MASTER OF TECHNOLOGY (INTELLIGENT SYSTEMS)

PROJECT REPORT (CA 1)
Tutor: Dr Zhu Fang Ming

Supervised Learning Techniques Applied to Competitive Gameplay Results

TEAM MEMBERS
YE CHANGHE
LIM LI WEI
PREM S/O PIRAPALA CHANDRAN

Contents

EXECUTIVE SUMMARY

Machine learning (ML) is an application of artificial intelligence (AI) which provides systems the functionality to learn automatically to improve from experience without being explicitly programmed. The process of learning begins with observations of data, such as examples of footages, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide to learn without human intervention or assistance and adjust actions accordingly. Machine learning is applied to a range of problems which can be distinguished broadly as 2 types: supervised and unsupervised. Supervised machine learning problems use training data which have already have labelled features as examples which are used to make predictions for which category based on a classification problem or prediction on a continuous scale for a regression value. Unsupervised machine learning problems are problems where the machine learning algorithms are applied to organise the data. The effectiveness and accuracy of a Machine Learning techniques depends on the algorithm(s) that is applied to the problem, from which there is a range.

For this project, we have applied supervised learning techniques using a range of different classifier models to predict the outcome of a competitive gameplay for an online game based on features of strategies used by the player. Based on the outcome of the techniques applied, we obtained an estimated 97% accuracy which can be attributed to the binary classification outcome to predict (win/lose). The report will elaborate further on the design of the models and provide further details on how the different strategies adopted influence the outcome.

1.0) PROBLEM DESCRIPTION

League of Legends is a popular MOBA (Multiplayer Online Battle Arena) game developed by RIOT GAMES INC. with over 27 million active players daily. With 5 players on each team (5 vs 5), players pick their own “Champions”, each with different roles and abilities, and attempt to take their key objective, which is to destroy the “Nexus” of the opposing team.

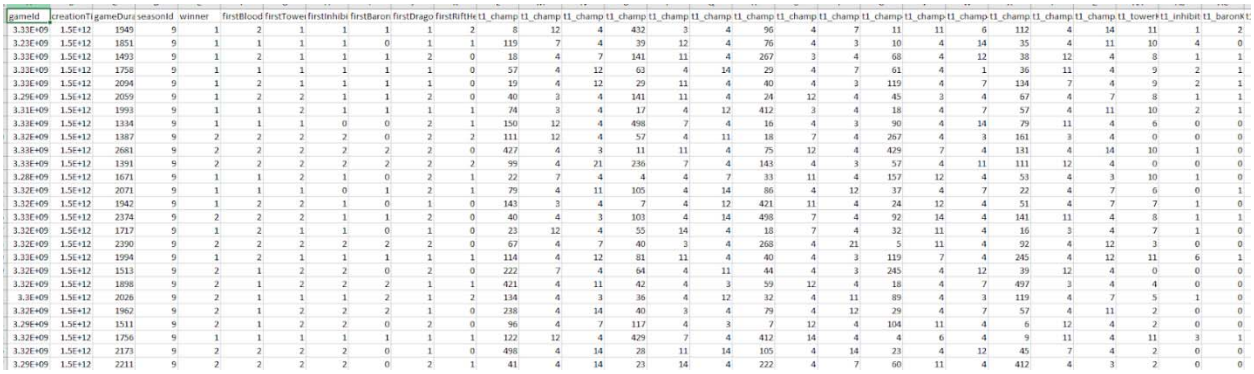
In order to win a match by taking the objective, there are several key factors that will affect the probability of winning. These are:

- Champion pick composition
- Champion ban composition
- First blood (First Kill)
- First Dragon, First Baron, Number of bosses killed (Neutral Bosses)
- Summoner Spells Picked
- Number of Towers Destroyed

1.1) ANALYSIS OF DATA SET

For this assignment, we have decided to use a dataset from Kaggle, which consists of over 50000 ranked games played in **Europe West** Servers. These highlight several features that will heavily impact the chance of victory for either team.

