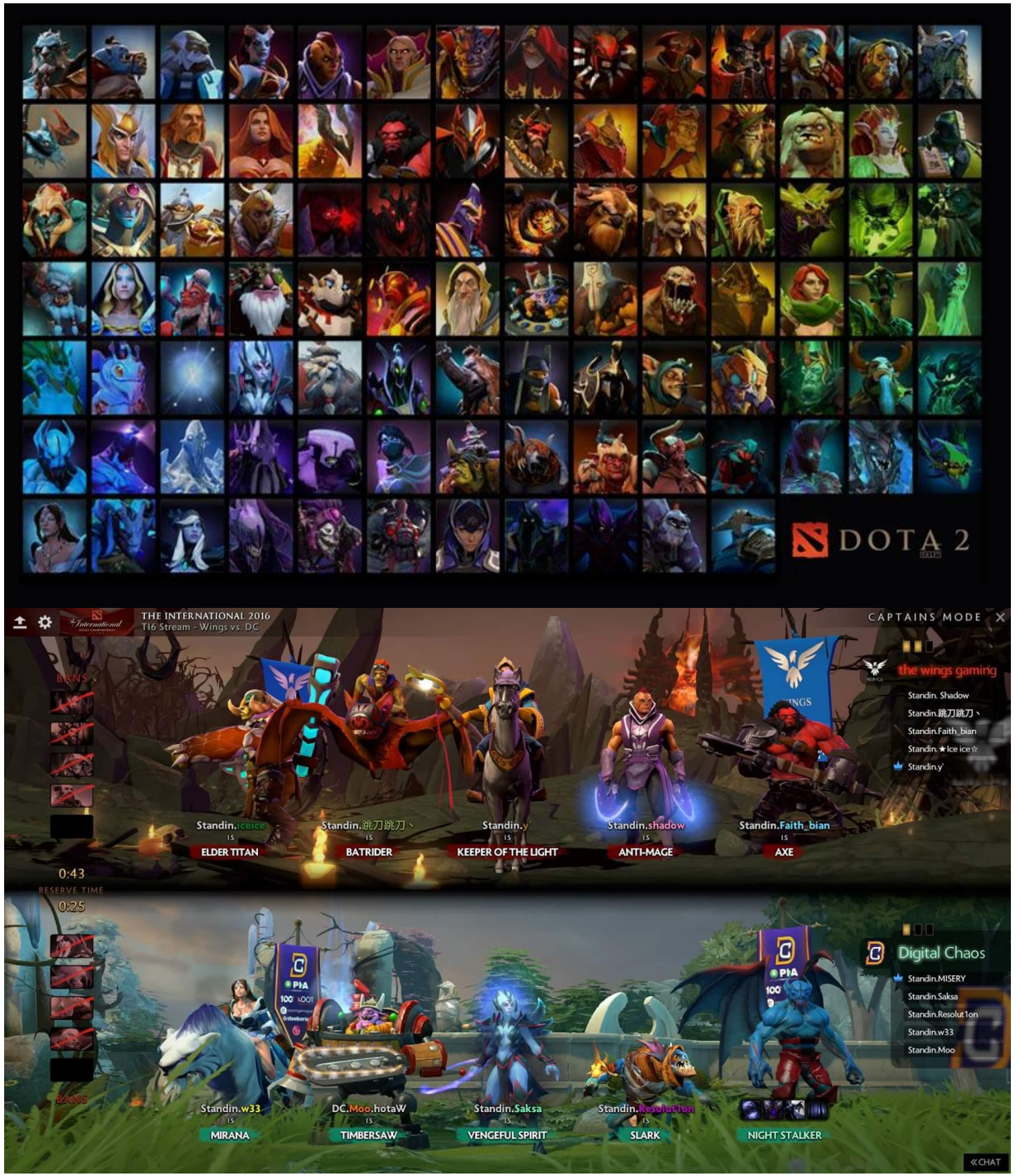
Dota2 Winning Prediction System

Presented by team 3
Hanwen Jiang
Renqiu Chen

Background

- ***Dota 2*** is a multiplayer online battle arena (MOBA) video game developed and published by Valve.
- **Multiplayer online battle arena** (**MOBA**) is a subgenre of strategy video games in which two teams of players compete against each other on a predefined battlefield.
- Each player controls a single character with a set of distinctive abilities that improve over the course of a game and which contribute to the team's overall strategy.



