Dota 2 Outcome Prediction

Anmol Agarwal
anmola2

Omkar Mehta
omehta2

Yinfei Zhu
yinfeiz2

I. Introduction

Dota 2 is an online video game in which two teams of five players compete to destroy the 'Ancient', a large structure defended by the opposing team, while defending their own. The two teams are known as Radiant and Dire. The Ancients sit in bases in opposite corners of the map that are connected by three paths called 'lanes'. The lanes are guarded by 'towers' that attack enemies within its range. Each team has three towers in each lane and one of them is located in the base to protect two buildings called 'barracks'. In front of the Ancient are two towers that must be destroyed before the Ancient. Each player picks a unique hero so that the game starts with ten different heroes.
The end goal of this project is to predict the chance of winning at a certain time of a game. We start this off by first creating a model that predicts the winner at the end of each game. This helped us find the important features that determined the victor of a match and also told us which models worked best with the dataset. We then constructed a new dataset that included data from different time intervals in the games and created models that predicted the winner at any time interval in the game.

II. Related Work

The dataset we are going to use has been part of a Kaggle competition so some related work on this dataset already exists.
'DOTA 2 - Predicting wins using hero and item data' (https://www.kaggle.com/ianchute/dota-2-predicting-wins-using-hero-and-item-data) is work that predicts the chances of winning. This dataset looks at the hero composition of teams and the items used during the match to predict which team won using a simple decision tree. They concluded that "the heroes and items are indeed predictive of victory or defeat in DOTA 2" with a cross validation score of 83%. They also were able to produce a list of important items that indicated victory and found that "The only hero mentioned in the upper portions of the decision tree is the Necrophos - one of the most effective mass healers in the game" suggesting that hero lineup is not a strong indicator of victory (other than Necrophos).
One other relevant work is 'Relevance of gold lead for win prediction' (https://www.kaggle.com/styggy/relevance-of-gold-lead-for-win-prediction). This work looks at the chances of a team winning at any point of match based on whether the team is currently holding the gold lead. This work has shown that gold lead is a good predictor of success when the game is nearing completion with "83% of teams that lead at minute 45 end up winning the game." and 97% of teams having the gold lead at the end of the match.

Identify applicable funding agency here. If none, delete this.

III. Methods

A. Basic statisical analysis

We did statistical analysis on:

- Total number of radiant wins and dire wins.
- The frequency of heroes.
- The probability distribution of the following features: gold_spent, kills, assists, hero_healing, xp_roshan, gold_killing_creeps, gold_per_min.
- The Pearson's correlation coefficient between gold_per_min and kills.

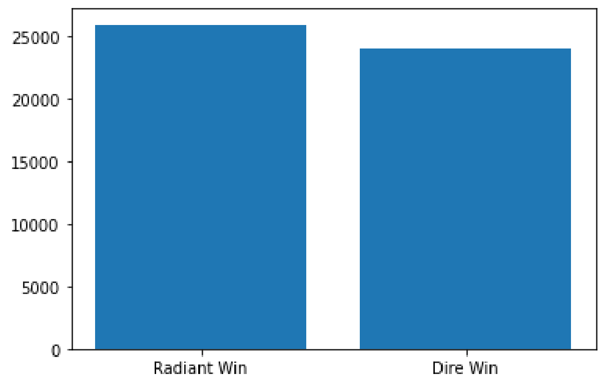


Fig. 1. Number of radiant wins and Dire wins

Most of the results fell in line with our expectations. There was not a significant difference between radiant wins and dire wins (the two possible teams you can play) [Figure 1]. Windranger was the most popular hero and Shadow Fiend was the second most popular hero that the players chose [Figure 2]. We will still include the heros as features even though they most likely do not play a significant role on which team will win based on previous work.

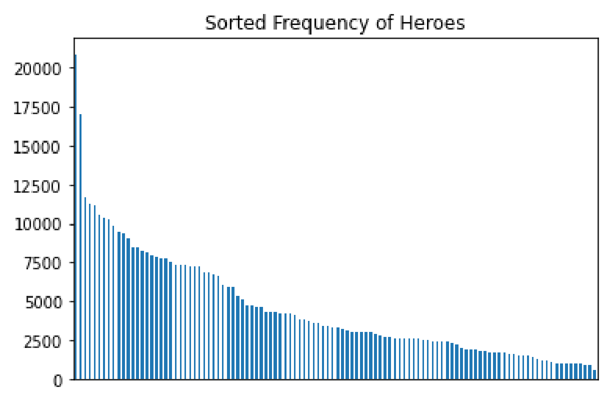


Fig. 2. Frequency of Heroes

Most of the probability distributions followed the Weibull distribution or Gaussian Distribution which was expected. One interesting distribution was the xp_roshan graph. This had a huge peak at 0 xp and some smaller peaks at 400 xp and 800 xp. We believe this is due to the fact that one team usually dominates the Roshans during the game, therefore at least half of the players in the game get 0 xp from the Roshans [Figure 3]. The 400 xp and 800 xp are smaller peaks due to how getting xp from Roshan functions. A more in depth analysis of the data can be seen in our ipynb file.