URL: https://www.kaggle.com/datasnaek/league-of-legends

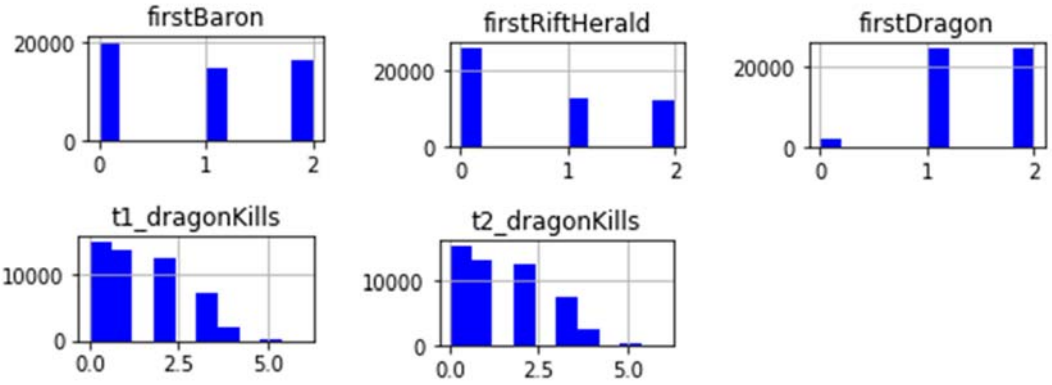


gameId	creationTi	gameDura	seasonId	winner	firstBlood	firstTower	firstInhibi	firstBaron	firstDrago	firstRiftHe	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_champ	t1_tower	t1_inhibit	t1_baronK
3.33E+09	1.5E+12	1949	9	1	2	1	1	1	1	2	8	12	4	432	3	4	96	4	7	11	11	6	112	4	14	11	1	2
3.23E+09	1.5E+12	1851	9	1	1	1	1	0	1	1	119	7	4	39	12	4	76	4	3	10	4	14	35	4	11	10	4	0
3.33E+09	1.5E+12	1493	9	1	2	1	1	1	2	0	18	4	7	141	11	4	267	3	4	68	4	12	38	12	4	8	1	1
3.33E+09	1.5E+12	1758	9	1	1	1	1	1	1	0	57	4	12	63	4	14	29	4	7	61	4	1	36	11	4	9	2	1
3.33E+09	1.5E+12	2094	9	1	2	1	1	1	1	0	19	4	12	29	11	4	40	4	3	119	4	7	134	7	4	9	2	1
3.29E+09	1.5E+12	2059	9	1	2	2	1	1	2	0	40	3	4	141	11	4	24	12	4	45	3	4	67	4	7	8	1	1
3.31E+09	1.5E+12	1993	9	1	1	2	1	1	1	1	74	3	4	17	4	12	412	3	4	18	4	7	57	4	11	10	2	1
3.33E+09	1.5E+12	1334	9	1	1	1	0	0	2	1	150	12	4	498	7	4	16	4	3	90	4	14	79	11	4	6	0	0
3.32E+09	1.5E+12	1387	9	2	2	2	2	0	2	2	111	12	4	57	4	11	18	7	4	267	4	3	161	3	4	0	0	0
3.33E+09	1.5E+12	2681	9	2	2	2	2	2	2	0	427	4	3	11	11	4	75	12	4	429	7	4	131	4	14	10	1	0
3.33E+09	1.5E+12	1391	9	2	2	2	2	2	2	2	99	4	21	236	7	4	143	4	3	57	4	11	111	12	4	0	0	0
3.28E+09	1.5E+12	1671	9	1	1	2	1	0	2	1	22	7	4	4	4	7	33	11	4	157	12	4	53	4	3	10	1	0
3.32E+09	1.5E+12	2071	9	1	1	1	0	1	2	1	79	4	11	105	4	14	86	4	12	37	4	7	22	4	7	6	0	1
3.32E+09	1.5E+12	1942	9	1	2	2	1	0	1	0	143	3	4	7	4	12	421	11	4	24	12	4	51	4	7	7	1	0
3.33E+09	1.5E+12	2374	9	2	2	2	1	1	2	0	40	4	3	103	4	14	498	7	4	92	14	4	141	11	4	8	1	1
3.32E+09	1.5E+12	1717	9	1	2	1	1	0	1	0	23	12	4	55	14	4	18	7	4	32	11	4	16	3	4	7	1	0
3.32E+09	1.5E+12	2390	9	2	2	2	2	2	2	0	67	4	7	40	3	4	268	4	21	5	11	4	92	4	12	3	0	0
3.33E+09	1.5E+12	1994	9	1	2	1	1	1	1	1	114	4	12	81	11	4	40	4	3	119	7	4	245	4	12	11	6	1
3.32E+09	1.5E+12	1513	9	2	1	2	2	0	2	0	222	7	4	64	4	11	44	4	3	245	4	12	39	12	4	0	0	0
3.32E+09	1.5E+12	1898	9	2	1	2	2	2	1	1	421	4	11	42	4	3	59	12	4	18	4	7	497	3	4	4	0	0
3.3E+09	1.5E+12	2026	9	2	1	1	1	2	1	2	134	4	3	36	4	12	32	4	11	89	4	3	119	4	7	5	1	0
3.32E+09	1.5E+12	1962	9	2	1	2	2	2	1	0	238	4	14	40	3	4	79	4	12	29	4	7	57	4	11	2	0	0
3.29E+09	1.5E+12	1511	9	2	1	2	2	0	2	0	96	4	7	117	4	3	7	12	4	104	11	4	6	12	4	2	0	0
3.32E+09	1.5E+12	1756	9	1	1	1	1	1	1	1	122	12	4	429	7	4	412	14	4	4	6	4	9	11	4	11	3	1
3.32E+09	1.5E+12	2173	9	2	2	2	2	0	1	0	498	4	14	28	11	14	105	4	14	23	4	12	45	7	4	2	0	0
3.29E+09	1.5E+12	2211	9	2	2	2	2	0	2	1	41	4	14	23	14	4	222	4	7	60	11	4	412	4	3	2	0	0

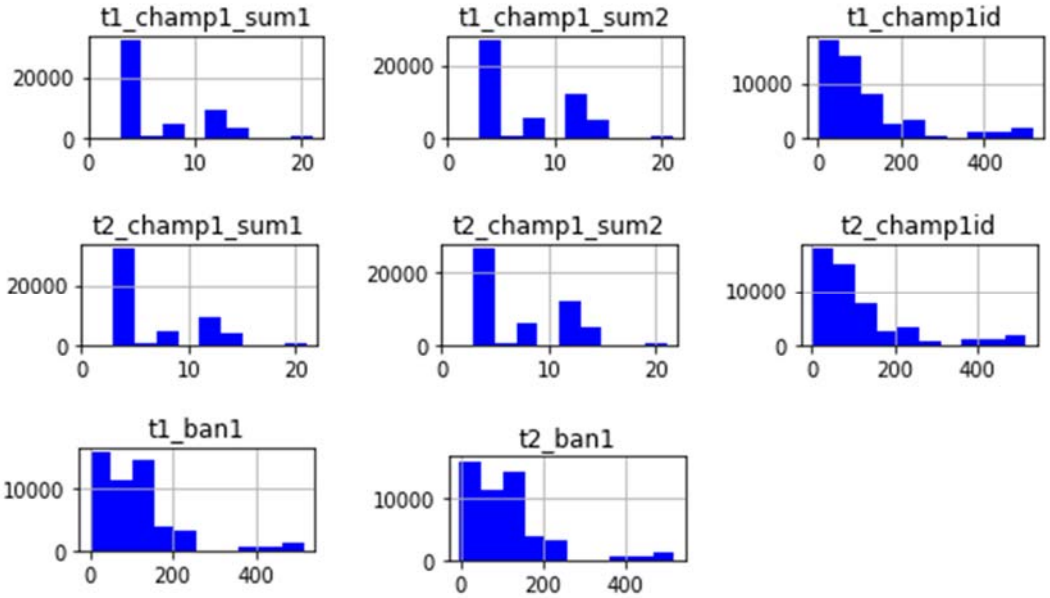
These are the features available in this data (Total of 61 Features)

- Game ID (Unique Number)
- Creation Time (in Epoch Timestamp format)
- Game Duration (in seconds)
- Season ID
- Winner (1 = team1, 2 = team2)
- First Baron, dragon, tower, blood, inhibitor and Rift Herald
  (1 = team1, 2 = team2, 0 = none)
- Champions and summoner spells for each team
  (Stored as Riot's champion and summoner spell IDs)
- The number of towers, inhibitor, Baron, dragon and Rift Herald kills each team has
- The 5 bans of each team (Again, champion IDs are used)

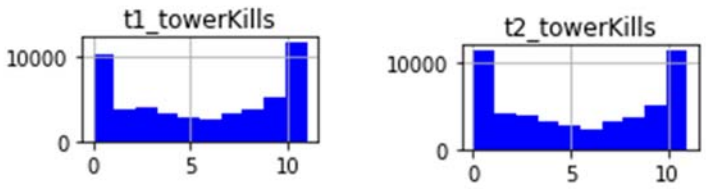
Taking a quick look at our data, the distribution of the data based on histograms plotted show the following observations. (A histogram is used to plot the frequency of score occurrences in a continuous data set that has been divided into classes)