Technology

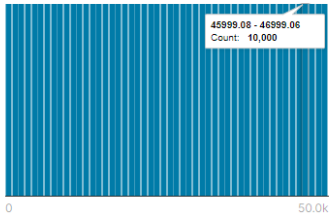
- ***Datasource:*** 5000 ranked ladder matched from the Dota2 data dump created by Opendota
- **Apache Spark** is an open-source unified analytics engine for large-scale data processing. Spark provides an interface for programming entire clusters with implicit data parallelism and fault tolerance.
- **Play Framework** is an open-source web application framework which follows the model–view–controller (MVC) architectural pattern. It is written in Scala and usable from other programming languages that are compiled to JVM bytecode

Dataset – Training part

players.csv (129.94 MB)

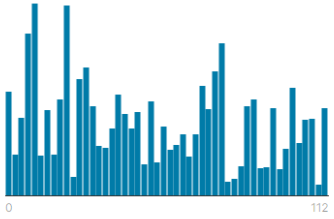
Detail Compact Column 3 of 73 columns

match_id



Valid	500k	100%
Mismatched	0	0%
Missing	0	0%
Mean	25k	
Std. Deviation	14.4k	
Quantiles	0	Min
	12.5k	25%
	25.0k	50%
	37.5k	75%
	50.0k	Max

hero_id



Valid	500k	100%
Mismatched	0	0%
Missing	0	0%
Mean	50.6	
Std. Deviation	32.8	
Quantiles	0	Min
	21	25%
	47	50%
	75	75%
	112	Max

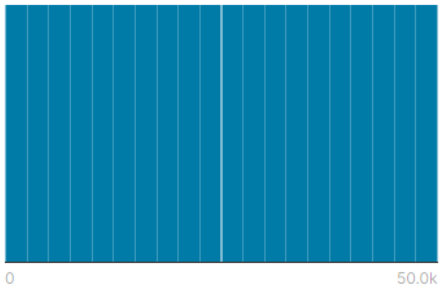
player_slot

Valid	500k	100%
Mismatched	0	0%
Missing	0	0%
Mean	66	
Std. Deviation	64	
Quantiles	0	Min
	2	25%
	128	50%
	130	75%
	132	Max

match.csv (2.69 MB)

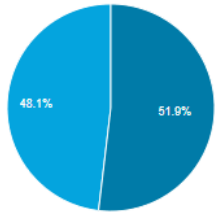
Detail Compact Column 2 of 13 columns

match_id



Valid	50.0k	100%
Mismatched	0	0%
Missing	0	0%
Mean	25k	
Std. Deviation	14.4k	
Quantiles	0	Min
	12.5k	25%
	25.0k	50%
	37.5k	75%
	50.0k	Max

