Investigations of Dota 2 Hero and Match Data

Rui Fang, Yuan Gao, Shujian Wen, Jiahuan Yu

Abstract

Dota 2 (Defense of the Ancients) is a multiplayer online video game, which is played between two teams, called Radiant and Dire, with five members on each side. The goal for each team is to defend its own and attack the opponents' tower and base or "Ancient". Whichever team succeeds in destroying the base of the other team wins the game. Each of the ten players independently controls a powerful character, known as the "hero". Different heroes have unique abilities and different styles of play. The ban pick of different heroes, the choice of purchasing items for different heroes, and the strategies for various heroes to cooperate together have always been a research topic of top professional Dota 2 teams.

In this project, we utilized data mining techniques to discover some interesting yet useful aspects of Dota 2 based on the analysis of over 60,000 top-level Dota 2 match records. We investigated clustering model of hero types solely based on their in-game statistics without knowing their roles using K-Means Clustering in order to further utilize the clustering result for some other relevant studies. We were able to divide the heroes in 8 clusters. We revealed some interesting facts (e.g. hero win rate, popularity rankings, ban pick statistics) and analyzed the relationship between different influencing factors and game results, as well as relationship between pick rate of heroes and heroes' win rate. We found that there is no strong relationship between pick rate and win rate. We also built prediction model using in-game data for win/lose probability prediction based on team drafts, using Decision Tree, Logistic Regression and k-Nearest Neighbors (KNN) model. We were able to predict game results with a best accuracy of 0.97 among these models.

Introduction

As the rise of the game Defense of the Ancient since year 2003, a new game genre, namely, Multiplayer Online Battle Arena (MOBA), has brought about an increasing number of players in this type of games, including Dota 2, League of Legends, etc. In this project, we utilized Dota 2 as an example and investigated some interesting yet useful aspects of this game, which would reveal the influencing factors leading to the victory of a Dota 2 match, and potentially apply to other MOBA games as well.

In this report, we intended to find out what are the factors influencing the outcome of a Dota 2 match, since how to win a game is basically what people mostly care about when they play a game. In Dota 2, players have the chance to pick the heroes they play and the choice of heroes may play an important role in affecting the result. Previous studies have been conducted to identify the roles and positions of heroes in Dota 2.[1] Gao et. al.[2] used Logistic Regression and Random Forest to classify heros' roles and positions. They managed to detect hero roles with 75% accuracy for both public and professional games and 85 % and 90% accuracy for hero positions respectively. Eggert et al.[3] improved this work also using Logistic Regression and got 96.15% test accuracy. Conley and Perry used Logistic Regression to predict win probability from team drafts from training 18000 examples and got 69.8% test accuracy. They also used KNN on various number of matches to obtain different test accuracy of predicting win probability. In this project, we tried to analyze 60,000 matches, which is a different sample size as previous studies. We also manipulated different training and analysis parameters to compare their test accuracy. We divided this big problem into three small sub-problems.

The first sub-problem we tried to solve was clustering model of hero types. As we know, team balancing and team draft may play an important role in affecting the outcome of a match. The definition of "balanced" for a team could sometimes be arbitrary or vague and could result in misleading conclusions. In Dota 2, the heroes already have their own characteristics based on their primary attributes, i.e., strength, agility and intelligence, and also have roles such as carry, support, pusher, initiator, etc. However, due to the complexity of the game itself, heroes may have the potential to take on more roles than what the game has defined. Therefore, we decided to group the heroes based on their actual in-game data, such as gold per minute, experience per minute, kill/death/assist, heals, to better investigate the actual role the heroes are playing using K-Means Clustering. This was the first task of our project.

The second sub-problem we were trying to solve was the factors influencing the outcome of a game. We mainly focused on two factors. As we know, different heroes play different roles and whether a team is balanced or not may heavily impact the team's performance in a game. We utilized the result we obtained in our first problem and analyzed the relationship between team balancing and match outcome using regression. To further argue whether team balancing has an impact on the outcome of the match, we also predicted win probability based on team drafts, using Decision Tree, Logistic Regression and k-Nearest Neighbors (KNN) model. We also studied the relationship between first blood and the win/loss of a match using regression model, since the advantage in the initial stage of a match may grow into a bigger advantage so this would be an interesting research topic and may bring about some insight for this game.

Finally, we found it intriguing to study the players choices, i.e., pick of the heroes and the reasoning behind these behaviors. We used Pearson product-moment correlation coefficient to analyze the relationship between pick rate of the heroes and their win rate. We hypothesized that heroes with higher pick rate may potentially mean that they lead to higher win rates. Interestingly, we found that the Pearson's r is -0.026 thus there is no relationship between pick rate and win rate.