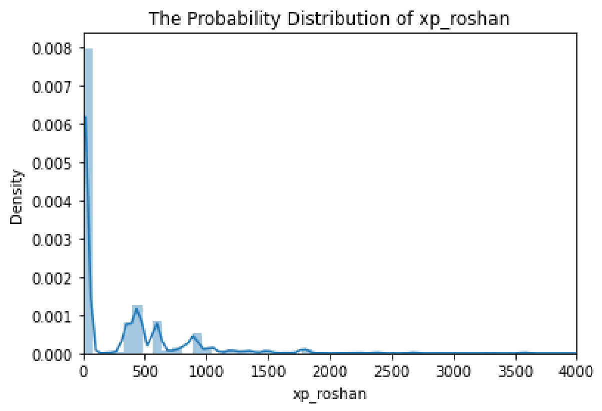


Fig. 3. Probability distribution of xp_roshan

B. Probabilistic Analysis of the Data

In this section we did a KS test to determine whether or not the distribution of gold_per_min for the radiant team and the distribution of gold_per_min for the dire team was the same. The null hypothesis is that the distribution of gold_per_min for radiant team and distribution of gold_per_min for dire team are the same. We failed to reject the null hypothesis through the KS test by getting a p-value close to zero. This indicates that neither team has any discernible advantage in gold over the course of the game.

C. Feature Engineering

Feature engineering has been a big portion of the project so far. First we performed two merge operations combining hero_names and players on hero_id, we call this Dota, and then merging Dota with match on the match_id feature. Individual players are identified by account_id but there is an option to play anonymously and roughly one third of the account_id are not available. Anonymous users have the value of 0 for account_id. It contains totals for kills, deaths, denies, etc. Player action counts are available, and are indicated by variable names beginning with unit_order. It counts for reasons for acquiring or losing gold, and gaining experience, have prefixes gold, and xp. Matches contain top level information about each match. Heroes dataset has hero lookup items: item lookup.

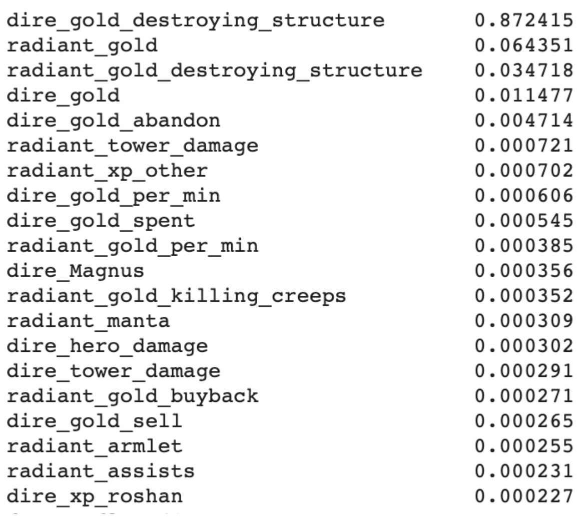
Then, we encoded whether or not a type of hero is present for that team (boolean) and compute the sum of the combined item counts for every team. That is, we summed each players' hero and item counts so that it will result in one row per match. The result is a row that encodes whether or not (1 or 0) a type of hero is present for that team and also the sum of the combined item counts for every team - (getting the item per player will be more memory intensive and hence, we are combining per team).

Then we encoded other features such as gold_per_min, xp_per_min, kills, deaths, assists, denies, last_hits, hero_damage, hero_healing, tower_damage, xp_hero, xp_creep, xp_roshan, gold_death, gold_destroying_structure, gold_killing_heros, etc. These are features that we believe are effective in determining which team is doing better based on both prior game knowledge and previous work.