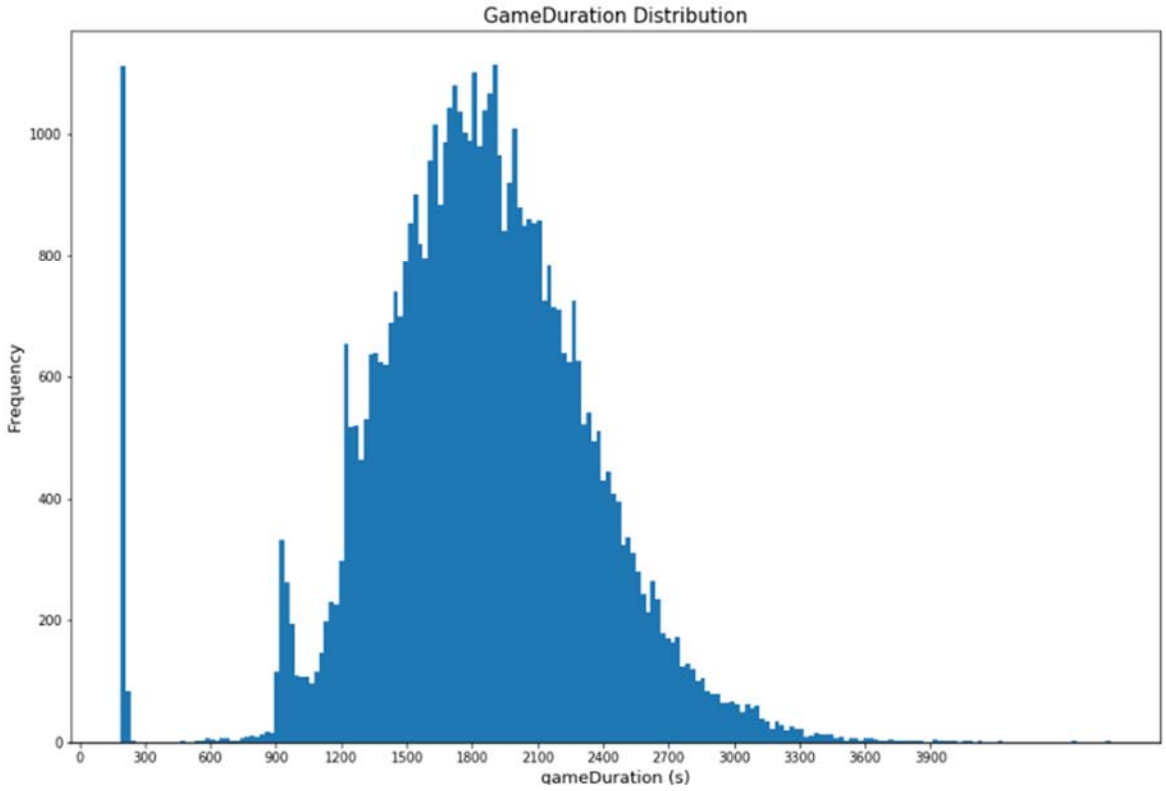
- People prefer not to kill harder neutral boss (Baron/Rift Herald) as it might be risky.
- Dragons are a lot easier and teams will try to kill them first to gain an edge, in a substantial amount of games, teams may choose not to pursue them as well.



- The charts look very alike proves that there is a ‘meta’ choice in which almost all players will aim to pick, and ban.
- The right pick in the favourite champions/spells will contribute a big factor to winning the game.



- This shows the distribution of total tower kills for all the games.
- From the data, this indicates that a majority of the games are lost very badly when lost, and triumphant victory when won, as shown by the bias in the data (Either most or all of the towers destroyed or Little to None destroyed.)
- Team 2 is performing just a bit worse, perhaps due to the geography the team is based on. (More chances of having no to a few tower kills)

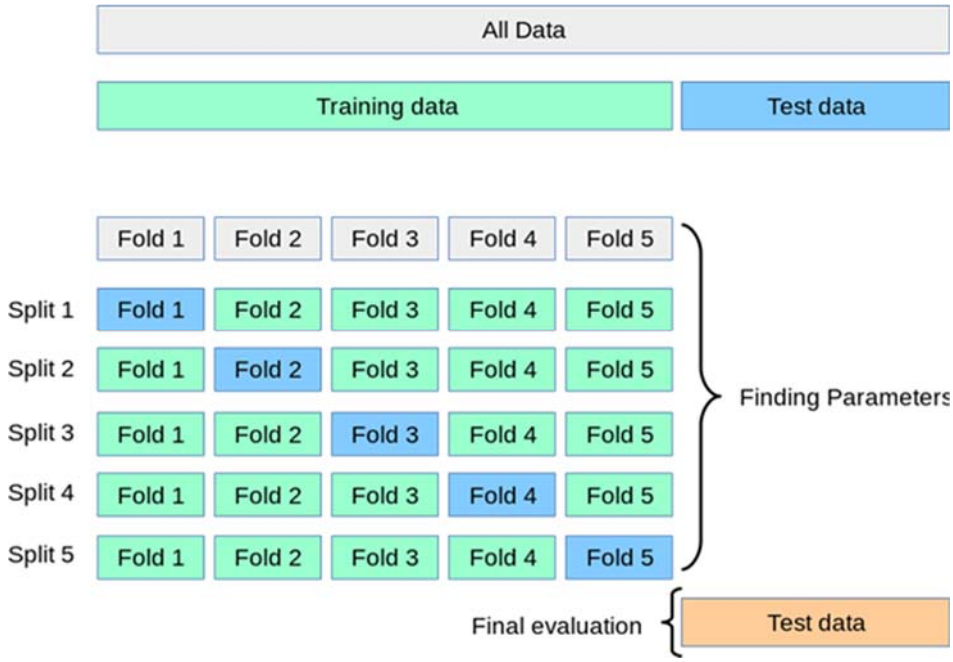


- Games normally end in 35 minutes.
- Quite a number of game end very early, these might be caused by an early surrender, due to multiple reasons. We will exclude these outliers from our data.

This will primarily be a classification problem, where our model will classify the data by predicting which team will win, given the input features. Before training the model, we do some data processing and cleaning in a few stages:

- Handling for any missing Data
- Dropping “Winner” and assign it to Output
- Dropping data that is not so useful (e.g. Game ID and Season ID)
- Removing data where matches end too early (Due to Surrendering)

We will then split these data into training and testing data (1/3 test data).



In some of the later steps where we do grid search to find the best hyperparameter, or when we do ensemble, we will use k-fold cross validation on the training data, where part of the training data is used for validation.