radiant_win

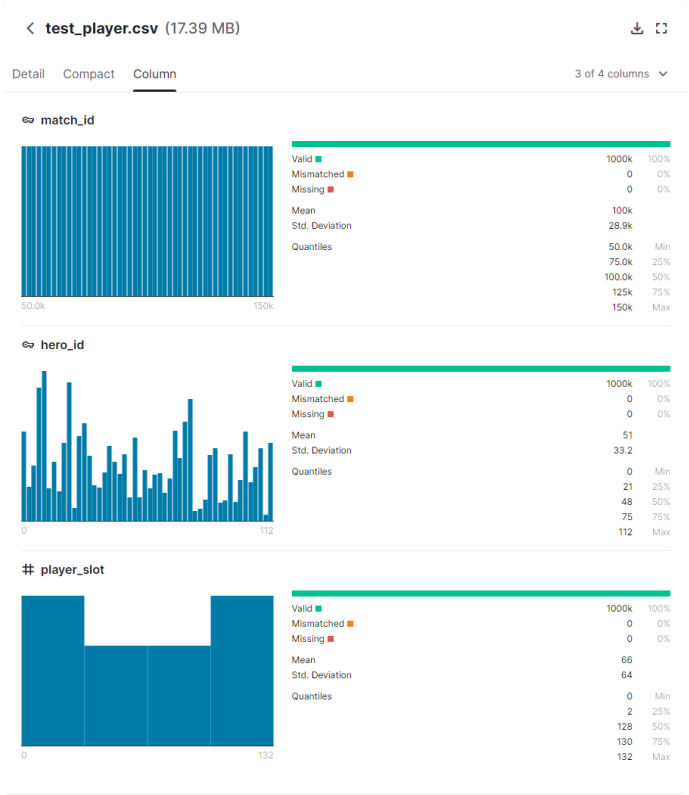


true 25.9k 52%

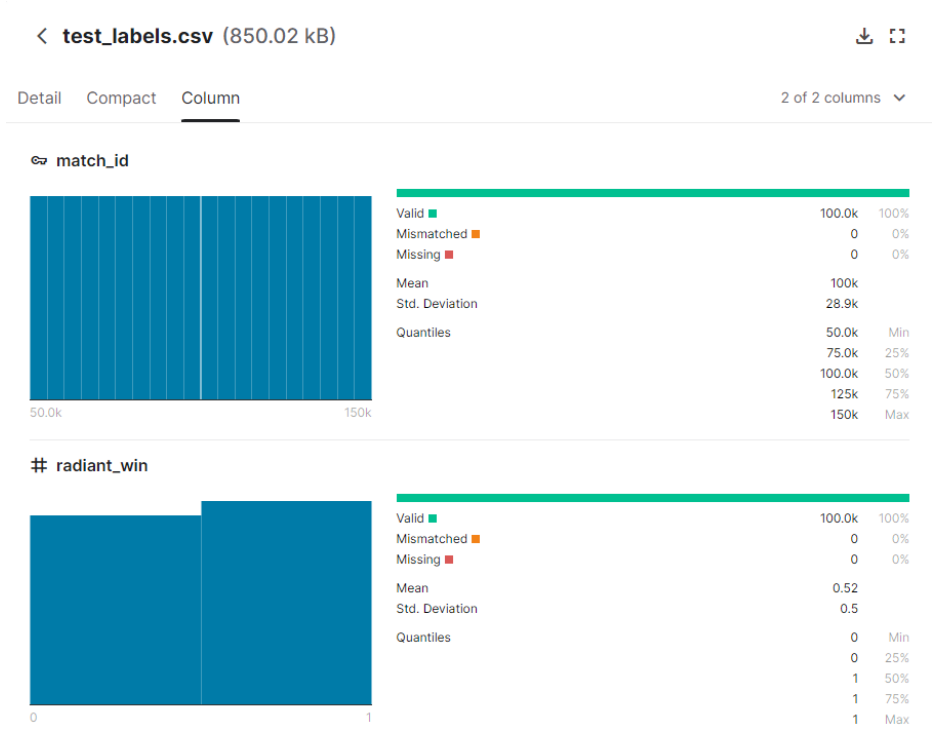
false 24.1k 48%

Valid	50.0k	100%
Mismatched	0	0%
Missing	0	0%
True	25.9k	52%
False	24.1k	48%

Dataset – Testing part



- match_id: Main key of the data, use to specify each match
- hero_id: The id of the hero, vary from 1 to 113
- player_slot: The id of ten players in one match, 0~4 is in one team. 128~132 is in another



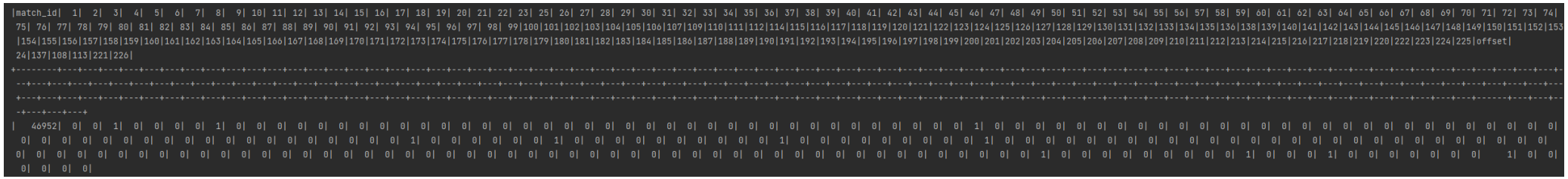
- radiant_win: The out come of one match, 0 means radiant win, 1 means dire win

Data pre-process - Cleansing

match_id	account_id	hero_id	player_slot
0 50.0k	0 158k	0 112	0 132
720	2956	0	3
1032	0	0	0
1108	4765	0	131
2134	9442	0	3
2134	9441	0	4
2773	13922	0	131
7098	0	0	3
7488	6069	0	2
7582	35721	0	132
7831	35936	0	4
13396	55450	0	132
14592	38021	0	0
22764	86659	0	131
24509	39206	0	130

Data noisy: abnormal hero_id

Data pre-process - Transformation



One-hot encoder: Used to extend feature. Its values are only 0 and 1. Different types are stored in the vector.