D. Classifiers

After this, we made classifiers that determine which team has won based on the features we have included. This could help us identify what each team and team member should be prioritizing on in order to have the highest chance of winning. We have decided to do a Naive Bayes classifier (baseline model), a Decision Tree classifier, a SVM classifier, etc.. We then compared how well they performed against each other. The training and testing data was randomized for all of the classifiers with an 80% training, 20% testing split.
The results for the classifiers were:

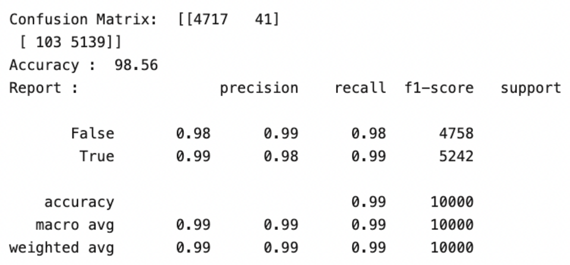
- Naive Bayes: Accuracy = 98.56
- SVM: Accuracy = 98.43
- Decision Tree: Accuracy = 98.92



```
dire_gold_destroying_structure        0.872415
radiant_gold                          0.064351
radiant_gold_destroying_structure     0.034718
dire_gold                             0.011477
dire_gold_abandon                     0.004714
radiant_tower_damage                  0.000721
radiant_xp_other                      0.000702
dire_gold_per_min                     0.000606
dire_gold_spent                       0.000545
radiant_gold_per_min                  0.000385
dire_Magnus                           0.000356
radiant_gold_killing_creeps           0.000352
radiant_manta                         0.000309
dire_hero_damage                      0.000302
dire_tower_damage                     0.000291
radiant_gold_buyback                  0.000271
dire_gold_sell                        0.000265
radiant_armlet                        0.000255
radiant_assists                       0.000231
dire_xp_roshan                        0.000227
```

Fig. 4. Important Features as per Decision Tree

As we can see from Figure 4, dire_gold_destroying_structure is the most important feature. As we had proposed, gold would be the most important feature in deciding the win of any team. Our model proves this proposal.
We have used a confusion matrix and classification_report from sklearn library.

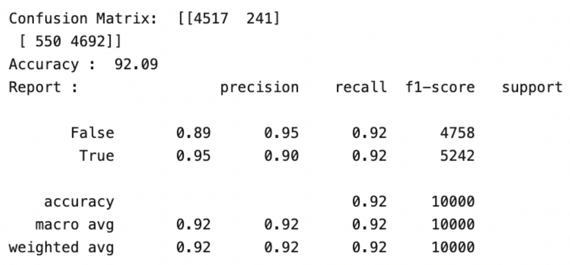


```
Confusion Matrix:  [[4717   41]
 [ 103 5139]]
Accuracy :  98.56
Report :               precision    recall  f1-score   support

       False       0.98      0.99      0.98      4758
        True       0.99      0.98      0.99      5242

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Fig. 5. Confusion matrix for Gaussian NB

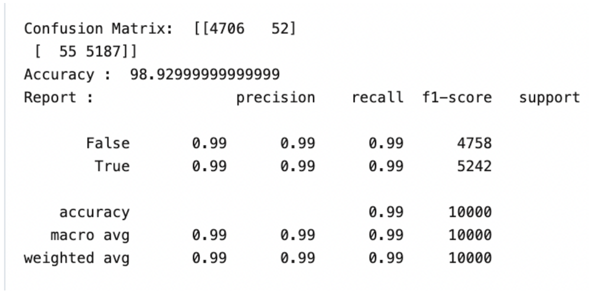


```
Confusion Matrix:  [[4517  241]
 [ 550 4692]]
Accuracy :  92.09
Report :               precision    recall  f1-score   support

       False       0.89      0.95      0.92      4758
        True       0.95      0.90      0.92      5242

    accuracy                           0.92     10000
   macro avg       0.92      0.92      0.92     10000
weighted avg       0.92      0.92      0.92     10000
```

Fig. 6. Confusion matrix for Bernoulli NB



```
Confusion Matrix:  [[4706   52]
 [  55 5187]]
Accuracy :  98.92999999999999
Report :               precision    recall  f1-score   support

       False       0.99      0.99      0.99      4758
        True       0.99      0.99      0.99      5242