2.0) TOOLS & TECHNIQUES

For this assignment, we are using python 3.7 as our coding language, importing pandas and numpy for data loading and pre-processing. For machine learning part we mainly leverage the packages under sklearn to do all the model building and training. Graphviz and matplotlib help us to display the result.

2.1) CLASSIFIER MODELS

For this problem, we have chosen to use these classifier models to solve it:

1. Decision Tree Method
2. Multilayer Perceptron Neural Network Method (1 Hidden Layer)
3. Logistic Regression Method
4. Voting Ensemble Model (For all 3)
5. Bagging Ensemble Model (For Neural Network)

2.2) MODEL PERFORMANCE EVALUATION

After a model's training, we can use the test data to predict results. These results will show how correctly a model is classifying the test data.

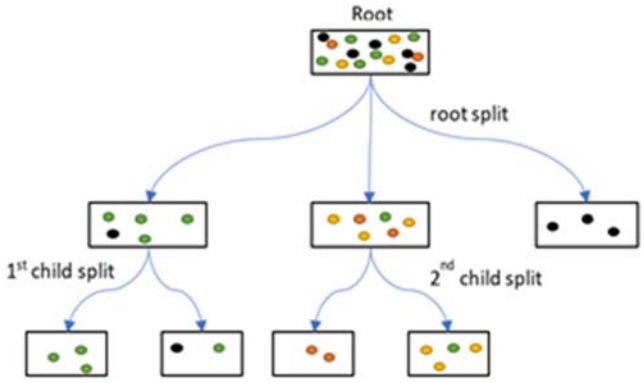
In order to interpret the performance of the machine learning models, it is necessary to compare the results of the different models using a set of standard and consistent evaluation metrics that can be applied across the models. As different evaluation metrics are used for different kinds of problems to discriminate the performance, based on the type of problem here (classification) and the implementation plan, we have chosen to focus comparison based on two metrics below:

No	Evaluation Metric	Description
1	Confusion Matrix	Shows an overview for true/false positive/negative, which can compute several important gauges such as error rate, accuracy, sensitivity, specificity, and precision.Scikit-Learn also helps to compute f1 score which is a better measure for performance.
2	ROC Curve	Shows a plot of sensitivity vs specificity. Allows a fast evaluation from visual inspection.
3	AUC Score	Scoring from the Area Under Curve Metric

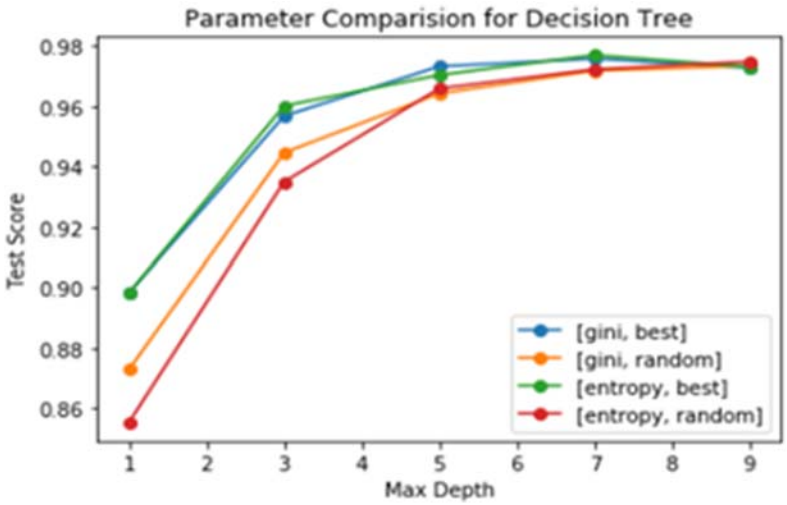
3.0) DESIGN OF THE MODELS & FINDINGS

For this segment of the report, we detail the design of the different models where we provide a diagram illustrating the model operation, hyperparameters defined and performance of the model based on the evaluation metrics of Confusion Matrix and ROC as explained earlier. Hyperparameters are the settings of an algorithm within each of the models that can be adjusted to optimize performance

Decision Tree



Firstly, we applied a decision tree to classify the data. And pre-define several hyper-parameters for grid search and output the best result of the accuracy:



The 'Hyperparameters' defined in this case is:

- **criterion (The function to measure the quality of a split):**

  ["gini", "entropy"]

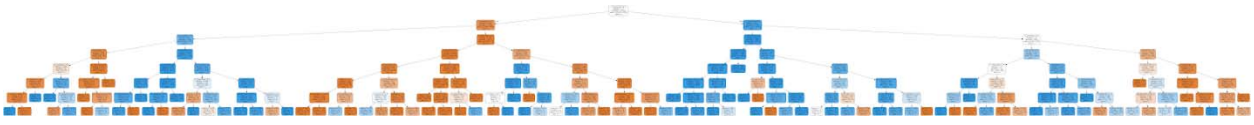
- **max_depth (The maximum depth of the tree):**

  [1, 3, 5, 7, 9]
- **splitter (The strategy used to choose the split at each node):**

  ["best", "random"]

From the results, it seems that the one with "**entropy**" criterion, "**best**" splitter and **7** max depth is outperforming the rest of the values.

The below picture is a decision tree generated by the model with the best hyperparameters



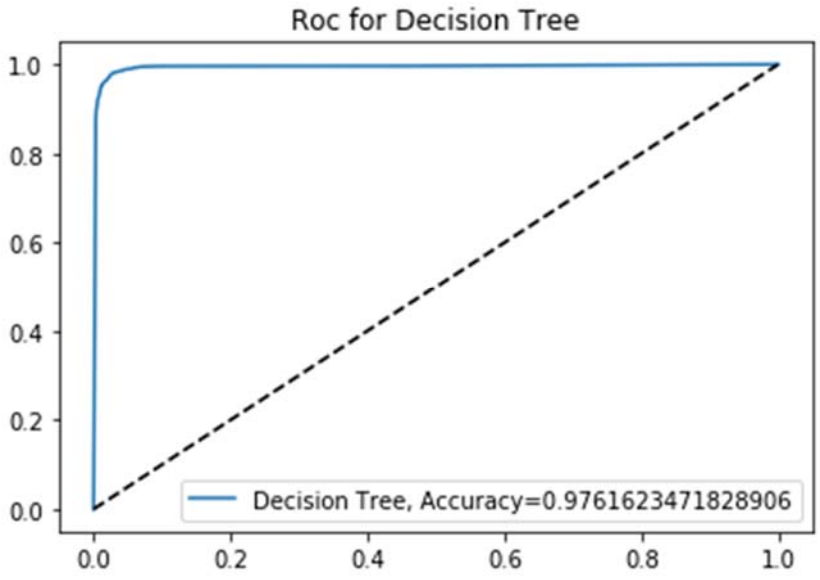
Performance:

- Accuracy on test set for Decision Tree: **0.976**
- Confusion Matrix

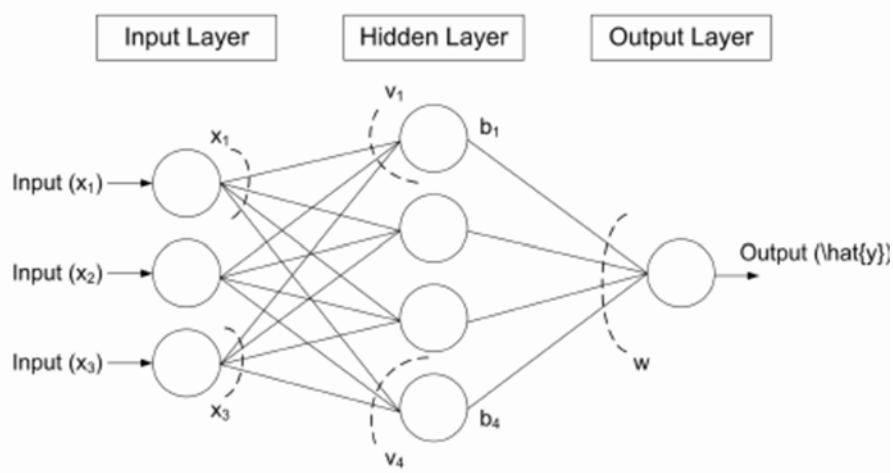
Total number N = 16,765	Predicted: Team 1	Predicted: Team 2
Actual: Team 1	8270	217
Actual: Team 2	183	8095

	precision	recall	f1-score	support
0	0.98	0.97	0.98	8487
1	0.97	0.98	0.98	8278
accuracy			0.98	16765
macro avg	0.98	0.98	0.98	16765
weighted avg	0.98	0.98	0.98	16765

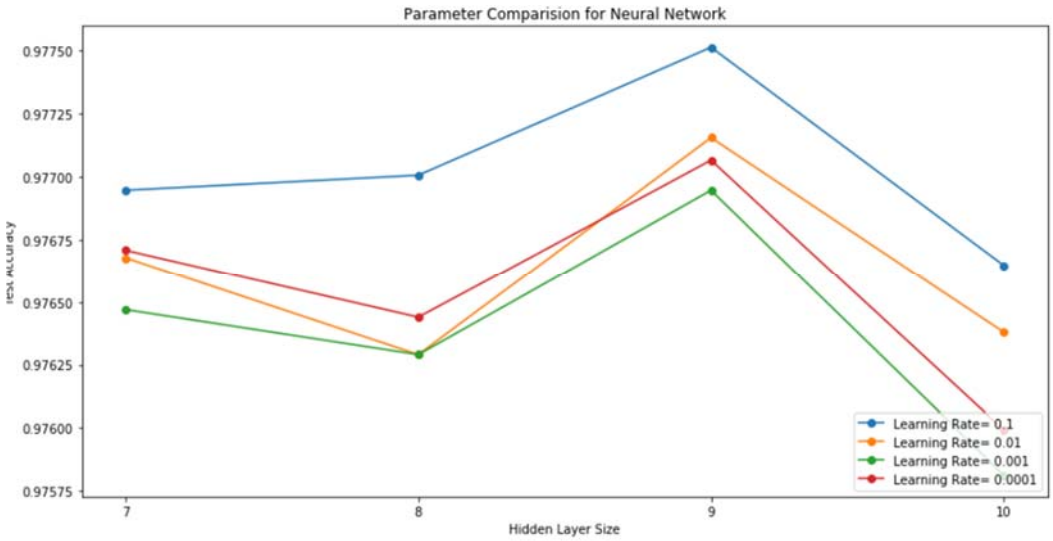
- ROC diagram



Neural Network



In our next model, we are using the neural network classifier to classify the data. For this assignment, we will be using multilayer perceptron design with 1 hidden layer, and we do grid search for the best parameter (Using K−Fold Cross Verification where K = 5). Grid search result as shown below.



The 'Hyperparameters' defined in this case is:

- **alpha (L2 penalty (regularization term) parameter):**

  [0.1, 0.01, 0.001, 0.0001, 0.00001]

- **hidden_layer_sizes (The ith element represents the number of neurons in the ith hidden layer):**

  [7, 8, 9, 10]

From the results, it seems that the one with **0.01** L2 penalty and **9** hidden nodes is outperforming the rest of the values.

Performance:

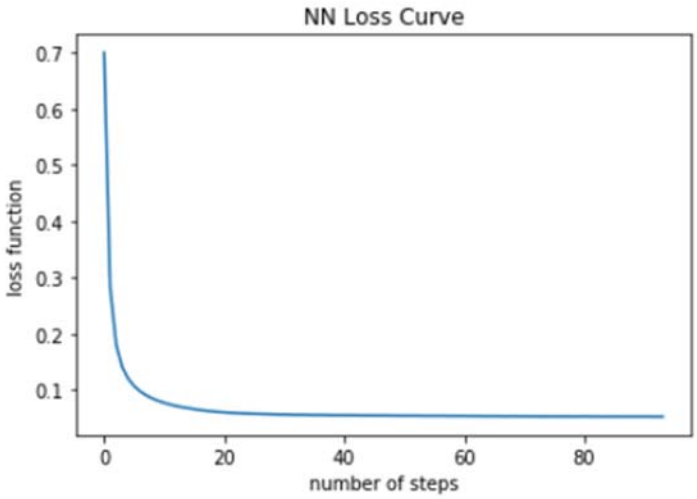
- Accuracy on test set for Neural Network: **0.976**
- Confusion Matrix

Total number N = 16,765	Predicted: Team 1	Predicted: Team 2
Actual: Team 1	8287	200
Actual: Team 2	192	8086

