier output**

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	708	78.6667 %
Incorrectly Classified Instances	192	21.3333 %
Kappa statistic	0.446	
Mean absolute error	0.1423	
Root mean squared error	0.3542	
Relative absolute error	51.2662 %	
Root relative squared error	98.1032 %	
Total Number of Instances	900	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0	0.855	0.408	0.857	0.855	0.856	0.593	0
1	0.595	0.145	0.587	0.595	0.591	0.819	1
2	0.595	0.145	0.587	0.595	0.591	0.819	2
Weighted Avg.	0.787	0.34	0.787	0.787	0.787	0.819	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	570	97	b = 1
0	94	138	c = 2

## 30% Holdout:

**Classifier**

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

**Test options**

☐ Use training set

☐ Supplied test set

☐ Cross-validation Folds

☒ Percentage split %

(Nom) winner

**Result list (right-click for options)**

- 14:03:33 - functions.MultilayerPerceptron
- 14:06:11 - functions.MultilayerPerceptron
- 14:08:35 - functions.MultilayerPerceptron
- 14:10:09 - functions.MultilayerPerceptron
- 14:12:10 - functions.MultilayerPerceptron
- 14:17:13 - functions.MultilayerPerceptron
- 14:19:51 - functions.MultilayerPerceptron
- 14:22:31 - functions.MultilayerPerceptron
- 14:25:44 - functions.MultilayerPerceptron

**Classifier output**

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	540	77.1429 %
Incorrectly Classified Instances	160	22.8571 %
Kappa statistic	0.3997	
Mean absolute error	0.1517	
Root mean squared error	0.3664	
Relative absolute error	59.3258 %	
Root relative squared error	99.5037 %	
Total Number of Instances	700	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
0	0.875	0.497	0.82	0.875	0.847	0.332	0
1	0.505	0.125	0.609	0.505	0.552	0.812	1
2	0.505	0.125	0.609	0.505	0.552	0.812	2
Weighted Avg.	0.771	0.393	0.76	0.771	0.764	0.806	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	442	63	b = 1
0	96	98	c = 2

## 50% holdout:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds   
☒ Percentage split %   
More options...

(Nom) winner ⬇

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	404	80.8	%
Incorrectly Classified Instances	96	19.2	%
Kappa statistic	0.5076		
Mean absolute error	0.1341		
Root mean squared error	0.337		
Relative absolute error	51.3595	%	
Root relative squared error	91.0925	%	
Total Number of Instances	500		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.403	0
	0.894	0.408	0.847	0.894	0.87	0.847	1
	0.596	0.106	0.689	0.596	0.639	0.847	2
Weighted Avg.	0.808	0.322	0.8	0.808	0.803	0.846	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	320	38	b = 1
0	57	84	c = 2

## 70% Holdout:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds   
☒ Percentage split %   
More options...

(Nom) winner ⬇

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	250	83.3333	%
Incorrectly Classified Instances	50	16.6667	%
Kappa statistic	0.5828		
Mean absolute error	0.1229		
Root mean squared error	0.3198		
Relative absolute error	45.9323	%	
Root relative squared error	84.7978	%	
Total Number of Instances	300		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.763	0
	0.919	0.367	0.854	0.919	0.885	0.855	1
	0.64	0.081	0.77	0.64	0.699	0.854	2
Weighted Avg.	0.833	0.281	0.826	0.833	0.827	0.855	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	193	17	b = 1
0	32	57	c = 2

## 90% Holdout:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds   
☒ Percentage split %   
More options...

(Nom) winner ▼

Start Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
**14:17:13 - functions.MultilayerPerceptron**  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	82	82	%
Incorrectly Classified Instances	18	18	%
Kappa statistic	0.5455		
Mean absolute error	0.1153		
Root mean squared error	0.3005		
Relative absolute error	45.5317	%	
Root relative squared error	86.0031	%	
Total Number of Instances	100		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	?	0
	0.842	0.25	0.914	0.842	0.877	0.903	1
	0.75	0.158	0.6	0.75	0.667	0.902	2
Weighted Avg.	0.82	0.228	0.839	0.82	0.826	0.903	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	0	0	a = 0
0	64	12	b = 1
0	6	18	c = 2

The highest percentage of correctly classified instances was 83.33% correct, which occurred at 70% holdout percent. This was a surprise, as it needed only 300 data point of the provided 1000 to train a very good neural network. This does make some sense though, as the 300 data point may provide less overfitting that could cause problems when attempting to correctly tune a neural network.

We then tested a neural network using K-Folding, with K-folds of 3, 5, 10 and 25.