    accuracy                           0.99     10000
   macro avg       0.99      0.99      0.99     10000
weighted avg       0.99      0.99      0.99     10000
```

Fig. 7. Confusion matrix for Decision Tree

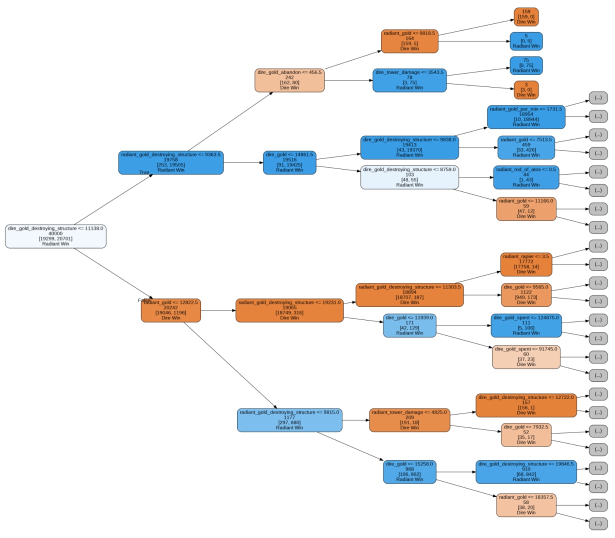

dire_gold_destroying_structure <= 11138.0 40000 [19299, 20701] Radiant Win
True
False
radiant_gold_destroying_structure <= 9363.5 19758 [253, 19505] Radiant Win
dire_gold_abandon <= 456.5 242 [162, 80] Dire Win
radiant_gold <= 9818.5 164 [159, 5] Dire Win
159 [159, 0] Dire Win
5 [0, 5] Radiant Win
dire_tower_damage <= 3543.5 78 [3, 75] Radiant Win
75 [0, 75] Radiant Win
3 [3, 0] Dire Win
dire_gold <= 14861.5 19516 [91, 19425] Radiant Win
dire_gold_destroying_structure <= 9838.0 19413 [43, 19370] Radiant Win
radiant_gold_per_min <= 1731.5 18954 [10, 18944] Radiant Win
radiant_gold <= 7513.5 459 [33, 426] Radiant Win
dire_gold_destroying_structure <= 8759.0 103 [48, 55] Radiant Win
radiant_rod_of_atos <= 0.5 44 [1, 43] Radiant Win
radiant_gold <= 11166.0 59 [47, 12] Dire Win
radiant_gold <= 12822.5 20242 [19046, 1196] Dire Win
radiant_gold_destroying_structure <= 19231.0 19065 [18749, 316] Dire Win
radiant_gold_destroying_structure <= 11303.5 18894 [18707, 187] Dire Win
radiant_rapier <= 3.5 17772 [17758, 14] Dire Win
dire_gold <= 9565.0 1122 [949, 173] Dire Win
dire_gold <= 11939.0 171 [42, 129] Radiant Win
dire_gold_spent <= 124875.0 111 [5, 106] Radiant Win
dire_gold_spent <= 91745.0 60 [37, 23] Dire Win
radiant_gold_destroying_structure <= 9815.0 1177 [297, 880] Radiant Win
radiant_tower_damage <= 4825.0 209 [191, 18] Dire Win
dire_gold_destroying_structure <= 12722.0 157 [156, 1] Dire Win
dire_gold <= 7932.5 52 [35, 17] Dire Win
dire_gold <= 15258.0 968 [106, 862] Radiant Win
dire_gold_destroying_structure <= 19846.5 910 [68, 842] Radiant Win
radiant_gold <= 18357.5 58 [38, 20] Dire Win
(...)

Fig. 8. Visualization for Decision Tree

This is the decision tree, which clearly explains the split-points using the most important features. The most important features are computed using gini-index.
We tried Random Forest Classifier, Support Vector Machines, Deep Neural Networks and XGBoost for prediction of the outcome of the match.
Random forest consists of a large number of individual decision trees that operate as an ensemble [6]. We have chosen the criterion to be 'entropy' that chooses the features in the order of reduction in uncertainty. The number of estimators are 10. We got the following confusion matrix: [class_0: dire_wins, class_1: radiant_wins.]