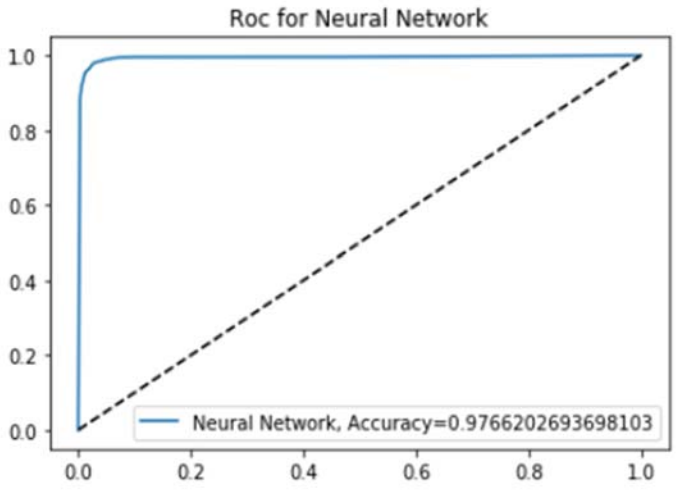
	precision	recall	f1-score	support
0	0.98	0.98	0.98	8487
1	0.98	0.98	0.98	8278
accuracy			0.98	16765
macro avg	0.98	0.98	0.98	16765
weighted avg	0.98	0.98	0.98	16765

```
Accuracy on training set for Neural Network: 0.983
Accuracy on test set for Neural Network: 0.977
```

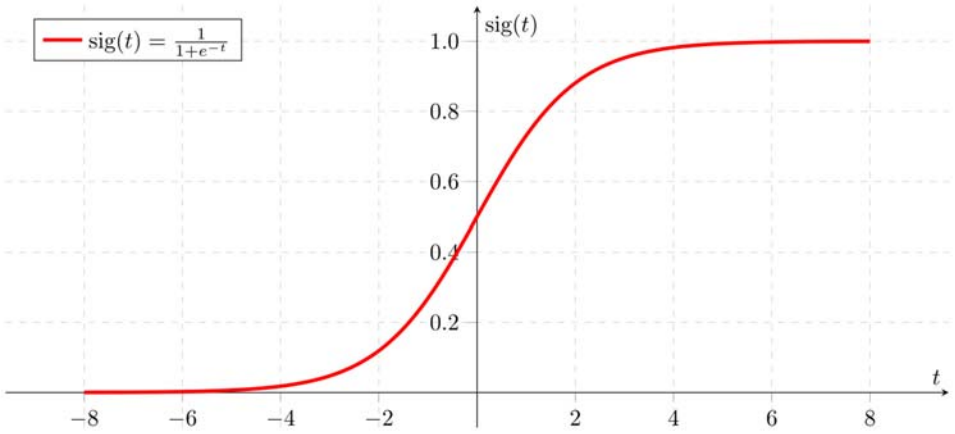
- Loss Curve diagram



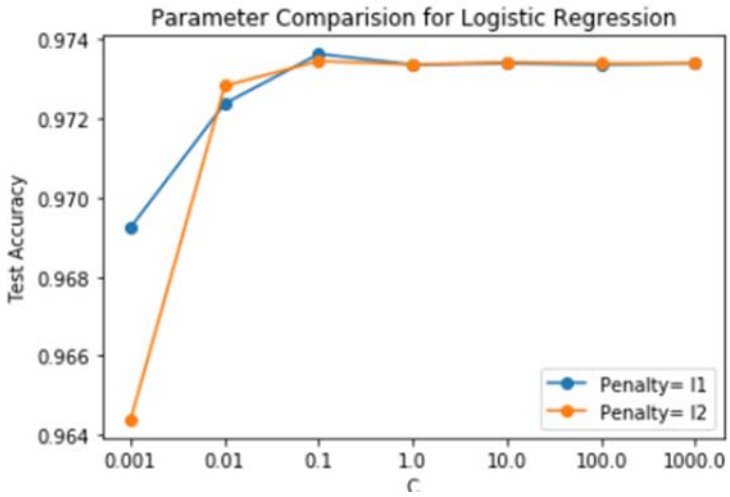
- ROC diagram


Roc for Neural Network
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.0
Neural Network, Accuracy=0.9766202693698103

Logistic Regression



We also use logistic regression to do classification for this problem. Logistic regression can be used as the output/result is in binary (Team 1 or Team 2).



The 'Hyperparameters' defined in this case is:

- **C (Inverse of regularization strength):**

  [0.001, 0.01, 0.1, 1.0, 10.0, 100.0, 1000.0]

- penalty (**Used to specify the norm used in the penalization**):

  ["L1", "L2"]

Referring to the grid search comparison, we get the highest test accuracy when C = **0.1**, using **L1** Regularization.

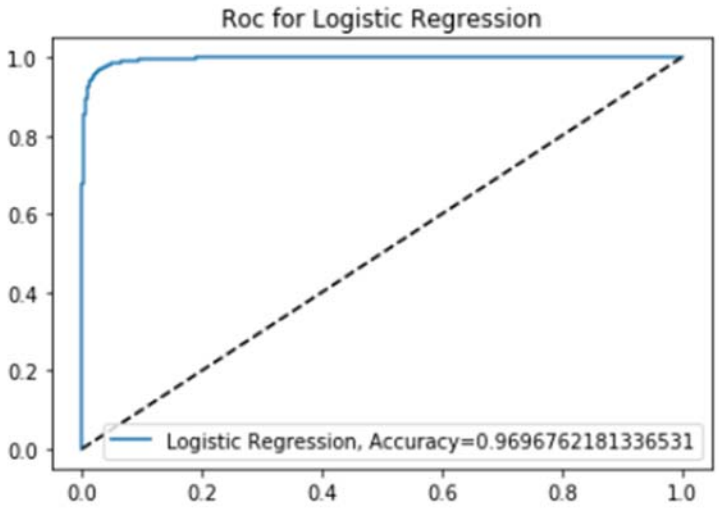
Performance:

- Accuracy on test set for Logistic Regression: **0.970**
- Confusion Matrix

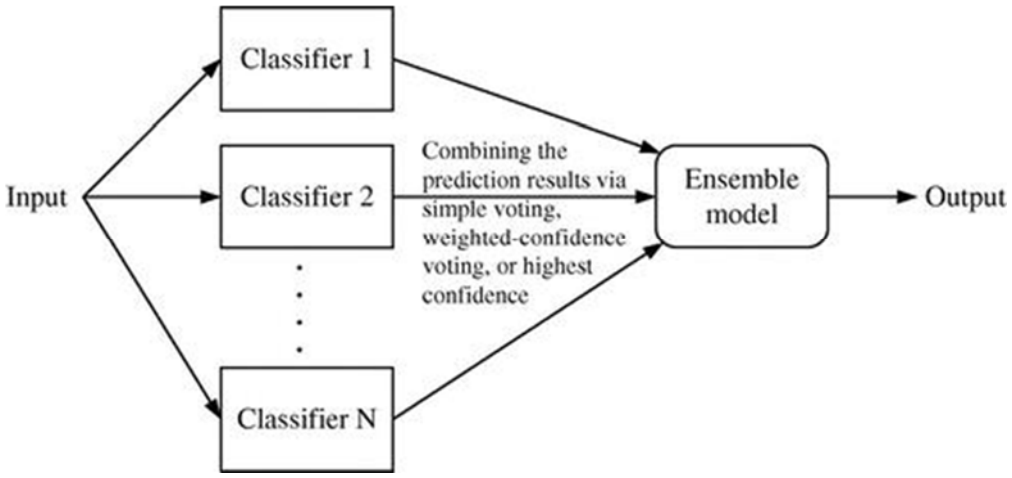
Total number N = 16,765	Predicted: Team 1	Predicted: Team 2
Actual: Team 1	8245	242
Actual: Team 2	266	8012

	precision	recall	f1-score	support
0	0.97	0.97	0.97	8487
1	0.97	0.97	0.97	8278
accuracy			0.97	16765
macro avg	0.97	0.97	0.97	16765
weighted avg	0.97	0.97	0.97	16765

