**DWIT COLLEGE**

**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

**Institute of Science and Technology**

****

**LINE UP PREDICTION SYSTEM FOR A MOBA GAME USING KNN ALGORITHM**

**A PROJECT REPORT**

**Submitted to**

**Department of Computer Science and Information Technology**

**DWIT College**

***In partial fulfillment of the requirements for the Bachelor’s Degree in Computer Science and Information Technology***

Submitted by

Aabhusan Gautam, Ashish Khanal

July, 2017

**DWIT College**

**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

**SUPERVISOR’S RECOMENDATION**

I hereby recommend that this project prepared under my supervision by AABHUSAN GAUTAM and ASHISH KHANAL entitled **“LINE UP PREDICTION SYSTEM FOR A MOBA GAME USING KNN ALGORITHM”** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology be processed for the evaluation.

…………………………………………

Birodh Rijal

Lecturer

DWIT College

**DWIT College**

**DEERWALK INSTITUTE OF TECHNOLOGY**

**Tribhuvan University**

# LETTER OF APPROVAL

This is to certify that this project prepared by AABHUSAN GAUTAM AND ASHISH KHANAL entitled **“LINE UP PREDICTION SYSTEM FOR A MOBA GAME USING KNN ALGORITHM”** in partial fulfillment of the requirements for the degree of B.Sc. in Computer Science and Information Technology has been well studied. In our opinion it is satisfactory in the scope and quality as a project for the required degree.

|  |  |
| --- | --- |
| ……………………………………  Birodh Rijal [Supervisor]  Lecturer  DWIT College | …………………………………………  Hitesh Karki  Chief Academic Officer  DWIT College |
| …………………………………………..  Jagdish Bhatta [External Examiner]  IOST, Tribhuvan University | ………………………………………….. |

# ACKNOWLEDGEMENT

We would like to thank Mr. Birodh Rijal, Lecturer, DWIT College, for supervising this project and giving assistance when faced with problems. Without his guidance and persistent help, this project would not have been possible.

Also, we would like to thank everyone who supported us in a direct or indirect manner that helped us to complete this project.

Aabhusan Gautam, Ashish Khanal

TU Exam Roll no: 3012/070, 3017/070

# STUDENT’S DECLARATION

I hereby declare that I am the only author of this work and that no sources other than the listed here have been used in this work.

... ... ... ... ... ... ... ...

Aabhusan Gautam, Ashish Khanal

Date: July, 2017

# ABSTRACT

This paper presents a line-up prediction system for a MOBA game (DOTA 2), using KNN algorithm. This system implements K Nearest Neighbors algorithm to find out the nearest neighbor for any given query data. A polynomial weight function is used to more elaborately find nearest neighbor for any given query.

This paper also provides insights to the process of data collection and feature selection before the implementation of the algorithm.

Keywords**:** K nearest neighbor classifier, Supervised Machine Learning, Steam Web API, DOTA 2, Heroes.

**TABLE OF CONTENTS**

[LETTER OF APPROVAL i](#_Toc489486404)

[ACKNOWLEDGEMENT ii](#_Toc489486405)

[STUDENT’S DECLARATION iii](#_Toc489486406)

[ABSTRACT iv](#_Toc489486407)

[LIST OF FIGURES vii](#_Toc489486408)

[LIST OF TABLES viii](#_Toc489486409)

[LIST OF ABBREVIATIONS ix](#_Toc489486410)

[CHAPTER 1: INTRODUCTION 10](#_Toc489486411)

[1.1 Background 10](#_Toc489486412)

[1.2 Problem Statement 10](#_Toc489486413)

[1.3 Objectives 11](#_Toc489486414)

[1.3.1 General objective: 11](#_Toc489486415)

[1.3.2 Specific objective: 11](#_Toc489486416)

[1.4 Scope 11](#_Toc489486417)

[1.5 Limitation 11](#_Toc489486418)

[1.6 Outline of Document 13](#_Toc489486419)

[CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS 14](#_Toc489486420)

[2.1 Literature Review 14](#_Toc489486421)

[2.2 Requirement Analysis 17](#_Toc489486422)

[Functional requirement 17](#_Toc489486423)

[Non-functional requirement 17](#_Toc489486424)

[2.3 Feasibility Analysis 18](#_Toc489486425)

[2.3.1 Technical feasibility 18](#_Toc489486426)

[2.3.2 Operational feasibility 18](#_Toc489486427)

[2.3.3 Schedule feasibility 18](#_Toc489486428)

[CHAPTER 3: SYSTEM DESIGN 19](#_Toc489486429)

[3.1 Methodology 19](#_Toc489486430)

[3.1.1 Data collection 19](#_Toc489486431)

[3.1.2 Feature Vector Construction 20](#_Toc489486432)

[3.1.3 Making Prediction 20](#_Toc489486433)

[3.1.3 Logic Implementation 21](#_Toc489486434)