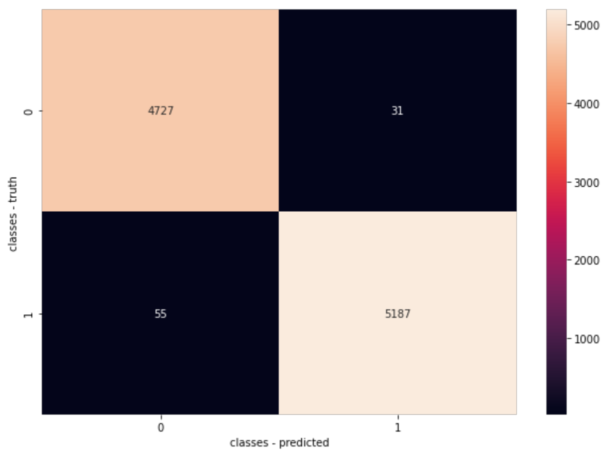


Fig. 9. Confusion matrix for Random Forest

Model Name	Accuracy	Precision	Recall	f1-score
Random Forest Classifier	99.14	0.99	0.99	0.99

Fig. 10. Model Evaluation for Random Forest

According to the random forest, the top 5 important features are as shown in figure 11

Feature Name	Feature Importance
dire_gold_destroying_structure	0.2627
radiant_gold_destroying_structure	0.1074
radiant_gold_per_min	0.1063
dire_gold_per_min	0.0825
dire_gold	0.0793

Fig. 11. Feature Importance for top five features

The figure 11 makes sense, since the destruction of the structures is the most important aspect to winning the game. If we use max_depth of 1, then we get the following visualization of the decision process of the random forest:

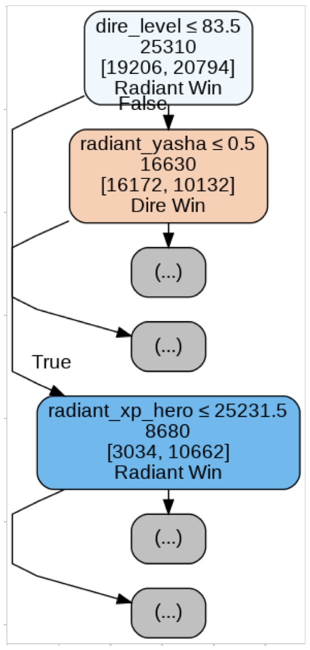


Fig. 12. Visualization of Random Forest

Support Vector Machines are supervised learning methods that can be used for a variety of purposes [7]. In our project, SVM is used for classification between radiant victory and dire victory. The initial attempt in training the SVM using all the features was very inefficient. Therefore, we decided to perform principal component analysis to reduce the number of features before training the SVM.

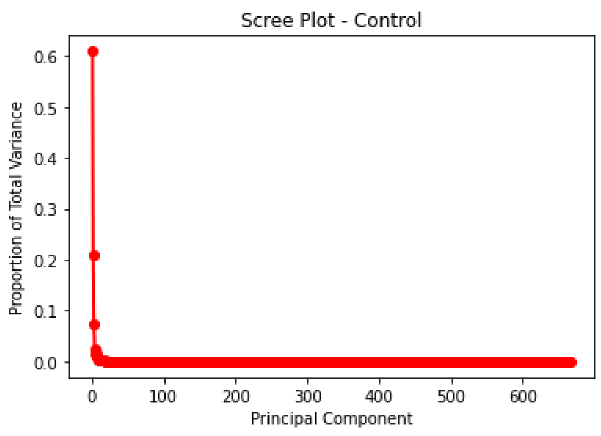
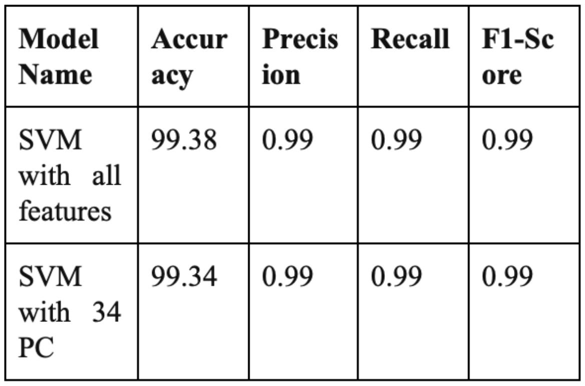


Fig. 13. Scree Plot for PCA

By PCA [Figure 13], we found that 34 principal components are able to explain 99.99% of the total variance. Training the SVM using these 34 principal components was much faster than the initial attempt and the accuracy was only slightly lower.



Model Name	Accuracy	Precision	Recall	F1-Score
SVM with all features	99.38	0.99	0.99	0.99
SVM with 34 PC	99.34	0.99	0.99	0.99

Fig. 14. Model Evaluation for SVM

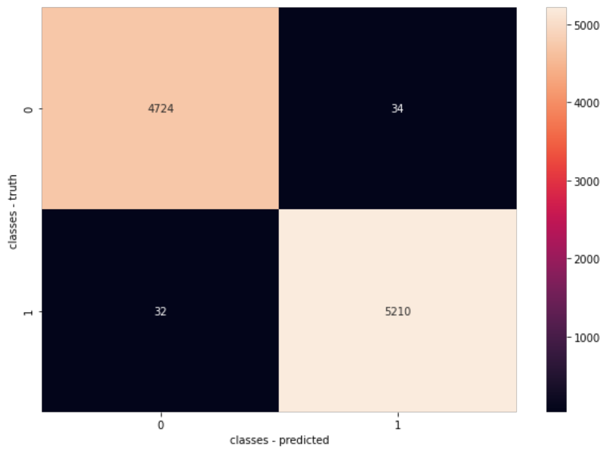


Fig. 15. Confusion Matrix for SVM

We have used Keras for training deep neural networks, since Keras is the most used deep learning framework. Keras can scale to large clusters of GPUs. For training purposes of the huge dataset that we have, we will be using a TPU pod. Our dataset contains numerical and categorical features. Keras library can handle both types of features. We will split the dataset into training and validation dataset.