Methodology

- Clustering model of hero types: K-Means Clustering
- Relationship between first blood and match outcome: Regression
- Win probability prediction based on team drafts / classification model for game result: KNN, Logic Regression, Decision Tree
- Relationship between hero pick rate and win rate: Pearson product-moment correlation coefficient

Code Description

Data collection. Raw data was downloaded from Open Dota Website by querying on the Explorer page: https://www.opendota.com/explorer.

- matches: meta data of matches
- player_matches: detailed performance of each player in a match
- heroes: heroes' information
- picks_ban: pick information of each hero in different matches

Data cleansing.

- Remove unnecessary columns.
- Join matches and player_matches tables to get each player's performance in matches and the match outcome.

Data processing.

Part 1. Clustering Model of Hero Types

1. Generate Hero Vectors
    - Select interested columns from the matches join player_matches table to get each heroes' in-game statistics. Each hero is a vector => [hero_id, kills, deaths, assists, gold, last_hits, denies, gold_per_min, xp_per_min, gold_spent, hero_damage, tower_damage, hero_healing, level]
    - Aggregate heroes by hero_id, calculate mean of each attribute
2. Clustering of Heroes
    - Apply K-Means Clustering
        - Generate and plot the SSQ statistics

    - k = 1 - 20
- Pick n_clusters = 8 and fit the data to get labels for different heroes
    - kmeans = KMeans(n_clusters=8).fit(hero_data)
    - labels = kmeans.labels_

Part 2. Match Outcome Influencing Factors

1. Match Outcome Prediction

- Select interested columns from the matches join player_matches table to get each heroes' in-game statistics. Each hero is a vector => [match_id, player_slot, kills, deaths, assists, gold, last_hits, denies, level, radient_win, is_radient, is_win]
- Aggregate team by match_id and is_raient, calculate the sum of each attribute
- Split training and testing dataset, and train the datasets with Decision Tree, Logistic Regression and k-Nearest Neighbors models
- Compute metrics including Accuracy, Precision, Recall and plot ROC curve to evaluate the performance of each prediction model

2. First Blood & Outcome Relationship

- Select first blood related columns ('firstblood claimed' and 'first blood time') from jointed table, then calculate the corresponding match result, and aggregate by 'match id' and 'is radiant'
- Split training and testing dataset, and train the linear regression model
- Compute metrics including Accuracy, Precision, Recall, Confusion matrix and plot ROC curve to evaluate the performance of the classifier, which also evaluates the correlation between first blood statistics with match outcome.

Part 3. Pick Rate & Win Rate Relationship

1. Hero selection statistics

- Calculate hero selection number and win number among all games and each camp specifically using pandas library.
- Generate histogram to visualize the result.

2. Generate table with hero pick rate and win rate for individual heroes

- Pick rate = (number of matches with hero existing) / (total number of matches)
- Win rate = (number of winning matches with hero existing) / (total number of matches with hero existing)

3. Relationship between pick rate and win rate
    - Pearson product-moment correlation coefficient

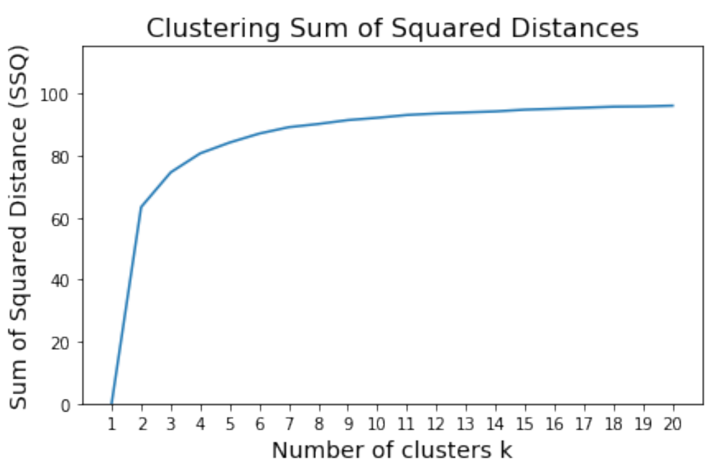
Results

Part 1. Hero Clustering

1. Generate hero vectors and aggregate by hero id

	kills	deaths	assists	last_hits	denies	gold_per_min	xp_per_min	gold_spent	hero_damage	tower_damage	level
	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean	mean
hero_id											
1	6.823423	2.962883	5.501261	374.985225	16.980541	652.436757	657.051892	21718.535135	13676.197117	4531.254054	20.953514
2	6.532593	6.262192	8.362144	144.345727	2.922743	393.459440	436.087880	11826.640512	12353.809029	329.294302	17.274988
3	2.949980	6.332933	11.152461	25.117047	3.522209	247.899760	308.888956	7775.392157	6757.598639	228.969988	14.617647
4	8.385020	5.354008	9.347572	236.993563	16.308953	506.483909	529.891750	16953.282621	19034.640140	2047.499122	19.747221
5	2.803379	6.296534	12.744538	59.079231	1.074862	291.036411	333.853772	9306.788523	8957.451209	208.286630	15.554908

2. Apply K-Means Clustering on the data and plot SSQ statistics



As we can see in this plot, the curve of SSQ reaches an elbow at around k = 8. Therefore, we proceeded and calculated the clustering of heroes based on k = 8.