[3.2 Algorithm 22](#_Toc489486435)

[3.2.1 K-Nearest Neighbor (KNN) 22](#_Toc489486436)

[3.3 System Design 24](#_Toc489486437)

[3.3.1 Class diagram 24](#_Toc489486438)

[3.3.2 Sequence diagram 25](#_Toc489486439)

[3.3.3 State diagram 25](#_Toc489486440)

[CHAPTER 4: IMPLEMENATION AND TESTING 27](#_Toc489486441)

[4.1 Implementation 27](#_Toc489486442)

[4.1.1 Tools used 27](#_Toc489486443)

[4.2 Description of Major Classes 27](#_Toc489486444)

[4.2.1 Preprocess 28](#_Toc489486445)

[4.2.2 test 28](#_Toc489486446)

[4.2.3 K\_nearest\_neighbors 28](#_Toc489486447)

[4.3 Testing 29](#_Toc489486448)

[4.3.1 Unit Testing 29](#_Toc489486449)

[4.3.2 Integration Testing 31](#_Toc489486450)

[CHAPTER 5: MAINTENANCE AND SUPPORT PLAN 32](#_Toc489486451)

[5.1 Maintenance Plan 32](#_Toc489486452)

[5.2 Support Plan 32](#_Toc489486453)

[CHAPTER 6: CONCLUSION AND RECOMMENDATION 33](#_Toc489486454)

[6.1 Conclusion 33](#_Toc489486455)

[6.2 Recommendation 34](#_Toc489486456)

[APPENDIX 35](#_Toc489486457)

[REFERENCES 36](#_Toc489486458)

# 

# LIST OF FIGURES

[Figure 1- Outline of document 12](#_Toc489146136)

[Figure 2 – Logistic Regression Learning Curve 14](#_Toc489146137)

[Figure 3 –Augmented Logistic regression learning curve](#_Toc489146138) 16

[Figure 4 –KNN Learning Curve](#_Toc489146139) 17

[Figure 5 – Class Diagram](#_Toc489146140) 24

[Figure 6 – Sequence Diagram](#_Toc489146141) 25

[Figure 7 – State Diagram](#_Toc489146142) 26

# 

# LIST OF TABLES

[Table 1- Functional and non-functional requirements 16](#_Toc489146166)

[Table 2 - Unit Testing of different Components](#_Toc489146168) 29

[Table 3 - Integration testing of the combination of different modules](#_Toc489146169) 31

# LIST OF ABBREVIATIONS

KNN: K-nearest neighbor

DOTA: Defense Of The Ancients

LR: Logistic Regression

# CHAPTER 1: INTRODUCTION

## Background

DOTA 2 (Defense Of The Ancients) is a 5v5 online, multiplayer, arena based game which originally started off as a customized version of Warcraft III. DOTA started out in 2003 and had a steady growth in players along with addition of features and interesting mechanisms to the game. DOTA 2 currently attracts more than 7 million unique players per month and the prize of competitive tournaments can amount to millions of dollars. The two opposing sides are known as “radiant” and the “dire”. Each side has a core structure called “ancient”. The players battle with each other until they destroy one of the ancient and win the match. The players are allowed to pick any one of the 106 heroes that are available in the game. “Heroes” are the characters in-game that have different working mechanisms and features. Each player picks a hero and battles with the enemy team alongside with their team.

Players not only have to consider the hero they’re playing with but also the synergistic relationship of that hero with other heroes in the team. Each hero has its own strengths and weaknesses. Each hero has these three primary attributes, “strength”, “agility” and “intelligence” and based on the amount of these attributes possessed by the hero, its role is defined in the game. Heroes also have other attributes like being “ranged” or “melee” or based on their base attribute they are usually classified into groups like carry (heroes that are weak during the early phase of the game but become stronger as the game progresses and carry the team to victory in the end) and support (heroes that are strong during the early phase of the game and help carries grow and ultimately lead the team to victory).

A player has to choose a hero to play with in the beginning of the match and hence, it can be considered as one of the most important phases of the game because players have to choose heroes that can work with other heroes present in the team effectively to have a high chance of winning that match.

## 1.2 Problem Statement

As the drafting stage (selection of particular heroes by the players) is one of the most important phases of the game. Some games can be lost or won during this stage given all the players have same skill levels. Players tend to pick heroes based on their individual skills and doing so, they do not consider the synergistic relationship of their hero with other heroes in the team. While they do need to consider their individual skills while picking a hero, a hero that doesn’t function well with the rest of the team can lead to defeat no matter how good the player is. Even the antagonistic relationship between heroes has to be considered during the drafting phase as some heroes counter their opposing heroes really well.

With the tremendous amount of increase in number of players, steady increase in heroes count and features, it calls for a system that will help players to choose their heroes based on the rest of the picks and the enemy picks. While good team interaction and experience helps to solve this problem to some degree, a system dedicated to counter this problem based on actual results would be equally reliable.