We have used MinMaxScaler to scale the entire dataset, since neural networks are efficient when the dataset is scaled.
We have created customized functions for making the model, training the model, plotting the error, visualizing the connectivity graph.
Model Architecture:

- Input Layer with 'normal' kernel initializer, 'relu' activation function and dropout regularizer.
- Four Hidden Layers with 'normal' kernel initializer, 'relu' activation function and dropout regularizer.
- Output Layer with 'sigmoid' activation function.

The model is compiled with loss as 'binary_crossentropy', metric as ['accuracy'] and 'adam' optimizer [9].
We trained the model on a training dataset with validation split of 0.1, epochs of 100, batch size of 16 (for faster training).
In order to avoid overfitting, we have used dropouts (the probability of dropping some neurons in all the layers). We have also used the EarlyStopping feature of the model to stop the training when the validation loss increases after a steep drop in the loss.

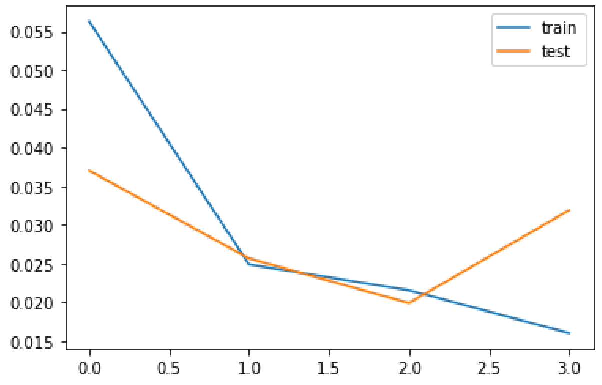
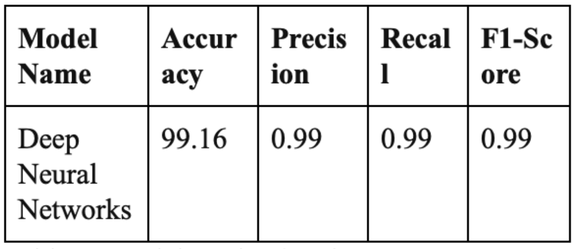


Fig. 16. Model Loss for Neural Networks

As we can see from [Figure 16] that due to Early Stopping, the training stopped after 3 steps. Here, 'test' refers to the validation set.



Model Name	Accuracy	Precision	Recall	F1-Score
Deep Neural Networks	99.16	0.99	0.99	0.99

Fig. 17. Model Evaluation for Deep NN

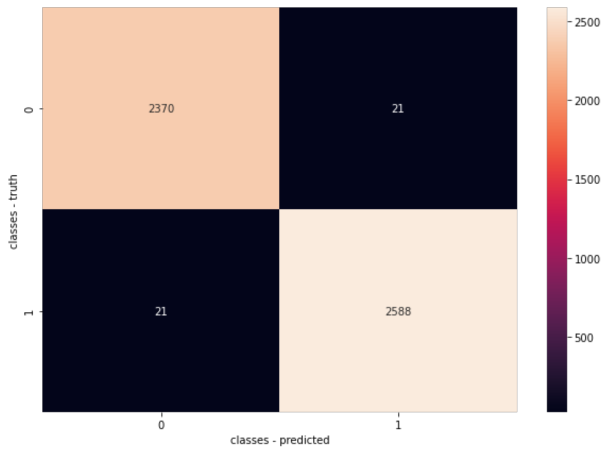


Fig. 18. Confusion Matrix for Deep Neural Networks

After this we used the player_time dataset to incorporate our work so far with time intervals so that we can predict the winner of the match at any point during the game. Based on the previous data we decided to train this dataset using Naive Bayes, Random Forest and XGBoost. The player_time dataset uses the features including the gold, xp and last hits of each player at each 60 second time interval for every match. As we found in previous work, gold is the most important factor and xp is the second, while last hits is closely related to gold especially in early game. For each time interval we trained the model with each method and have plotted the accuracy of predicting the match winner for every minute of the match.

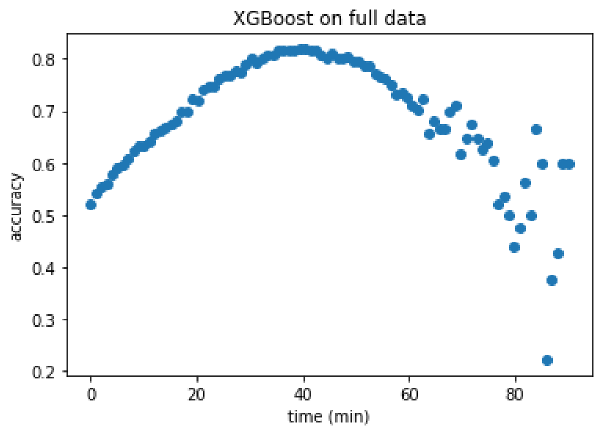


Fig. 21. XGBoost Accuracy of predicting match winner vs time (minutes)

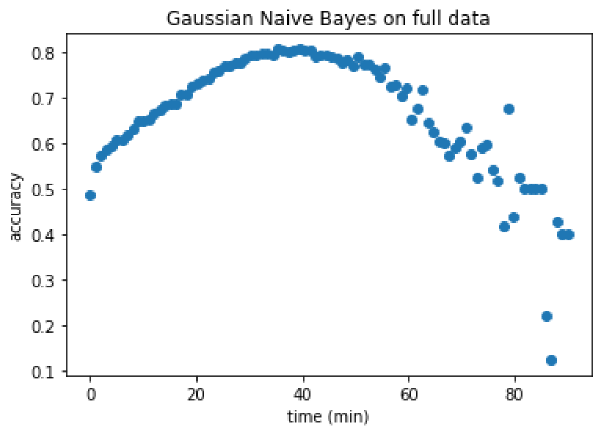


Fig. 19. GNB Accuracy of predicting match winner vs time (minutes)