3. Fit the data with n_clusters = 8 to get clustering for heroes

cluster	hero_id
0	[3, 20, 26, 28, 30, 50, 57, 66, 71, 83, 84, 85...
1	[4, 9, 13, 15, 17, 19, 25, 36, 39, 43, 44, 47,...
2	[2, 14, 16, 23, 51, 55, 58, 65, 69, 78, 92, 96...
3	[34, 113]
4	[6, 18, 21, 33, 41, 42, 49, 53, 61, 77, 81, 89...
5	[1, 8, 10, 11, 12, 46, 48, 73, 80, 82, 94, 95,...
6	[5, 7, 27, 29, 31, 32, 37, 38, 60, 62, 64, 68,...
7	[22, 35, 40, 45, 67]

Part of the result is shown below for demonstration with the names of the heroes:

Group	Hero Names					
0	Bane	Vengeful Spirit	Lion	Dazzle	Witch Doctor	Omniknight
1	Bloodseeker	Mirana	Puck	Razor	Storm Spirit	Tiny
2	Axe	Pudge	Sand King	Kunkka	Clockwerk	Dark Seer
3	Tinker	Arc Warden				
4	Drow Ranger	Sven	Windranger	Dragon Knight	Faceless Void	Wraith King
5	Anti-Mage	Juggernaut	Morphling	Shadow Fiend	Phantom Lancer	Templar Assassin
6	Crystal Maiden	Earthshaker	Shadow Shaman	Tidehunter	Lich	
7	Zeus	Sniper	Venomancer	Pugna	Spectre	

Interestingly, the heroes that are clustered into one label have something in common. For example, for Group 0, all these heroes are suitable supports, with excellent controlling abilities, mostly for controlling individual heroes. Heroes in Group 6 are also good supports, but they are better in team fights. Heroes in Group 1 are heroes that are good for taking the middle lane with excellent escape abilities, while heroes in Group 3 may distribute traps all across the map.

Part 2. Match Outcome Influencing Factors

1. Win probability prediction based on team drafts

With match and player data provided, we are able to use data mining techniques to predict game results given two team’s information.

1.1 Data preparation

The first step for game prediction is to find relevant player features that could impact the result of a game: kills, deaths, assists, last hits, denies, gold and levels. We group the table by match id and player slot, so that each data row is a team, and then aggregate the above feature columns by the sum of each team member.

Win probability prediction based on team drafts

```
[ ] player_data = df[["match_id", "player_slot", "kills", "deaths", "assists", "gold", "last_hits", "denies", "level", "radiant_win"]]
    # get rid of NaN gold rows
    player_data = player_data[np.isfinite(player_data['gold'])]

    # add column to check win/lose
    player_data["is_radiant"] = pd.to_numeric(player_data["player_slot"]) <= 4
    player_data["is_win"] = (player_data["is_radiant"] == player_data["radiant_win"])
```

```
[ ] # check any NaN value
    player_data.isnull().values.any()
```

```
False
```

```
[ ] player_data.head()
```

	match_id	player_slot	kills	deaths	assists	gold	last_hits	denies	level	radiant_win	is_radiant	is_win
69130	1384504402	2	1	6	4	1.0	166	25	16	False	True	False
69131	1384504402	3	3	4	3	1124.0	90	2	12	False	True	False
69132	1384504402	4	5	5	1	1.0	244	22	17	False	True	False
69133	1384504402	128	11	3	8	4393.0	136	7	19	False	False	True
69134	1384504402	129	1	3	11	1936.0	27	2	13	False	False	True

```
[ ] m_data = player_data.groupby(['match_id', 'is_radiant']).agg({"kills": ["sum"], "deaths": ["sum"],
                                                               "assists": ["sum"], "gold": ["sum"], "last_hits": ["sum"],
                                                               "denies": ["sum"], "level": ["sum"], "is_win":[lambda x: x.all()]})

    ow many teams
    .(team_data)
```

```
83878
```

```
[ ] team_data.head()
```

		kills	deaths	assists	gold	last_hits	denies	level	is_win
		sum	sum	sum	sum	sum	sum	sum	<lambda>
match_id	**is_radiant**								
1384504402	**False**	23	12	50	14721.0	659	33	85	True
	True	12	24	18	2480.0	565	53	68	False
1384528617	**False**	3	17	12	1326.0	407	24	53	False
	True	16	4	31	9162.0	474	47	65	True
1384560471	**False**	29	14	40	13196.0	765	48	90	True