## 1.3 Objectives

### 1.3.1 General objective:

To implement KNN algorithm to build a system that recommends heroes based on the provided query along with winning rate for that particular configuration.

### 1.3.2 Specific objective:

a) To find out what character is most viable given a certain configuration of heroes in “radiant” and “dire” side.

## 1.4 Scope

DOTA 2 recommendation engine can be used by normal players to help them pick heroes efficiently and coaches and professional players to elaborately learn about impact of picking certain heroes in certain configurations.

## 1.5 Limitation

a) Memory usage is not optimized as additional information of each dataset remains there.

b) The system does not consider the change in gameplay mechanism which is updated frequently.

c) Static data set is used which causes the data relevancy to decrease with the updates and passage of time.

d) The optimal polynomial scaling factor was calculated using hit and trial method and varied only across few numbers due to timing constraint.

## 

## 1.6 Outline of Document

The report is organized as follows

* Title Page
* Abstract
* Table of Contents
* List of figures and Tables

Preliminary

Section

Introduction

Section

* Background
* Problem Statement
* Objectives
* Scope
* Limitation

Requirement and Feasibility Analysis Section

* Literature Review
* Requirement Analysis
* Feasibility Analysis
* Methodology
* Algorithm
* System Design

System Design

Section

Implementation and Testing Section

* Implementation
* Description of Major Classes
* Testing

Maintenance and Support Plan Section

* Maintenance Plan
* Support Plan

Conclusion and Recommendation Section

* Conclusion
* Recommendation

Figure 1- Outline of document

# 

# CHAPTER 2: REQUIREMENT AND FEASIBILITY ANALYSIS

## 2.1 Literature Review

Dota2, due to its popularity and consistent growth, has drawn fair amount of attention in machine learning projects.

A paper presented by Kevin Conley and Daniel Perry in 2013[1] discussed different approaches to increase the accuracy of recommendation systems. Both KNN and Logistic regression were compared and their benefits and drawbacks were presented.

Logistic regression is a model that predicts binary output using a weighted sum of predictor variables. A simple logistic regression model was trained with an intercept term to predict the outcome of the match in the paper.

The testing accuracy of the model asymptotically approached 69.8% at about an 18,000 training set size indicating that it was the optimal training set size for logistic regression model.

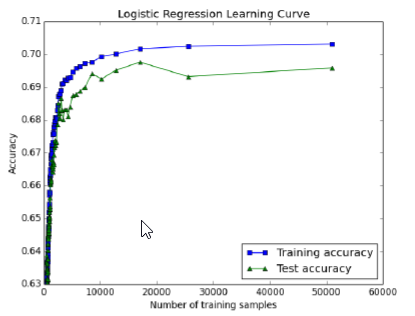


Figure 2: Logistic Regression learning curve

Logistic regression model shows that hero selection alone is an important indicator of the outcome of a DOTA 2 match. However, single logistic regression is purely weighted sum of constructed feature vector (which only indicates which heroes are on either team), it fails to capture other essentials things like synergistic and antagonistic relationship between heroes.

However, a paper presented by Kaushik Kalyanaraman in 2014[2] gave more insight and explored more algorithms that could be used to predict the outcome of a DOTA 2 match. In this paper, an augmentation to existing machine learning algorithms was proposed and tested.

Pure logistic regression was used initially to test the accuracy of the model which did not change much from what was previously calculated. However, adding a new measure of success (of winning a dota2 match) along with the implementation of genetic algorithm and community detection process improved both the training and testing accuracy.

As discussed before, purely weighted sum of feature vectors fails to capture synergistic relationships and the addition of the “success set” which was used to calculate probability for a team to win a match captured such relations between heroes.

Success measure was defined as:

Success Probability = |heroes picked| n |success heroes|

|heroes picked|

And the Final Success Probability = regression probability + success probability

2

In doing so, the test accuracy asymptotically approached the value of 74.1% and even the training accuracy increased to be more than the testing accuracy in this model.

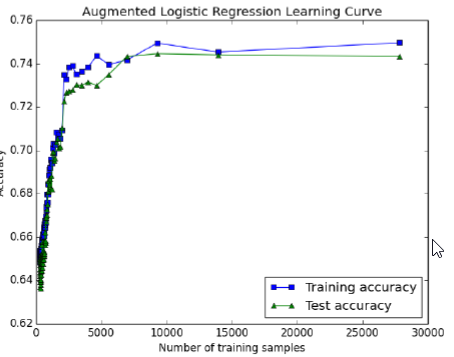


Figure 3: Augmented Logistic Regression learning curve

Even though augmented logistic regression model gave more accuracy overall, the detection of community along with implementation of genetic algorithm and success prediction system was found to be complex to implement.