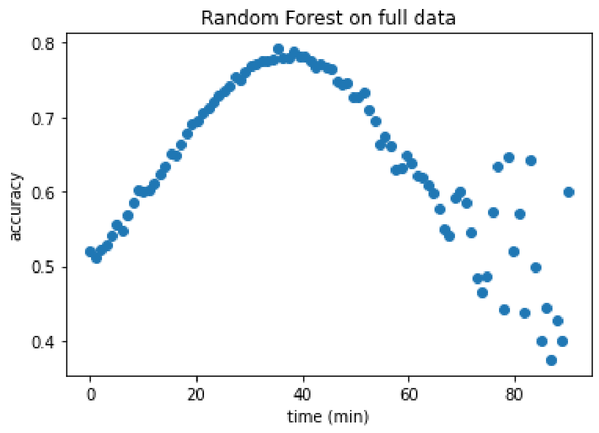


Fig. 20. Random Forest Accuracy of predicting match winner vs time (minutes)

Figure 19, 20 and 21 show the accuracy of the models as a function of the minute when the prediction is made. All three figures have the same pattern. The accuracy first increases until around 40 minutes, and then decreases in general with irregular behaviors. The accuracy at 0 minute is around 0.5 because there is almost no information at the beginning of a game. The increase in accuracy is easy to interpret because more information is given as the game progresses. There are several possible reasons why the accuracy decreases after 40 minutes. First, in late game, most players already have enough gold for necessary items and enough xp for necessary levels. In such case, gold and xp are no longer the leading factors of the outcome. Instead, a single critical fight may determine the winner of the game. Another possible reason is that long games happen much less often and thus do not generalize well.
To improve the accuracy, we decided to treat long games as outliers and remove them from the player_time dataset. By looking at the number of matches of each length, we found that more than 80% of the games are within 50 minutes. So we keep only the matches within 50 minutes and train the dataset using the same models.

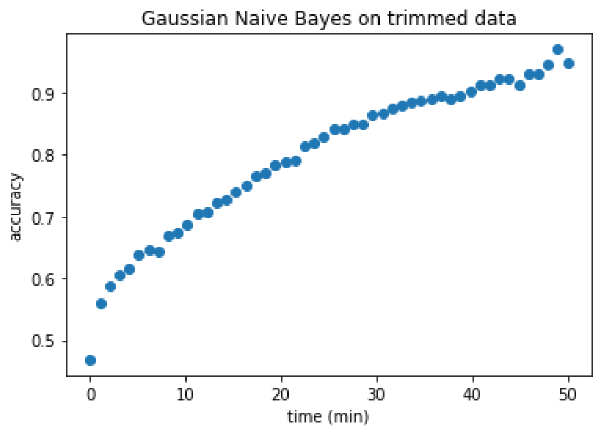


Fig. 22. GNB Accuracy of predicting match winner vs time (minutes) with trimmed data

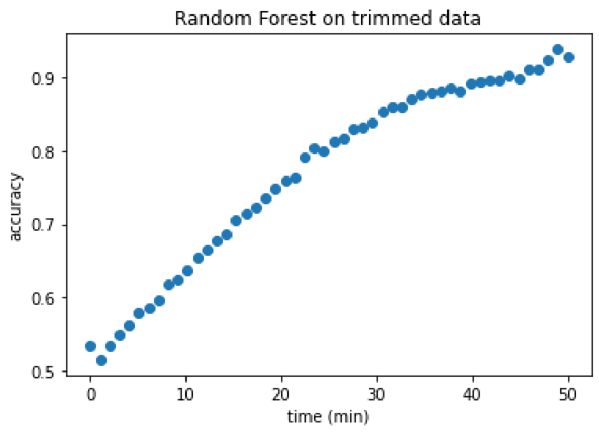


Fig. 23. Random Forest Accuracy of predicting match winner vs time (minutes) with trimmed data

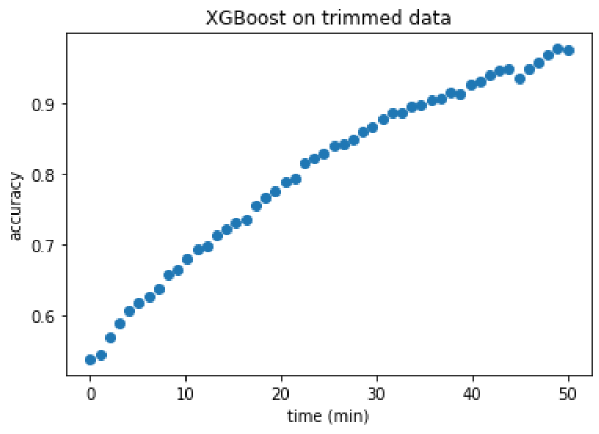


Fig. 24. XGBoost Accuracy of predicting match winner vs time (minutes) with trimmed data

Figure 22, 23 and 24 show the accuracy of the models with trimmed data. Again, all three figures have the same pattern. This time the accuracy increases as the game progresses even after 40 minutes, and the accuracy at each time point is higher than that of the models with full data.

IV. Challenges

The main challenge was to engineer novel features based on the experience of the game. Once we knew which features were important, we trained our models for predicting the outcome of the game at any instant. This required manipulation of the dataset by creating a new function which returns the tuples of the time-dependent features and labels.