1.2 Create training and testing datasets

The second step is to split the data into training and testing datasets. The label to predict is the game result (is_win). Here we use 30% of the data as test datasets.

```
# Create training and testing datasets
from sklearn.model_selection import train_test_split
dt_data = team_data.iloc[:, :-1]
dt_data = dt_data.as_matrix()
dt_label = team_data['is_win']
dt_label = dt_label.as_matrix()
x_train, x_test, y_train, y_test = train_test_split(dt_data, dt_label, test_size=0.3)
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:3: FutureWarning: Method .as_matrix will be removed in a future version. Us
  This is separate from the ipykernel package so we can avoid doing imports until
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: FutureWarning: Method .as_matrix will be removed in a future version. Us
  """
```

1.3 Decision Tree Model

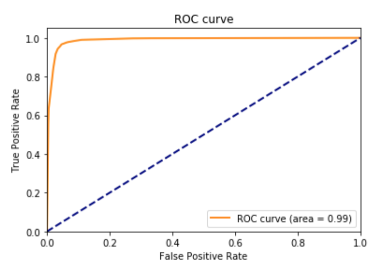
We apply decision tree model to train and predict game results.

Decision Tree Model

```
# Use decision tree model to train the datasets
from sklearn import tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix

clf = tree.DecisionTreeClassifier(max_depth=5)
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

print("Accuracy score is: ", accuracy_score(y_test, y_pred))
print("Precision score is: ", precision_score(y_test, y_pred))
print("Recall score is: ", recall_score(y_test, y_pred))
print("Confusion matrix is: \n", confusion_matrix(y_test, y_pred))
```

```
Accuracy score is:  0.9597838181529169
Precision score is:  0.9550544201706992
Recall score is:  0.9653343886030866
Confusion matrix is:
 [[11955   574]
 [  438 12197]]
```

```
# Plot ROC curve
from sklearn.metrics import roc_curve, auc

y_score = clf.predict_proba(x_test)[:,1]
fpr, tpr, threshold = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()
```



1.4 Logistic Regression Model

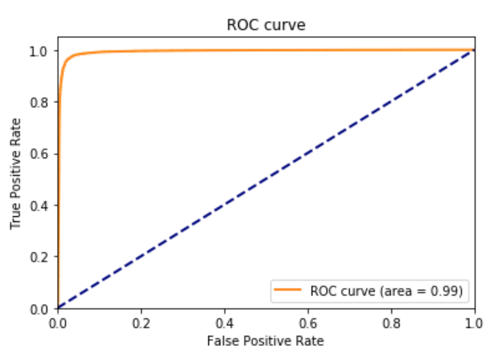
We apply logistic regression model to train and predict game results.

Logistic Regression Model

```
[ ] # Use logistic regression model to train the datasets
    from sklearn.linear_model import LogisticRegression

    clf = LogisticRegression()
    clf = clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)

    print("Accuracy score is: ", accuracy_score(y_test, y_pred))
    print("Precision score is: ", precision_score(y_test, y_pred))
    print("Recall score is: ", recall_score(y_test, y_pred))
    print("Confusion matrix is: \n", confusion_matrix(y_test, y_pred))
```

```
/usr/local/lib/python3.6/dist-packages/sklearn/linear_model/logistic.:
  FutureWarning)
/usr/local/lib/python3.6/dist-packages/sklearn/utils/validation.py:72
  y = column_or_1d(y, warn=True)
Accuracy score is:  0.9695596884438086
Precision score is:  0.9690933523041657
Recall score is:  0.9703205381875742
Confusion matrix is:
 [[12138   391]
 [  375 12260]]
```

```
[ ] # Plot ROC curve
    y_score = clf.predict_proba(x_test)[:,1]
    fpr, tpr, threshold = roc_curve(y_test, y_score)
    roc_auc = auc(fpr, tpr)
    plt.figure()
    lw = 2
    plt.plot(fpr, tpr, color='darkorange',
    lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
    plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
    plt.xlim([0.0, 1.0])
    plt.ylim([0.0, 1.05])
    plt.xlabel('False Positive Rate')
    plt.ylabel('True Positive Rate')
    plt.title('ROC curve')
    plt.legend(loc="lower right")
    plt.show()
```



1.5 K-Nearest Neighbor Model

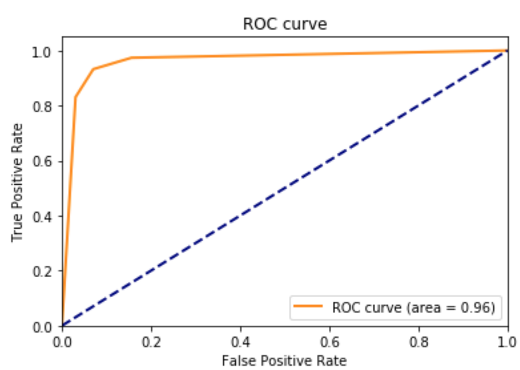
We apply logistic regression model to train and predict game results.