K = 3:

Classifier

Choose MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☒ Cross-validation Folds 3  
☐ Percentage split %  
More options...  
(Nom) winner  
Start Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron  
14:32:47 - functions.MultilayerPerceptron

Classifier output

=== Stratified cross-validation ===  
=== Summary ===  

Correctly Classified Instances	817	81.7	%
Incorrectly Classified Instances	183	18.3	%
Kappa statistic	0.5242		
Mean absolute error	0.1266		
Root mean squared error	0.3224		
Relative absolute error	48.6495	%	
Root relative squared error	89.519	%	
Total Number of Instances	1000		

  
=== Detailed Accuracy By Class ===  

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.625	0
	0.882	0.364	0.871	0.882	0.876	0.866	1
	0.639	0.118	0.659	0.639	0.649	0.871	2
Weighted Avg.	0.817	0.299	0.814	0.817	0.816	0.867	

  
=== Confusion Matrix ===  

a	b	c	<-- classified as
0	1	0	a = 0
0	649	87	b = 1
0	95	168	c = 2

K = 5:

Classifier

Choose MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☒ Cross-validation Folds 5  
☐ Percentage split %  
More options...  
(Nom) winner  
Start Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron  
14:32:47 - functions.MultilayerPerceptron  
14:37:16 - functions.MultilayerPerceptron

Classifier output

=== Stratified cross-validation ===  
=== Summary ===  

Correctly Classified Instances	822	82.2	%
Incorrectly Classified Instances	178	17.8	%
Kappa statistic	0.5355		
Mean absolute error	0.1231		
Root mean squared error	0.3147		
Relative absolute error	47.343	%	
Root relative squared error	87.3876	%	
Total Number of Instances	1000		

  
=== Detailed Accuracy By Class ===  

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.209	0
	0.887	0.36	0.873	0.887	0.88	0.872	1
	0.643	0.113	0.671	0.643	0.656	0.876	2
Weighted Avg.	0.822	0.294	0.819	0.822	0.82	0.873	

  
=== Confusion Matrix ===  

a	b	c	<-- classified as
0	1	0	a = 0
0	653	83	b = 1
0	94	169	c = 2

K = 10:

**Test options**

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds

☐ Percentage split %

More options...

(Nom) winner ▼

Start Stop

**Result list (right-click for options)**

- 14:05:53 - functions.MultilayerPerceptron
- 14:06:11 - functions.MultilayerPerceptron
- 14:08:35 - functions.MultilayerPerceptron
- 14:10:09 - functions.MultilayerPerceptron
- 14:12:10 - functions.MultilayerPerceptron
- 14:17:13 - functions.MultilayerPerceptron
- 14:19:51 - functions.MultilayerPerceptron
- 14:22:31 - functions.MultilayerPerceptron
- 14:25:44 - functions.MultilayerPerceptron
- 14:32:47 - functions.MultilayerPerceptron
- 14:37:16 - functions.MultilayerPerceptron

**Classifier output**

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	835	83.5	%
Incorrectly Classified Instances	165	16.5	%
Kappa statistic	0.5699		
Mean absolute error	0.1151		
Root mean squared error	0.3079		
Relative absolute error	44.2527	%	
Root relative squared error	85.4941	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.55	0
	0.895	0.333	0.882	0.895	0.889	0.873	1
	0.669	0.104	0.696	0.669	0.682	0.879	2
Weighted Avg.	0.835	0.273	0.832	0.835	0.834	0.874	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	659	77	b = 1
0	87	176	c = 2

K = 25:

**Test options**

☐ Use training set

☐ Supplied test set Set...

☒ Cross-validation Folds

☐ Percentage split %

More options...

(Nom) winner ▼

Start Stop

**Result list (right-click for options)**

- 14:05:53 - functions.MultilayerPerceptron
- 14:06:11 - functions.MultilayerPerceptron
- 14:08:35 - functions.MultilayerPerceptron
- 14:10:09 - functions.MultilayerPerceptron
- 14:12:10 - functions.MultilayerPerceptron
- 14:17:13 - functions.MultilayerPerceptron
- 14:19:51 - functions.MultilayerPerceptron
- 14:22:31 - functions.MultilayerPerceptron
- 14:25:44 - functions.MultilayerPerceptron
- 14:32:47 - functions.MultilayerPerceptron
- 14:37:16 - functions.MultilayerPerceptron
- 14:45:20 - functions.MultilayerPerceptron
- 14:56:19 - functions.MultilayerPerceptron

**Classifier output**

=== Stratified cross-validation ===

=== Summary ===

Correctly Classified Instances	821	82.1	%
Incorrectly Classified Instances	179	17.9	%
Kappa statistic	0.53		
Mean absolute error	0.1229		
Root mean squared error	0.3204		
Relative absolute error	47.285	%	
Root relative squared error	88.9575	%	
Total Number of Instances	1000		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.386	0
	0.89	0.371	0.87	0.89	0.88	0.872	1
	0.631	0.11	0.672	0.631	0.651	0.876	2
Weighted Avg.	0.821	0.302	0.817	0.821	0.819	0.873	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	655	81	b = 1
0	97	166	c = 2

The best number of folds we found was k = 10, which trained the neural network to correctly classify instances in 83.5% of cases.

K-folding was much more consistent in percentage classified correctly than holdout percentages when looking across a wide range of different holdout percentages or k-folds. K-folding was on average a better method, too. K-folding and holdout percentages both were able to produce almost exactly the same level of correctly classified instances at peak, however. Because of this, we are unable to draw any strong conclusions as to which is a better method for training a neural network.