V. Conclusions

The main purpose of this project was to predict the outcome of the game, as the game progresses. There are multiple datasets, which we needed to combine, based on the hero_id, match_id, player_id. We used the combined dataset to get the most important features, using Decision Trees and Random Forests. We concluded that the gold_destroying_structure was the most important feature, and all other features containing gold are also important. We used different classifiers and reported the model evaluation metrics for the models like Gaussian Naive Bayes, Bernouilli Naive Bayes, Decision Trees, Random Forests, Support Vector Machines, Deep Neural Networks, XGBoost. While this gave us an idea of how the models were performing on the classification task when we knew the whole story, i.e., data regarding all the matches till their end was known. We wanted to classify how the model performs at a particular instant based on the information available to the model until that instant. We used player_times dataset present on the Kaggle website for including the time component. We got a very fair accuracy for the models. No one has been able to predict the outcome of the match at any instant on Kaggle website or anywhere.

References

[1] "Dota 2 Matches " retrieved by Devin Anzelmo https://www.kaggle.com/devinanzelmo/dota-2-matches

[2] Data description https://wiki.teamfortress.com/wiki/WebAPI/GetMatchDetailsTower$_{S}tatus$

[3] Dota Game https://dota2.gamepedia.com/Dota$_{2}Map$

[4] 'How to evaluate your machine learning models with python code' by Terence Shin from https://towardsdatascience.com/how-to-evaluate-your-machine-learning-models-with-python-code-5f8d2d8d945b

[5] "Evaluating Classification Models" by Ishaan Dey from https://towardsdatascience.com/hackcvilleds-4636c6c1ba53

[6] 'Classification of Data with Random Forests and Decision Trees' by Shashank from https://github.com/srp98/Classification-of-Data-with-Random-Forests-and-Decision-Trees/blob/master/RandomForests.py

[7] "HyperParameter Tuning by SVM MNIST" from https://github.com/Madmanius/HyperParameter$_{t}uning_{S}VM_{M}NIST/blob/master/I$

[8] "Structured Data Classification from Scratch" from https://keras.io/examples/structured$_{d}ata/structured_{d}ata_{c}lassification_{f}rom_{s}crato$

[9] "Binary Classification Tutorial with the keras deep learning library" from https://machinelearningmastery.com/binary-cla