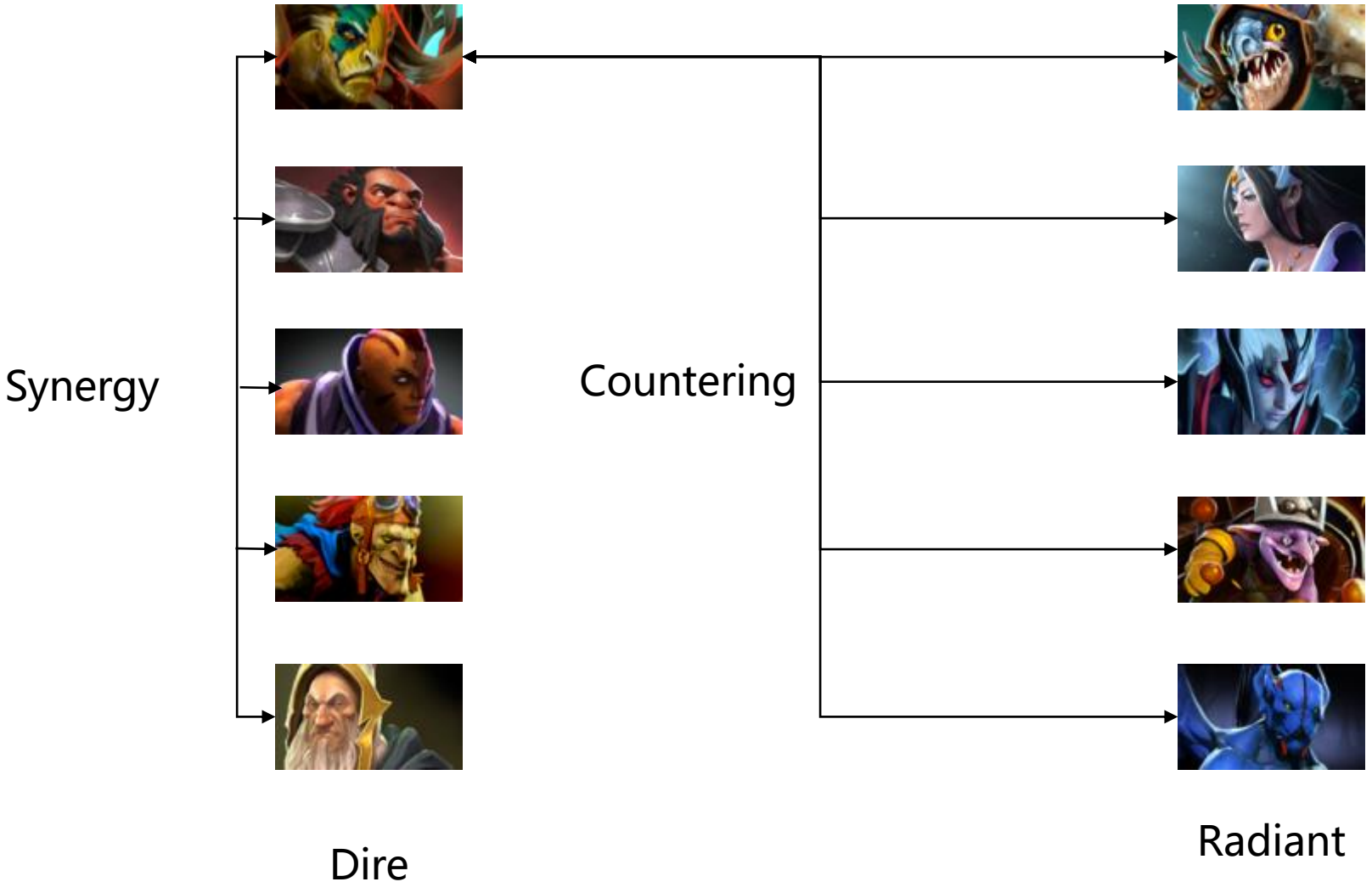
$$X_{1+i} = \begin{cases} 1 & \text{if hero } i \text{ is on Radiant side} \\ 0 & \text{otherwise} \end{cases}$$

$$X_{111+i} = \begin{cases} 1 & \text{if hero } i \text{ is on Dire side} \\ 0 & \text{otherwise} \end{cases}$$