## Appendix A:

20% Holdout:

Classifier

Choose **MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a**

Test options

☐ Use training set

☐ Supplied test set 

Set...

☐ Cross-validation Folds 

10

☒ Percentage split %

More options...

(Nom) winner

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron

14:06:11 - functions.MultilayerPerceptron

14:08:35 - functions.MultilayerPerceptron

14:10:09 - functions.MultilayerPerceptron

14:12:10 - functions.MultilayerPerceptron

14:17:13 - functions.MultilayerPerceptron

14:19:51 - functions.MultilayerPerceptron

14:22:31 - functions.MultilayerPerceptron

14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	612	76.5 %
Incorrectly Classified Instances	188	23.5 %
Kappa statistic	0.3936	
Mean absolute error	0.1562	
Root mean squared error	0.3695	
Relative absolute error	60.3668 %	
Root relative squared error	102.1035 %	
Total Number of Instances	800	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.68	0
	0.846	0.458	0.835	0.846	0.841	0.808	1
	0.545	0.153	0.563	0.545	0.554	0.799	2
Weighted Avg.	0.765	0.376	0.762	0.765	0.763	0.806	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	496	90	b = 1
0	97	116	c = 2

40% Holdout:

Classifier

Choose **MultilayerPerceptron -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a**

Test options

☐ Use training set

☐ Supplied test set 

Set...

☐ Cross-validation Folds 

10

☒ Percentage split %

More options...

(Nom) winner

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron

14:06:11 - functions.MultilayerPerceptron

14:08:35 - functions.MultilayerPerceptron

14:10:09 - functions.MultilayerPerceptron

14:12:10 - functions.MultilayerPerceptron

14:17:13 - functions.MultilayerPerceptron

14:19:51 - functions.MultilayerPerceptron

14:22:31 - functions.MultilayerPerceptron

14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	466	77.6667 %
Incorrectly Classified Instances	134	22.3333 %
Kappa statistic	0.4153	
Mean absolute error	0.1534	
Root mean squared error	0.3637	
Relative absolute error	59.3431 %	
Root relative squared error	98.2343 %	
Total Number of Instances	600	

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.165	0
	0.884	0.494	0.819	0.884	0.85	0.83	1
	0.509	0.116	0.632	0.509	0.564	0.832	2
Weighted Avg.	0.777	0.387	0.765	0.777	0.768	0.83	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	380	50	b = 1
0	83	86	c = 2

## 60% Holdout:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds   
☒ Percentage split %   
More options...

(Nom) winner ⬇

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
**14:19:51 - functions.MultilayerPerceptron**  
14:22:31 - functions.MultilayerPerceptron  
14:25:44 - functions.MultilayerPerceptron

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	323	80.75	%
Incorrectly Classified Instances	77	19.25	%
Kappa statistic	0.5207		
Mean absolute error	0.1275		
Root mean squared error	0.3235		
Relative absolute error	48.074	%	
Root relative squared error	85.6437	%	
Total Number of Instances	400		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.083	0
	0.896	0.4	0.839	0.896	0.867	0.861	1
	0.605	0.103	0.713	0.605	0.655	0.865	2
Weighted Avg.	0.808	0.311	0.8	0.808	0.802	0.86	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	251	29	b = 1
0	47	72	c = 2

## 80% Holdout:

Classifier

Choose **MultilayerPerceptron** -L 0.3 -M 0.2 -N 500 -V 0 -S 0 -E 20 -H a

Test options

☐ Use training set  
☐ Supplied test set Set...  
☐ Cross-validation Folds   
☒ Percentage split %   
More options...

(Nom) winner ⬇

Start

Stop

Result list (right-click for options)

14:03:33 - functions.MultilayerPerceptron  
14:06:11 - functions.MultilayerPerceptron  
14:08:35 - functions.MultilayerPerceptron  
14:10:09 - functions.MultilayerPerceptron  
14:12:10 - functions.MultilayerPerceptron  
14:17:13 - functions.MultilayerPerceptron  
14:19:51 - functions.MultilayerPerceptron  
14:22:31 - functions.MultilayerPerceptron  
**14:25:44 - functions.MultilayerPerceptron**

Classifier output

=== Evaluation on test split ===

=== Summary ===

Correctly Classified Instances	166	83	%
Incorrectly Classified Instances	34	17	%
Kappa statistic	0.5655		
Mean absolute error	0.1192		
Root mean squared error	0.3078		
Relative absolute error	45.0615	%	
Root relative squared error	83.6153	%	
Total Number of Instances	200		

=== Detailed Accuracy By Class ===

	TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area	Class
	0	0	0	0	0	0.608	0
	0.903	0.357	0.867	0.903	0.884	0.889	1
	0.655	0.097	0.72	0.655	0.686	0.887	2
Weighted Avg.	0.83	0.284	0.822	0.83	0.825	0.887	

=== Confusion Matrix ===

a	b	c	<-- classified as
0	1	0	a = 0
0	130	14	b = 1
0	19	36	c = 2