A simpler algorithm, KNN (while not having as much accuracy as Augmented Logistic Regression Model), could easily be implemented to build a recommendation system while proving to be better than the basic logistic regression model.

KNN is a non-parametric method for classification and regression that predicts objects’ class memberships based on the K-closest training examples in the feature space. Using the same feature vectors as other models and adding a custom weight function filter out dissimilar training examples, the test accuracy of the KNN model reached up to 70% for around 50,000 match datasets.

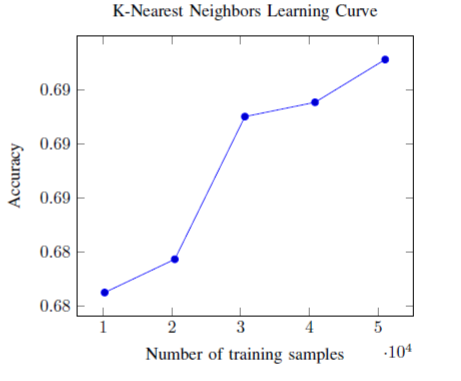


Figure 4: Learning curve of KNN model

In conclusion, KNN being simple, gives more accuracy than base logistic regression model and can be implemented easily in comparison to augmented logistic regression model which makes KNN suitable to use in building recommendation systems for DOTA 2.

## 2.2 Requirement Analysis

Table 1- Functional and non-functional requirements

|  |  |
| --- | --- |
| Functional requirement | Non-functional requirement |
| 1. Implement KNN algorithm to predict the outcome of a match with highest accuracy for a given query. 2. Recommend heroes based on the query. | 1. Display the winning chance based on the query and recommend heroes for any number of queries. |

Table 1 describes the basic functionality of line up prediction system for a MOBA game using KNN algorithm.

## 2.3 Feasibility Analysis

### 2.3.1 Technical feasibility

Line up prediction system for a MOBA game using KNN algorithm is a web application that uses Python Web framework – Flask. It uses Flask, HTML, and Bootstrap CSS to design front end and Python as the back end.

It supports both Windows and Linux platform for its operation. All of the technology required by line up prediction system for a MOBA game using KNN algorithm are available and can be accessed freely. Hence, it was determined to be technically feasible.

### 2.3.2 Operational feasibility

Line up prediction system for a MOBA game using KNN algorithm has a simple design and is easy to use. It uses two-tier architecture (i.e. Client and Server). Any user accessing the application remotely via internet can use the application to find out the winning chance of their team based on their or enemy picks. Hence, this system was determined to be operationally feasible.

### 2.3.3 Schedule feasibility

The scheduled time for the completion of the project was 90 days. The project was completed and tested within 87 days. Hence, Line up prediction system for a MOBA game using KNN algorithm was determined to be feasible according to schedule.

# CHAPTER 3: SYSTEM DESIGN

## 3.1 Methodology

Since the implementation simplicity and accuracy for KNN is high, it was considered the most suitable algorithm to implement in a DOTA 2 hero recommendation system. Scikit-learn python library was used to implement the algorithm.

### 3.1.1 Data collection

Valve’s steam web API was used to pull data for 56691 matches between June 5 and July 7. The collected data satisfies the following requirements:

1. The game mode is either all pick, single draft, all random, random draft, captain’s draft, captain’s mode, or least played. These game modes are the closest to the true vision of Dota 2, and every hero has the potential to show up in a match.
2. The skill level of the players is “very-high,” which corresponds to roughly the top 8% of players. Utilizing only very-high skill level matches allows us to best represent heroes at their full potential.
3. No players leave the match before the game is completed. Such matches do not capture how the absent players’ heroes affect the outcome of the match.

The data for each match is structured as JSON and includes which heroes were chosen for the team, how these heroes performed over the course of game and which team won the game. MongoDB was used to store the data.

90% of the data (51, 022 matches) was used as training set and the rest 10% of the data (5669 matches) was used as testing set.

### 3.1.2 Feature Vector Construction

There are 2 configurations for teams i.e. “radiant” and “dire” and 106 playable heroes in the game. The web API uses heroes ID ranging from 1 to 108 (since 2 hero ids are not used). So, the feature vector can be considered to be,

Such that:

Also another label, is defined to be:

### Making Prediction

Since the dataset contains information about heroes on teams in specific radiant and dire configurations, simply running the algorithm on each dataset does not fully utilize the data. Hence, we make predictions using the following procedure:

1. Run the algorithm on *radiant\_query* to get *radiant\_prob* (probability that the team in radiant query wins the match).
2. Construct *dire\_query* by swapping the radiant and dire teams in *radiant\_query* so that the dire team is now the top half of the feature vector.
3. Run the algorithm on *dire\_query* to get *dire\_prob* (the probability that the radiant team in *radiant\_query* loses the match if dire team was present).
4. Calculate the overall probability overall\_prob as:

*Overall\_prob* = (*rad\_prob* + (1-*dire\_prob*)) /2

1. Predict the outcome by specifying that radiant team won if *overall\_prob* > 0.5 and the dire team winning otherwise.

### 3.1.3 Logic Implementation

First of all, the data collection process is carried out. Here, steam’s web API was used to extract the publicly available data of matches during a certain period of time as explained under data collection.