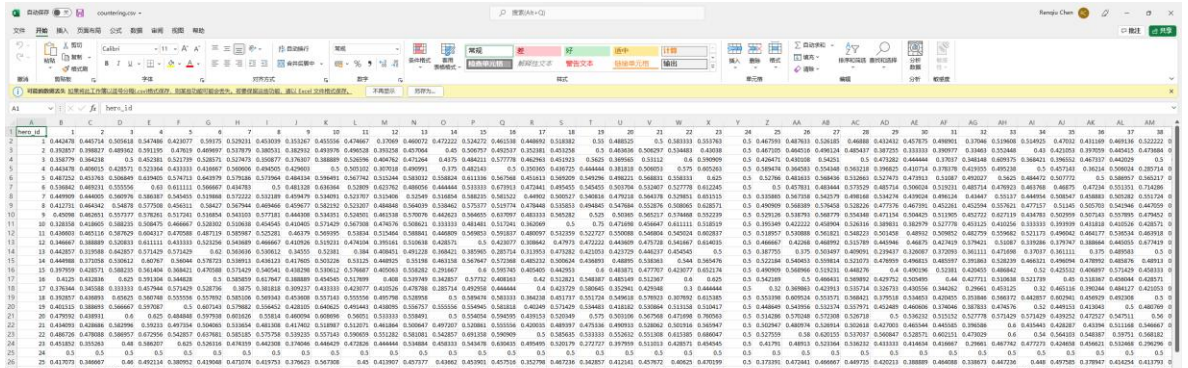
match_id	0	1	2	3	4	128	129	130	131	132
38422	4	101	98	12	76	38	20	35	44	7
26425	55	21	25	53	104	100	3	28	26	106
11458	67	26	100	37	75	17	5	86	107	8
22373	68	100	84	8	73	33	31	109	106	78
49855	87	81	39	11	8	72	9	44	17	7

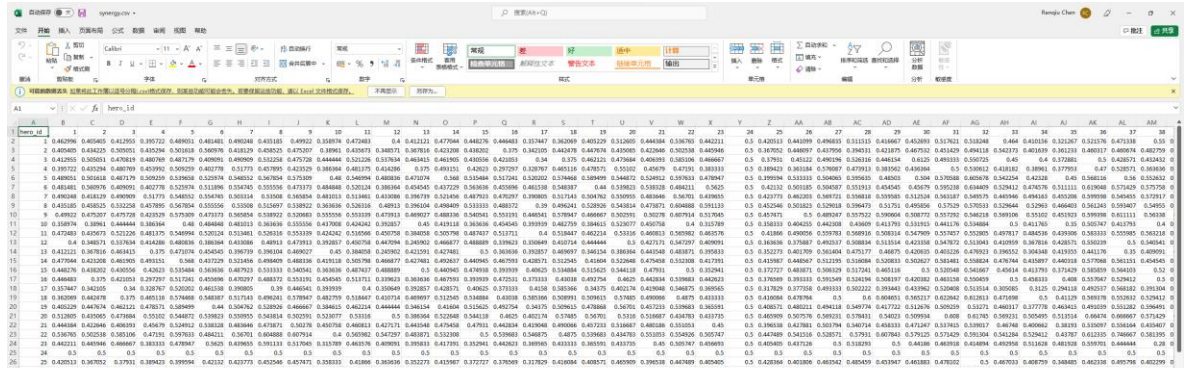
Group by match_id: Easy to extract feature