K-Nearest Neighbor Model

```
[140] from sklearn.neighbors import KNeighborsClassifier

      clf = KNeighborsClassifier(n_neighbors=3)
      clf = clf.fit(x_train, y_train)
      y_pred = clf.predict(x_test)

      print("Accuracy score is: ", accuracy_score(y_test, y_pred))
      print("Precision score is: ", precision_score(y_test, y_pred))
      print("Recall score is: ", recall_score(y_test, y_pred))
      print("Confusion matrix is: \n", confusion_matrix(y_test, y_pred))
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: Dat
  after removing the cwd from sys.path.
Accuracy score is:  0.9308933396916229
Precision score is:  0.9303997471954495
Recall score is:  0.9320933913731698
Confusion matrix is:
 [[11648   881]
 [  858 11777]]
```

```
[141] # Plot ROC curve
      y_score = clf.predict_proba(x_test)[:,1]
      fpr, tpr, threshold = roc_curve(y_test, y_score)
      roc_auc = auc(fpr, tpr)
      plt.figure()
      lw = 2
      plt.plot(fpr, tpr, color='darkorange',
      lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
      plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
      plt.xlim([0.0, 1.0])
      plt.ylim([0.0, 1.05])
      plt.xlabel('False Positive Rate')
      plt.ylabel('True Positive Rate')
      plt.title('ROC curve')
      plt.legend(loc="lower right")
      plt.show()
```



2. First Blood & Outcome Relationship

2.1 Select first blood related fields from the dataset

```
player_data = df[["firstblood_claimed", "first_blood_time", "match_id", "player_slot", "radiant_win"]]
# get rid of NaN firstblood_claimed rows
player_data = player_data[np.isfinite(player_data['firstblood_claimed'])]
# add column to check win/lose
player_data["is_radiant"] = pd.to_numeric(player_data["player_slot"]) <= 4
player_data["is_win"] = (player_data["is_radiant"] == player_data["radiant_win"])
team_data = player_data.groupby(['match_id', 'is_radiant']).agg({"firstblood_claimed": ["max"], "first_blood_time": ["mean"],
                                                                 "is_win":[lambda x: x.all()]})
```

2.2 Verify first 5 entries

```
team_data.head()
```

		firstblood_claimed	first_blood_time	is_win
		max	mean	<lambda>
match_id	is_radiant			
18355350	True	0.0	139	True
19009163	True	0.0	206	False
19249598	True	0.0	403	False
19254348	True	0.0	54	True
19266829	True	0.0	62	False

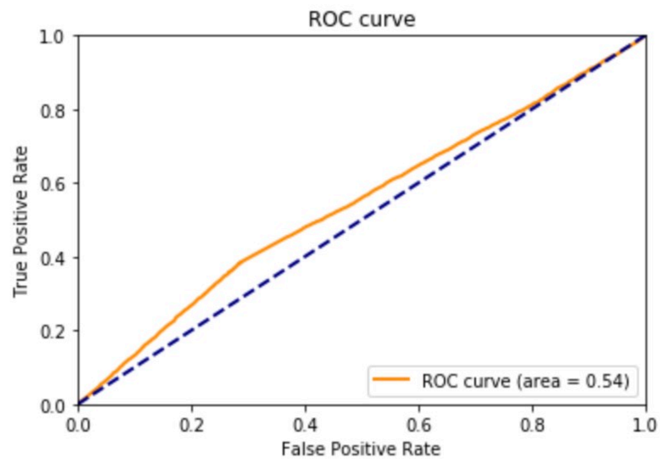
2.3 Train linear regression model and calculate metrics

```
from sklearn import tree
from sklearn.metrics import accuracy_score, precision_score, recall_score, confusion_matrix
import matplotlib.pyplot as plt
from sklearn.linear_model import LogisticRegression

clf = LogisticRegression()
clf = clf.fit(x_train, y_train)
y_pred = clf.predict(x_test)

print("Accuracy score is: ", accuracy_score(y_test, y_pred))
print("Precision score is: ", precision_score(y_test, y_pred))
print("Recall score is: ", recall_score(y_test, y_pred))
print("Confusion matrix is: \n", confusion_matrix(y_test, y_pred))
```

```
Accuracy score is:  0.5468085106382978
Precision score is:  0.5787131466716113
Recall score is:  0.385737238907428
Confusion matrix is:
 [[5617 2272]
 [4970 3121]]
```