After the data have been stored in the database, we preprocessed them and categorized them into train and test data.

A training matrix X was created where each row was a different match and each column was a feature. Features are the bit vectors indicating whether heroes were either picked (1) or not picked (0). The first “N” features corresponded to radiant and the last “N” features corresponded to dire.

After pre-processing was complete, we needed to create train and test models.

We created two train models: evaluation model and recommendation model. Evaluation model was used to calculate the probability of winning based on the query and recommendation model was used to recommend heroes based on the query.

For evaluation model, the pre-processed X matrix and Y vector from training model was used as input. After defining the distance between two vectors and weight for those distances, we implemented the K neighbors classifier specifying “n\_neighbors”, “metric” and “weights” and finally produced the evaluation model.

For recommendation model, along with this, we filter X matrix and Y vector for both “radiant” and “dire” configurations and appended K neighbors classifier to both of those models (radiant and dire loop). Finally, a recommendation model was generated.

For testing, we needed to calculate the accuracy of algorithm we had implemented. Hence, taking the pre-processed test data and the evaluation model as inputs. Overall probability was calculated averaging the radiant and dire probability.

Prediction was defined to be 1 if overall probability > 0.5 i.e. radiant won and -1 otherwise (dire won). The predicted result and result were compared for all the test data until the accuracy was calculated. Accuracy of the implemented model turned out to be 70.8074%.

For the implementation of KNN itself, we used both of our models as input (recommend and evaluate), defined enemy team and allies team and took in queries on both teams. And for each parameters: allies team, enemy team and given hero candidate, radiant and dire probabilities were calculated and recommendation model was used to recommend heroes based on those probabilities. Also, using the evaluation model, the probability of allies’ team winning against the enemy team was calculated.

## 3.2 Algorithm

### 3.2.1 K-Nearest Neighbor (KNN)

K-nearest neighbors is a non-parametric method for classification and regression that predicts objects’ class membership based on K closes training examples in the feature space. It’s called non-parametric because unlike other supervised learning algorithms, KNN doesn’t learn an explicit mapping factor from the training data but rather simply uses training data at the test time to make predictions. We use custom weight and distance function and choose to utilize all of our training examples as “nearest neighbors”.

K-nearest neighbor helps to better model the relationships between heroes instead of simply taking into account wins when a certain hero is present. Here, matches are weighed according to how similar they are to the query match that the user is interested in. example, if we are interested in projecting who will win a specific five on five matchup (our query match), a match with nine of the heroes from the query match present will give us more information on who will win the query match than a match with only one hero from the query match present.

The need for the weight arises when we need to consider all the data sets available as the basic principle of KNN makes prediction from a simple average of a small subset of nearby points and all the other datasets are completely ignored. Hence, we need to smooth out the function and use weighted average instead. This process is also known as Kernel Regression.

Calculating weight,

The following equation represents a combined distance and weighting function used in our K nearest neighbors simulations:

Num\_in\_query

Here,

* d = polynomial scaling of the weight function.
* x = feature vector for the training match i.
* q = query vector which is compared by the logical AND operator to x.
* j = hero ID index of each respective vector.
* Num\_in\_query = number of heroes present in query vector.

The function is normalized to be between 0 and 1, and it gives more weight to matches that more closely resemble the query match. To do this, the function compares the query match vector to the training match vector and counts every instance where a hero is present in both vectors. A larger d will result in similar matches getting much more weight than dissimilar matches. Alternatively, a low d, for example d = 1, will result in each match being weighted solely by how many heroes in common the match has with the query match. Stated another way, a high d will choose to put more emphasis on the synergistic and antagonistic relationships between heroes, while a lower d will put more emphasis on the independent ability of a hero.

Choosing an Optimal Weight Dimension,

To choose the optimal d dimension, number 4 was found out to be optimal through k-fold cross validation method which was performed on 20,000 datasets using k=2.

Due to the time constraint, pre-calculated weight dimension was used as it had an accuracy of 67.43% during the k-fold cross validation.