- ROC diagram



Voting Ensemble (Across 3 different Models)



After going through the above 3 models, we can feed them into a voting ensemble model to attempt to get a better accuracy. Output/Results will be based on a certain vote ruling.

For the Ruling, both hard and soft classifier is used here.

Hard classifier model does a Majority Vote, while for soft Classifier, every vote has a weight, in which an average is taken to obtain the final output.

```
Accuracy: 0.9738 (+/- 0.0018) [Logistic Regression]
Accuracy: 0.9772 (+/- 0.0007) [Neural Network]
Accuracy: 0.9750 (+/- 0.0015) [Decision Tree]
Accuracy: 0.9779 (+/- 0.0008) [Voting_Classifier_Hard]
Accuracy: 0.9787 (+/- 0.0007) [Voting_Classifier_Soft]
```

Using K-Fold = 5 cross verification for the above 3 models, the above comparison table is constructed. This shows that ensemble models do perform better than single models, with Soft Classifier being better than hard classifier

From here, we construct the confusion matrix/ROC curve for the soft classifier.

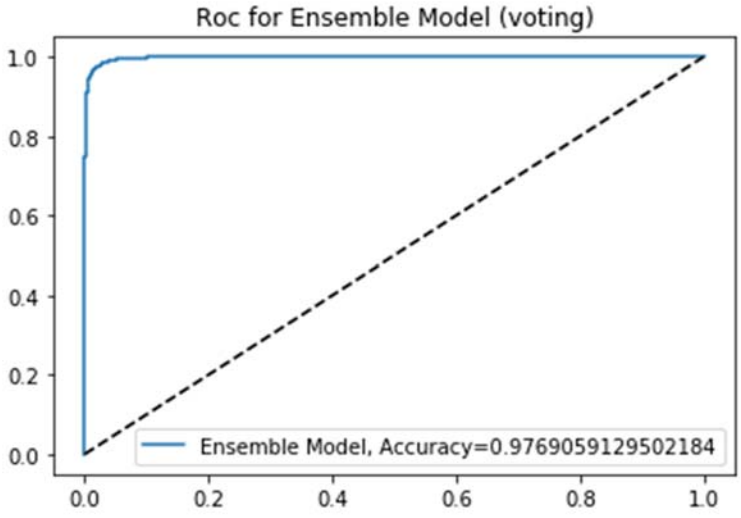
Performance:

- Accuracy on test set for Logistic Regression: **0.977**
- Confusion Matrix

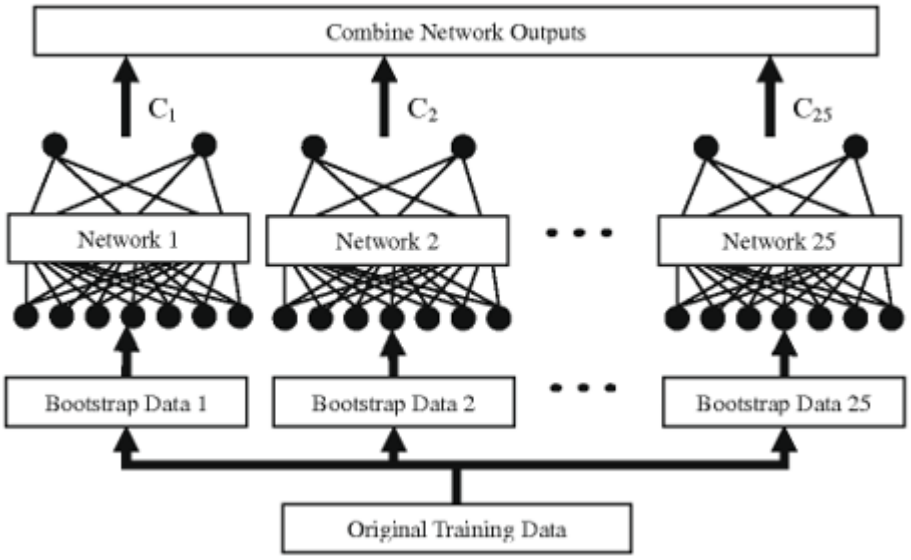
Total number N = 16,765	Predicted: Team 1	Predicted: Team 2
Actual: Team 1	8298	189
Actual: Team 2	198	8080

	precision	recall	f1-score	support
0	0.98	0.98	0.98	8487
1	0.98	0.98	0.98	8278
accuracy			0.98	16765
macro avg	0.98	0.98	0.98	16765
weighted avg	0.98	0.98	0.98	16765

- ROC diagram



Bagging Ensemble (NN)



For this model, we are doing the bootstrap aggregation approach, where bootstrap samples are taken by random out of training data and performed N times on the same model. We are taking the Neural Network model here to be used for the Bagging Model.

In this scenario, we will be using N=25 to perform the ensemble classification.