2.4 Plot ROC curve

```
from sklearn.metrics import roc_curve, auc

y_score = clf.predict_proba(x_test)[:,1]
fpr, tpr, threshold = roc_curve(y_test, y_score)
roc_auc = auc(fpr, tpr)
plt.figure()
lw = 2
plt.plot(fpr, tpr, color='darkorange',
lw=lw, label='ROC curve (area = %0.2f)' % roc_auc)
plt.plot([0, 1], [0, 1], color='navy', lw=lw, linestyle='--')
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.0])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('ROC curve')
plt.legend(loc="lower right")
plt.show()
```



Based on the above experiment, we can see from the metric and ROC curve, the relationship between first blood (firstblood_claimed and first_blood_time) and match result is fairly weak. It is not reliable to predict the match result

Part 3. Pick Rate & Win Rate Relationship

1. Hero selection among all games

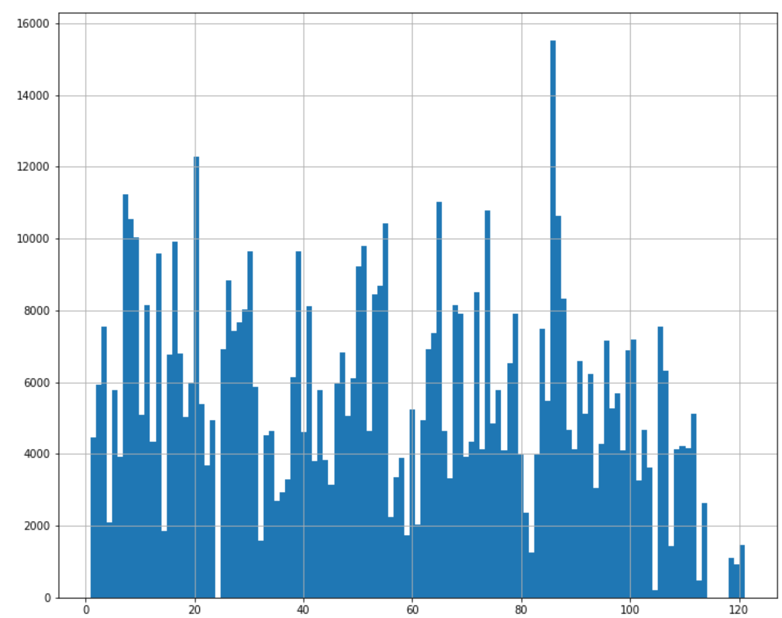


Figure above shows the hero selection counts in all games.

```
[ ] heros = df['hero_id']
    heros.value_counts()
```

```
86      15518
20      12279
7       11248
65      11036
74      10797
        ...
82       1245
119      1112
120       931
113       467
105       206
Name: hero_id, Length: 116, dtype: int64
```

Top 5 of the most popular heroes among all game are Rubick, Vengeful Spirit, Earthshaker, Batrider, Invoker and lease popular are Meepo, Dark Willow, Pangolier, Arc Warden, Techies.

2. Radiant win hero selection

```
[27] heros_radiant_win = radiant_win['hero_id']
     heros_radiant_win.value_counts()
```

```
86      4018
20      3154
7       2927
65      2899
9       2806
        ...
82       348
120      281
119      270
113      117
105       53
Name: hero_id, Length: 116, dtype: int64
```

3. Dire win hero selection

```
[28] heros_dire_win = dire_win['hero_id']
     heros_dire_win.value_counts()
```

```
86      3812
20      3194
65      2727
8       2654
74      2635
        ...
82       271
119      263
120      207
113      111
105       48
Name: hero_id, Length: 116, dtype: int64
```

4. Hero selection within all winning team

```
heros_win = df_win_match['hero_id']
heros_win.value_counts()
```

```
86     7830
20     6348
65     5626
7      5508
8      5434
       ...
82      619
119     533
120     488
113     228
105     101
Name: hero_id, Length: 116, dtype: int64
```