Data pre-process – Feature extraction



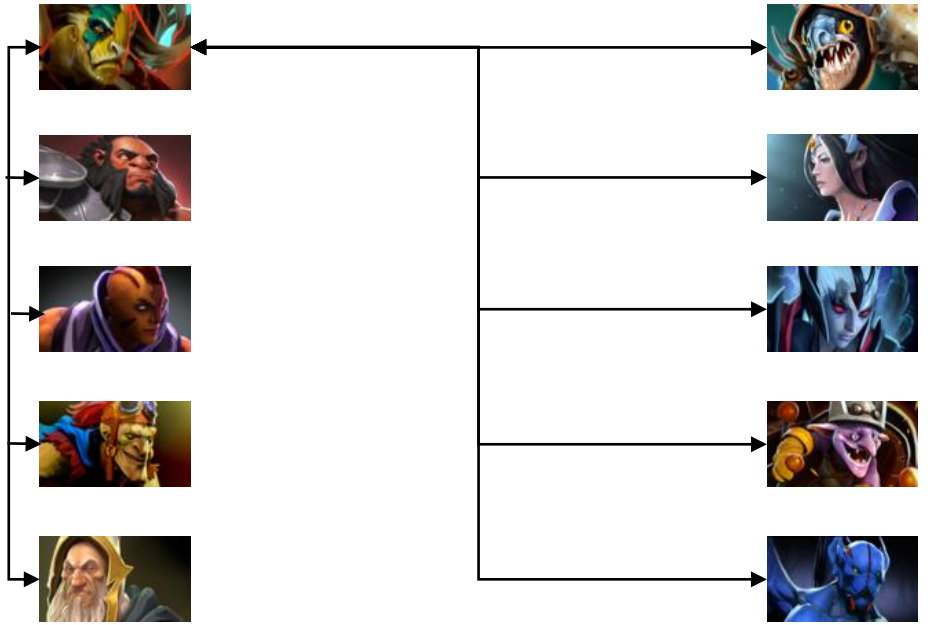
Synergy, Countering Matrix(113*113)





Synergy(i)(j) contains the hero_i and hero_j winrate when they are in the same team
Countering(i)(j) contains the hero_i and hero_j winrate when they are in the different team

Data pre-process – Feature extraction



$$Synergy = \sum_{ij} Synergy\_\mathrm{Rad}iant - \sum_{ij} Synergy\_Dire$$

$$Countering = \sum_{ij} Countering\_\mathrm{Rad}iant - \sum_{ij} Countering\_Dire$$

$$Offset = 1$$

(let the model to consider the difference between Radiant and dire)

Data pre-process – Final result

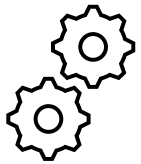
```
|match_id|radiant_win|  1|  2|  3|  4|  5|  6|  7|  8|  9| 10| 11| 12| 13| 14| 15| 16| 17| 18| 19| 20| 21| 22| 23| 25| 26| 27| 28| 29| 30| 31| 32| 33| 34| 35| 36| 37| 38| 39| 40| 41| 42| 43| 44| 45| 46| 47| 48| 49| 50| 51| 52| 53| 54| 55| 56| 57| 58| 59| 60| 61| 62| 63| 64| 65| 66| 67| 68| 69| 70| 71|
 72| 73| 74| 75| 76| 77| 78| 79| 80| 81| 82| 83| 84| 85| 86| 87| 88| 89| 90| 91| 92| 93| 94| 95| 96| 97| 98| 99|100|101|102|103|104|105|106|107|109|110|111|112|114|115|116|117|118|119|120|121|122|123|124|125|126|127|128|129|130|131|132|133|134|135|136|138|139|140|141|142|143|144|145|146|147|148|149|150
 |151|152|153|154|155|156|157|158|159|160|161|162|163|164|165|166|167|168|169|170|171|172|173|174|175|176|177|178|179|180|181|182|183|184|185|186|187|188|189|190|191|192|193|194|195|196|197|198|199|200|201|202|203|204|205|206|207|208|209|210|211|212|213|214|215|216|217|218|219|220|222|223|224|225
 |offset| 24|137|108|113|221|226|              synergy|          countering|
+--------+-----------+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+-
 --+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---
 +---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+---+-----
 -+---+---+---+---+---+---+-------------------+-------------------+
|     148|        1.0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|
 0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
 1|  0|  0|  0|  0|  0|  0|-0.03914255800000005|0.015759247999998394|
|     463|        1.0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  1|  1|  0|  0|
 0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
 1|  0|  0|  0|  0|  0|  0|        0.383974935|-0.1218113139999856|
|     471|        1.0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
 0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  1|
 1|  0|  0|  0|  0|  0|  0| -0.8696013339999997|  0.4234631130000004|
```

```
|     471|        1.0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|
 0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  1|
 1|  0|  0|  0|  0|  0|  0| -0.8696013339999997|  0.4234631130000004|
|     496|        1.0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|
 1|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|
 1|  0|  0|  0|  0|  0|  0| -0.6479660530000001|  0.6697397880000011|
|     833|        0.0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
 0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  1|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|  0|
 1|  0|  0|  0|  0|  0|  0|        0.373444967|   0.454159461999998|
```

229 Dense Vector

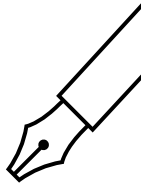
113 hero *2 + Synergy + Countering + Offset