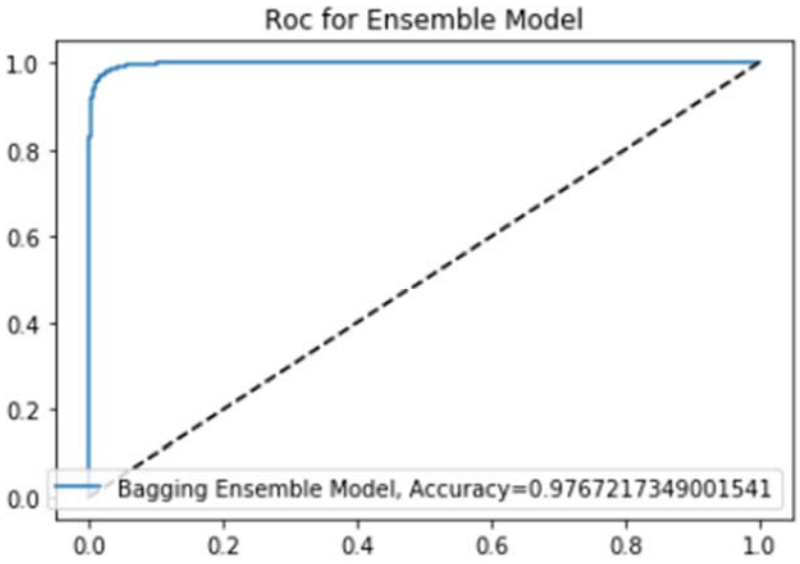
Performance:

- Accuracy on test set for Neural Network: **0.977**
- Confusion Matrix

Total number N = 16,765	**Predicted: Team 1**	**Predicted: Team 2**
Actual: Team 1	**8300**	**187**
Actual: Team 2	**203**	**8075**

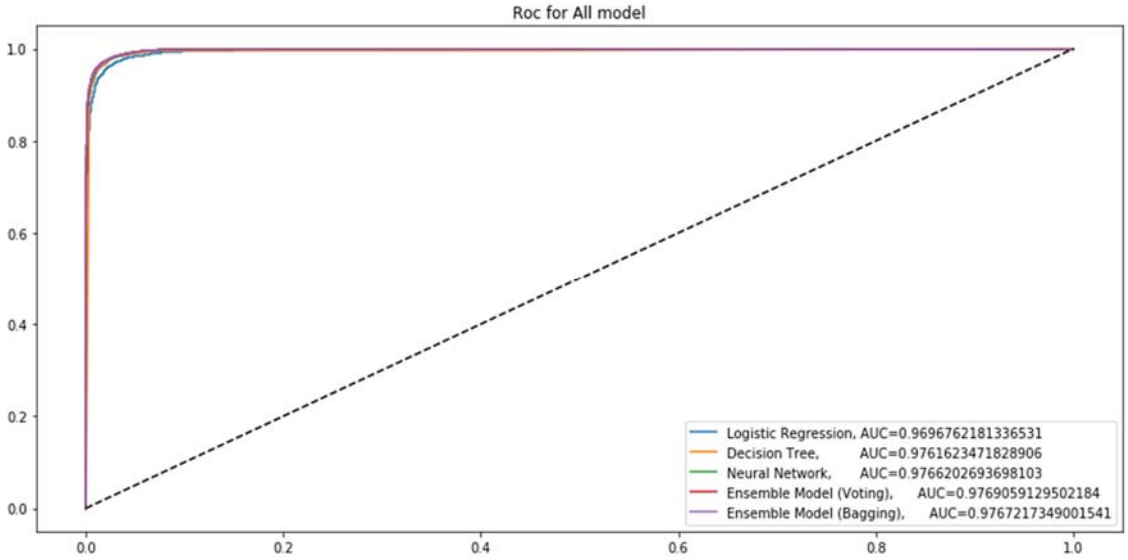
	precision	recall	f1-score	support
0	0.98	0.98	0.98	8487
1	0.98	0.98	0.98	8278
accuracy			0.98	16765
macro avg	0.98	0.98	0.98	16765
weighted avg	0.98	0.98	0.98	16765

- ROC diagram

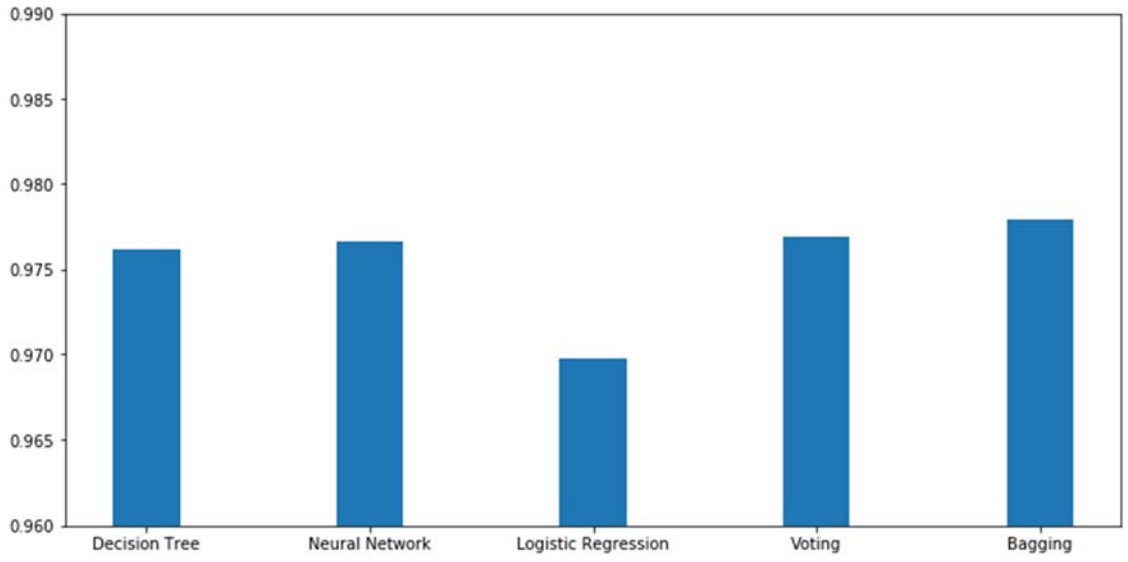


Comparison for all models

- ROC for all the models



- Test Accuracy for all the models



3.1) RESULTS AND FINDINGS

From classification by decision tree, we can have some insights from the tree as generated above, from the feature importance:

- Increase in the number of Tower and inhibitor kills will improve chance of winning substantially
- Next up is from killing Neutral Boss Monsters, especially Baron Kills.

Based on the overall results as shown, the test data accuracy for all models is around 97%, which is quite high. These is probably due to the cleanliness of the data that was downloaded straight from Kaggle, and enough data is provided (50,000 ++ Rows). Also, the data might be easily separable.

From all the grid searches that was done to improve the models, hyperparameter does play a big role in getting a better model.

For this dataset, all the methods seem to work almost equally well except for Logistic Regression. This is probably due to the data being non-linear, which logistic regression cannot handle as well.

Because the accuracy is near the 100% mark, there seems to be a problem as the accuracy will then be saturated so we might not be able to see much difference across different models in terms of accuracy.

To showcase the ability each type of models can do, it is better to pick a dataset of a lower quality in the future.

3.2) CONCLUSION

Classification problems can be solved by many ways, which includes deep learning and machine learning. In this assignment, we can see that these learning methods can be further improved by tuning the hyperparameters and certain techniques. To improve further, ensembles serve to help improve accuracy and reduce variance.