5. Generate pick rate and win rate

```
hero_data = df[["match_id", "hero_id", "player_slot", "radiant_win"]]
number_of_matches = 66009
# add column to check win/lose
hero_data["is_radiant"] = pd.to_numeric(hero_data["player_slot"]) <= 4
hero_data["is_win"] = (hero_data["is_radiant"] == hero_data["radiant_win"])
hero_data.head()
# hero_aggregation_data = hero_data.groupby(['hero_id']).agg({"hero_pick", "is_win"})
```

```
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:4: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_g
  after removing the cwd from sys.path.
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:5: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/user_g
  """
```

	match_id	hero_id	player_slot	radiant_win	is_radiant	is_win
0	1000018815	15	2	True	True	True
1	1000018815	86	130	True	False	False
2	1000018815	78	129	True	False	False
3	1000018815	30	1	True	True	True
4	1000018815	67	131	True	False	False

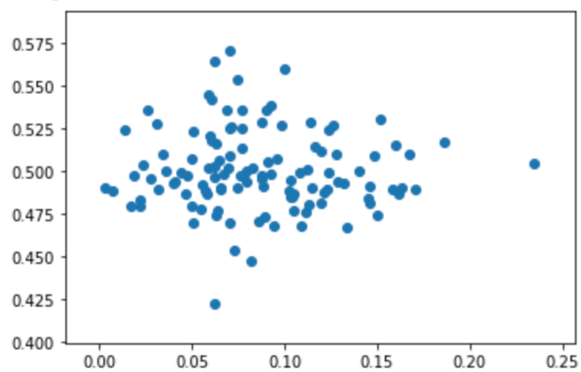
```
# Prepare data with hero pick rate and win rate
mid_table = hero_data.groupby('hero_id').agg({"is_radiant": [lambda x: float((x.sum() + (x == False).sum()) * 10 / len(hero_data.index)
                                              "is_win": [lambda x: float(x.sum() / (x.sum() + (x == False).sum()))]})
pick_win_rate = mid_table.rename(columns={'is_radiant': 'pick_rate', 'is_win': 'win_rate'})
pick_win_rate.head(5)
```

	pick_rate	win_rate
	<lambda>	<lambda>
hero_id		
1	0.067370	0.498313
2	0.089609	0.473035
3	0.114378	0.528609
4	0.031708	0.489250
5	0.087730	0.528579

6. Plot showing relationship between pick rate and win rate:

```
# Plot between pick rate and win rate
import matplotlib.pyplot as plt
xs=pick_win_rate['pick_rate']
ys=pick_win_rate['win_rate']
plt.scatter(xs,ys)
```

```
<matplotlib.collections.PathCollection at 0x7f714a655390>
```



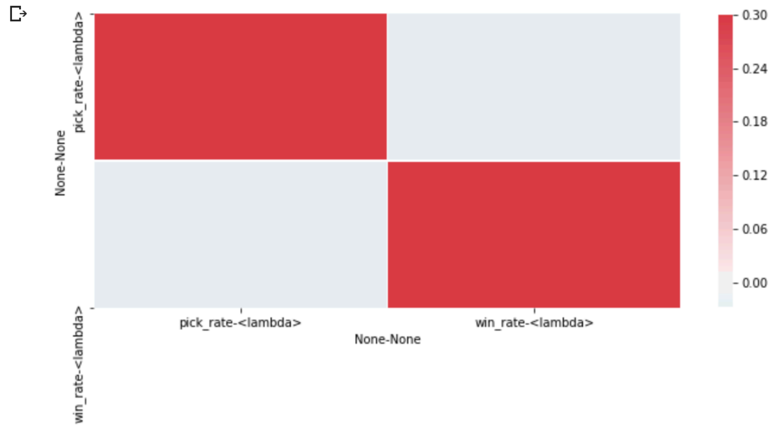
From the plot above we can see the scatter is really spread and seems there is no obvious pattern between pick rate and win rate.

7. Pearson's r and heat map:

```
# Pearson's r
corr=pick_win_rate.corr()
print(corr)
```

```
                      pick_rate  win_rate
                       <lambda>  <lambda>
pick_rate <lambda>   1.000000 -0.026916
win_rate  <lambda> -0.026916  1.000000
```

```python
import seaborn as sns
import matplotlib.pyplot as plt
f, ax = plt.subplots(figsize=(11, 9))
# Generate a custom diverging colormap
cmap = sns.diverging_palette(220, 10, as_cmap=True)

# Draw the heatmap with the mask and correct aspect ratio
sns.heatmap(corr, cmap=cmap, vmax=.3, center=0,
            square=True, linewidths=.5, cbar_kws={"shrink": .5})
plt.show()
```



We can see the Pearson's r is -0.026916 between pick rate and win rate. According to the definition of Pearson's r, we can say there is no relationship between pick rate and win rate.