Training



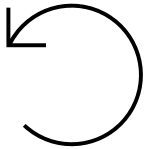
VectorAssembler

Assemble 229 feature into one Vector



StringIndexer

Transform the label into certain type



StringIndexer

Automatically identify categorical feature, and index them

Model selection

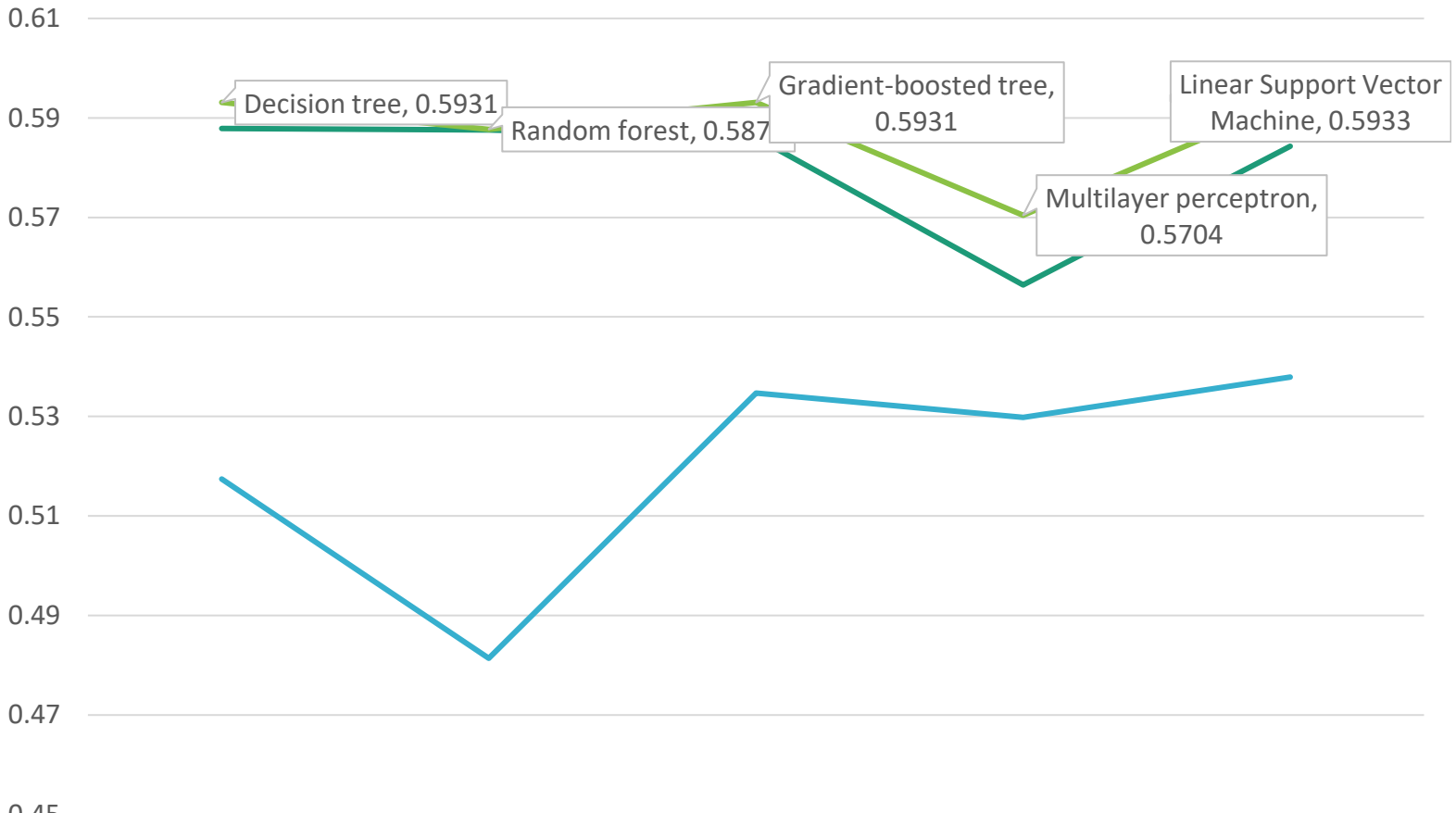
Train the data according to different model