## 3.3 System Design

### 3.3.1 Class diagram

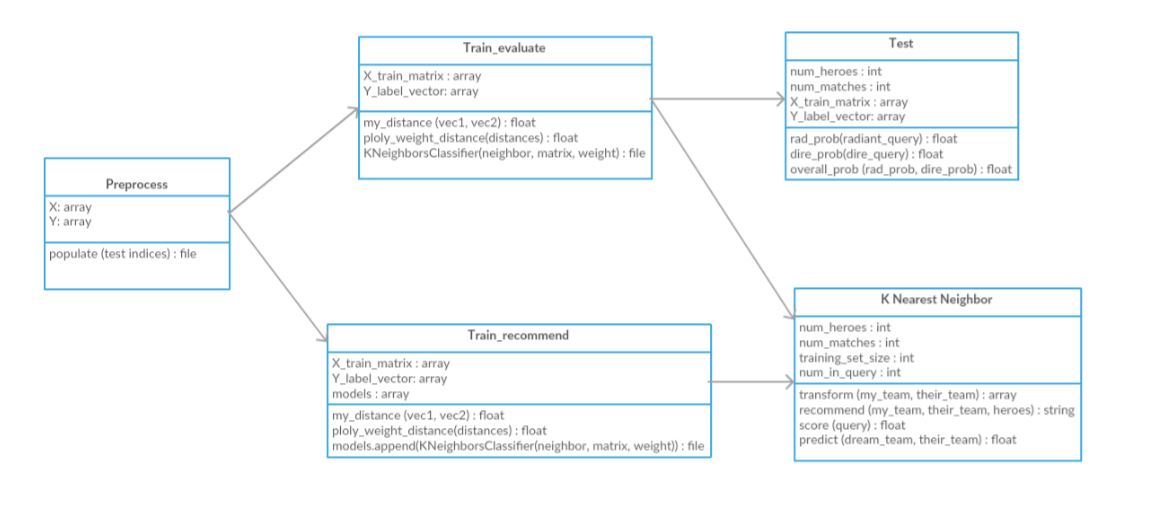


Figure 5 - Class Diagram of Line up prediction system for a MOBA game using KNN algorithm

Figure 5 explains the classes used in Line up prediction system for a MOBA game using KNN algorithm. There are five important classes used in total, preprocess, train\_evaluate, train\_recommend, test, K nearest neighbor.

### 3.3.2 Sequence diagram

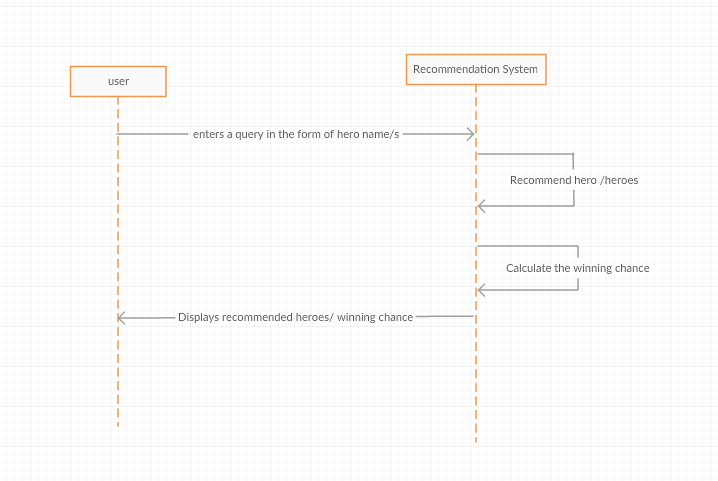


Figure 6 - Sequence Diagram of Line up prediction system for a MOBA game using KNN algorithm

Figure 6 explains the sequence of the Lineup prediction system for a MOBA game using KNN algorithm. Initially, a user opens a browser and enters the query in the query text box. Then the engine looks for the recommended heroes with most winning chances and outputs it along with the winning chance for that configuration.

### 3.3.3 State diagram

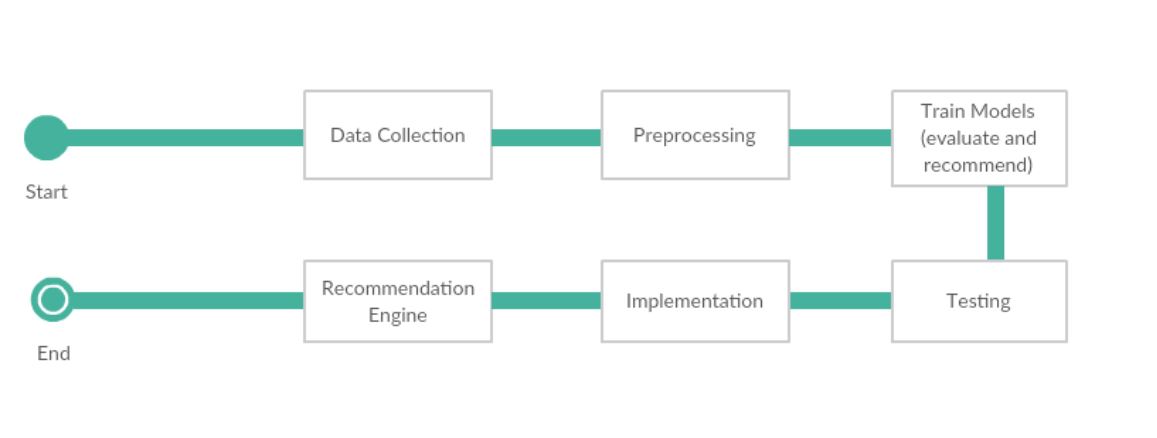


Figure 8 **–** State Diagram of Line up prediction system for a MOBA game using KNN algorithm

Figure 8 explains the state diagram for the system. From the start, states can be seen in orderly fashion as data collection state, preprocessing state, training models state, testing state, implementation state and finally the recommendation engine state.