Discussion

Part 1. Hero Clustering

We were able to cluster the heroes into 8 groups using K-Means Clustering only based on the in-game performance of each hero. The attributes we actually looked at include kills, deaths, assists, gold, last hits, denies, gold per min, xp per min, gold spent, hero damage, tower damage, hero healing and level at the end of the match. Based on the result, the clustering of the heroes is somewhat meaningful when we combine it with practical matches. For example, Bane, Vengeful Spirit, Lion, etc. are all good supports with excellent controlling abilities and are especially good for fighting with individuals or smaller team fights. Axe, Pudge, Sand King, etc. are all heroes that are suitable for taking the hard lane. Heroes that belong to Group 1 and Group 7 are all good for taking the middle lane, but our clustering helps better group them into different types. Heroes in Group 1 have better escape abilities, while heroes in Group 7 focus more on damage. Overall speaking, the clustering of heroes agrees with the actual matches pretty well.

Part 2. Match Outcome Influencing Factors

- **Match Outcome Prediction**

This is the result table grouped from the three prediction models:

Model	Accuracy	Precision	Recall	Roc
Decision Tree	0.960	0.955	0.965	0.99
Logistic Regression	0.970	0.969	0.970	0.99
K-Nearest Neighbor	0.931	0.930	0.932	0.96

As is shown from the table, all prediction models achieved great performance with a demonstrated accuracy, precision and recall scores above 93%. Among them, Logistic Regression has the best performance.

- **First Blood & Outcome Relationship**

The experiment result shows that the relationship between first blood (firstblood_claimed and first_blood_time) and match result is fairly weak. The resulting Accuracy, Precision and Recall scores are 0.546, 0.578 and 0.385 respectively. Confusion matrix is: [[5617 2272], [4970 3121]]. The area below ROC curve is also only 0.55, which is pretty much close to a random classifier. Therefore, it is not reliable to predict the match result. More features such as statistics of first wower may be required to train a more reliable predictor.

Part 3. Pick Rate & Win Rate Relationship

In statistics, the Pearson correlation coefficient (PCC, pronounced /ˈpɪərsən/), also referred to as Pearson's r, the Pearson product-moment correlation coefficient (PPMCC) or the bivariate correlation, is a measure of the linear correlation between two variables X and Y. According to the Cauchy–Schwarz inequality it has a value between +1 and −1, where 1 is total positive linear correlation, 0 is no linear correlation, and −1 is total negative linear correlation.[4] We can see the Pearson’s r is -0.026916 between pick rate and win rate. According to the definition of Pearson’s r, we can say there is no relationship between pick rate and win rate.

Based on the result and the well-known fact of moba game, it is reasonable that hero with high pick rate may not have high win rate. The victory rate of the hero has a lot to do with the strength of the version while the popularity of a hero does not. Also if a hero is really popular and has really high picking rate, it is also possible that the level of players are more likely to be uneven. The winning rate would be affected by that. So it would be better to analyze the result by dividing the matches to different player levels.

Future Work

For hero clustering, one reasonable work to do in the future is using hero clustering to find the definition of “balanced” to study what is considered a balanced team and how the team draft could affect the outcome of a match. We can also combine our results with the roles of heroes defined in Dota 2 game to find out whether the definition inside the game is reasonable and helping players make their choices when they decide on the team draft. What’s more, it would be better to analyze the game result and different game factors by dividing the matches into different player levels. Another thing we can do is to do a hero recommendation during the hero selection stage based on machine learning results.

Conclusion

We were able to build the clustering model and group the Dota 2 heroes into 8 types based on their in-game statistics using K-Means Clustering. The results agree well with practical matches and would help players determine their team draft and potentially help with team balancing in the future. We were also able to apply three different prediction models to the data set. Logistic Regression has the best performance over Decision Tree and Linear Regression prediction models, with a demonstrated accuracy, precision and recall scores above 96.9%. We also find out that first blood information has weak correlation with the match outcome. In order to make a better match outcome prediction, we would like to incorporate some more statistics such as first tower to evaluate more precise result. In terms of pick rate and win rate, there is no strong relationship between the hero popularity and victory rate.

References

1. N. Pobiedina and J. Neidhardt, “On successful team formation,” tech. rep., 2013.
2. L. Gao, J. Judd, D. Wong, and J. Lowder, “Classifying Dota 2 Hero Characters Based on Play Style and Performance,” 2013.
3. C. Eggert, M. Herrlich, J. Smeddinck, and R. Malaka, “Classification of Player Roles in the Team-Based Multi-player Game Dota 2," in *Entertainment Computing - ICEC 2015* (K. Chorianopoulos, M. Divitini, J. Baalsrud Hauge, L. Jaccheri, and R. Malaka, eds.), vol. 9353 of *Lecture Notes in Computer Science*, (Cham), pp. 112-125, Springer International Publishing, 2015.
4. Wikipedia: Pearson correlation coefficient

   https://en.wikipedia.org/wiki/Pearson_correlation_coefficient