# CHAPTER 4: IMPLEMENATION AND TESTING

## 4.1 Implementation

User can access the application through a browser and see the interface. They can view a default set of heroes recommended in the application. They can now enter any query heroes in their configuration and any query heroes in the enemy configuration and get the results on the right tab under “recommended heroes”.

As explained, a query with more number of heroes gives more precise results than fewer number of heroes. The recommended heroes and winning chance is changed in real time according to the change in query provided by the user.

### 4.1.1 Tools used

CASE tools:

1. Draw.io

Client Side:

1. Flask
2. HTML
3. Bootstrap CSS

Server side:

1. Python

This section describes the technologies used to build Line up prediction system for a MOBA game using KNN algorithm. This application is a web application that uses Python to develop the back-end and Python Web framework – Flask, HTML and Bootstrap CSS to design the front-end.

Scikit-learn python library was used to implement KNN algorithm.

## 4.2 Description of Major Classes

The major classes in the application are:

### 4.2.1 Preprocess

This class is used to load the stored data in the MongoDB, initialize training matrix, label vector, enumerates the data based on hero IDs and generates train and test sets.

Input: Takes in data stored in database.

Process: Processes each data set and generates train and test sets.

Output: Produces train and test data sets for further processing.

### 4.2.2 test

This class is used to test the accuracy of the implementation of algorithm. It calculates overall probability, compares the predicted result with the actual result and gives the accuracy percentage in the end.

Input: Test data and evaluation model.

Process: Calculates overall probability through comparison between predicted and actual results.

Result: Produces the accuracy percentage of implemented algorithm.

### **4.2.3** K\_nearest\_neighbors

This class implements the K nearest neighbor algorithm using both of the recommend and evaluate files. It calculates the overall probability of recommended team winning based on the given query against enemy team and returns recommended results to itself.

Input: It takes in the evaluation and recommendation models as input.

Process: Scores the query using evaluation model considering both radiant and dire teams and returns the probability of winning against enemy team.

Output: Returns the probability of winning against enemy team to itself along with recommended heroes to complete the query team.

## 4.3 Testing

### 4.3.1 Unit Testing

Each unit of the system was tested for its correct and proper functionality.

Table 3 - Unit Testing of different Components

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test no. | Unit | Test | Expected Result | Test Outcome | Evidence |
| 1 | Preprocess | Classifying data and generating train and test sets. | Train and test datasets as outputs. | Successful | Test 1.1 |
| 2 | Test | Test for the accuracy of the implemented KNN algorithm. | Return the accuracy of implemented algorithm. | Successful | Test 1.2 |
| 3 | K\_nearest\_neighbor | Implement the algorithm recommend heroes to complete query, score the query and return the probability of winning. | Calculates the probability of recommended team, recommends heroes and returns results to itself. | Successful | Test 1.3 |

**Test 1.1: Train and test datasets successfully generated**

**Input:**

Data stored in the database.

**Output:**

Train and test models generated without error.

**Test 1.2: Accuracy of implemented KNN algorithm successfully calculated.**

**Input:**

Test dataset for x matrix and y vector and evaluate model.

**Output:**

Accuracy of KNN model: 0.708074  
 Precision: 0.764119601329  
 Recall: 0.673499267936  
 F1 Score: 0.715953307393  
Support: 3415

**Test 1.3: Algorithm successfully implemented**

**Input:**

Recommendation and evaluation models.

**Output:**

Obtains the recommended heroes for given query, scored the query using evaluation model and probability of winning against enemy for given query without errors.

### 4.3.2 Integration Testing

For integration testing, different individual modules were combined and tested as a group. The integration testing of the system is illustrated in the table below.

Table 5 - Integration testing of the combination of different modules

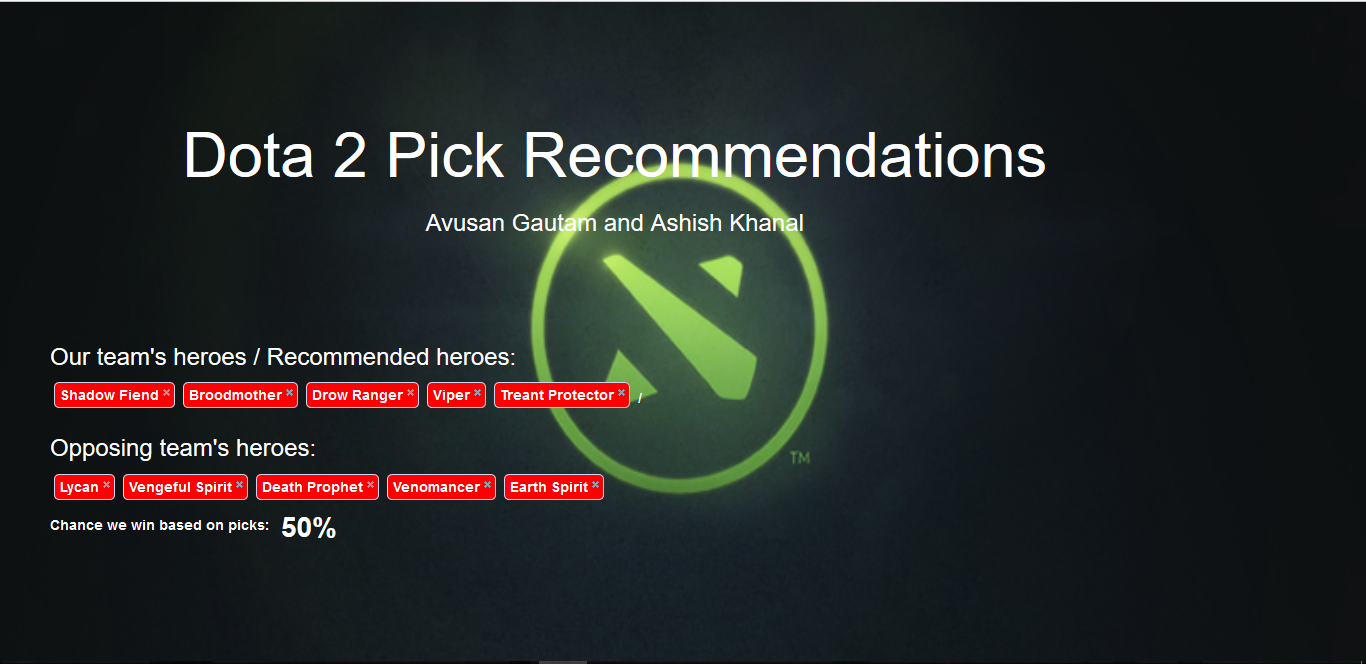
|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Test no. | Part | Test | Expected Result | Test Outcome | Evidence |
| 1 | Engine | Correct implementation of algorithm with hero recommendation and probability calculation. | Recommend heroes based on a given query and output the winning chance. | Successful | Test 2.1 |

**Test 2.1: Engine test**

**Input:**

Any number of heroes (10 max) that count as a query.

**Output:**



# CHAPTER 5: MAINTENANCE AND SUPPORT PLAN

## 5.1 Maintenance Plan

Line up prediction system for a MOBA game using KNN algorithm will implement corrective maintenance for resolving different bugs and errors that may occur when this project is made live. Perfective maintenance will be implemented for further increment in efficiency.

## 5.2 Support Plan

Line up prediction system for a MOBA game using KNN algorithm will be presented to normal players as well as professional players and coaches so that players from all around the globe can use the system to help them select heroes effectively and others can use it to learn hero and team composition for maximum efficiency.

To sustain the application, the DOTA community can explore more algorithms that can be implemented to better study the relationship and impact of heroes in a team and against a team.

# CHAPTER 6: CONCLUSION AND RECOMMENDATION

## 6.1 Conclusion

Line up prediction system for a MOBA game using KNN algorithm was hence, successfully implemented and tested. The recommendation engine recommends heroes based on the given query and provides the winning chance for that configuration. Here, in this project, the average overall accuracy of the KNN algorithm was found to be 70%.

## 6.2 Recommendation

This project did not consider the ever evolving gaming mechanisms and the shift in patches. Therefore, different match datasets can be used in different point of time for maintaining data relevancy.

The binary feature vector used could be stored as an integer in binary representation to improve memory usage.

Also, other algorithms could be used with KNN to produce more accuracy and predict heroes more precisely.

# APPENDIX

# REFERENCES

* [1] K. Conley and D. Perry, “*How does he saw me? a recommendation engine for picking heroes in Dota2*,” CS229 Previous Projects, 2013.
* [2] K. Kalyanaraman, “*To win or not to win? A prediction model to determine the outcome of a dota 2 match*”, CS229 Previous Projects, 2014.
* [3] K. Song, T. Zhang and C. Ma “*Predicting the winning side of Dota2*”, CS229 Previous Projects, 2015.
* DotaBuff, “*Highest win rate, this month*,” Dec. 2016.
* Dota2 Counter-Pick. Dota2 Counter-Pick. N.p., n.d. Web. 14 Nov.2013. <http://dota2cp.com/>.
* A. Agrawal and M. Pierce “*Learning Dota2 team compositions*”, 2014.