Challenges

0.5	0.21875	0.412698	0.416667	0.405797
0.5	0.416	0.452915	0.487179	0.551351
0.5	0.5	0.52	0.529412	0.523404
0.5	0.371134	0.43299	0.557692	0.303371
0.5	0.435789	0.488889	0.506098	0.58705
0.5	0.402985	0.461988	0.411765	0.529412
0.5	0.45625	0.440476	0.6	0.610714
0.5	0.382353	0.375	0.533333	0.480519
0.5	0.333333	0.37931	0.6	0.61194
0.5	0.504798	0.571992	0.572034	0.632495
0.5	0.428571	0.461255	0.538462	0.570025
0.5	0.479542	0.472318	0.590674	0.571721
0.5	0.464072	0.515723	0.576336	0.612698
0.5	0.48917	0.497619	0.570815	0.572727

- ☒ Abnormal synergy data caused by insufficient data quantity
- ☑ The running time is too long because the query is required every time
- ☒ The feature Vector is so large that it’ s hard to handle
- ☑ The Synergy and Countering is hard to calculate due to limitation of dataframe operation

Testing - Performance



	Decision tree	Random forest	Gradient-boosted tree	Multilayer perceptron	Linear Support Vector Machine
Precision	0.5879	0.5876	0.586	0.5564	0.5843
Accuracy	0.5931	0.5877	0.5931	0.5704	0.5933
Callback	0.5174	0.4814	0.5347	0.5298	0.5379

Close to Acceptance Criteria 60%

User Interface

Acceptance Criteria

Model	Accuracy
Decision Tree	59.31%
Linear Support Vector Machine	59.33%
Random Forest	58.77%
Multilayer Perceptron	57.04%
Gradient-boosted Tree	59.31%

Time for Training Models:
Around 45 minutes

Time for Loading model:
Around 3 minute

Time for predicting :
Around 0.3 seconds

Test Case

- Raw dataframe should contain all rows and all columns
- Raw dataframe should not contain unsuitable data
- Processed dataframe should satisfy some equations

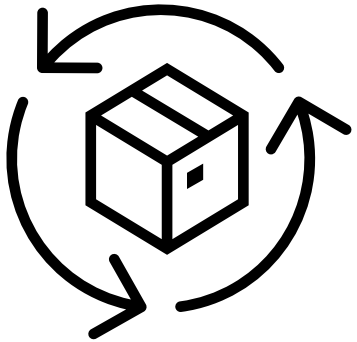
```
behavior of "raw dataset"
it should "work for row length" in {
  val rowPlayers = players.count()
  val rowMatch = matches.count()
  val synergy_row=synergy_result.count()
  val countering_row=countering_result.count()
  val row_tm=test_labels.count()
  val row_tp=test_player.count()
  rowPlayers shouldBe 50000
  rowMatch shouldBe 50000
  synergy_row shouldBe 113
  countering_row shouldBe 113
  row_tm shouldBe 1000000
  row_tp shouldBe 1000000
}
```

```
behavior of "data strict"
it should "work for value restrictions" in {
  val numRW = players.filter(players("radiant_win") === "true").count()
  val numDW = players.filter(players("radiant_win") === "flase").count()
  numRW+numDW shouldBe 50000

  val heroName114 = heronames.filter(heronames("hero_id") >= 114).count()
  heroName114 shouldBe 0
  val heroName0 = heronames.filter(heronames("hero_id") < 0).count()
  heroName0 shouldBe 0
  val heroName24 = heronames.filter(heronames("hero_id") === 24).count()
  heroName0 shouldBe 0
  val heroName108 = heronames.filter(heronames("hero_id") === 108).count()
  heroName0 shouldBe 0
  val heroName113 = heronames.filter(heronames("hero_id") === 113).count()
  heroName0 shouldBe 0

  val player_slot0 = players.filter(players("player_slot") < 0).count()
  player_slot0 shouldBe 0
  val player_slot4128 = players.filter(players("player_slot") > 4 && players("player_slot") < 128).count()
  player_slot4128 shouldBe 0
  val player_slot132 = players.filter(players("player_slot") > 132 ).count()
  player_slot132 shouldBe 0

  val player_hero0= players.filter(players("hero_id") < 0 ).count()
  player_hero0 shouldBe 0
  val player_hero24= players.filter(players("hero_id") === 24 ).count()
  player_hero24 shouldBe 0
  val player_hero113= players.filter(players("hero_id") === 113 ).count()
  player_hero113 shouldBe 0
  val player_hero108= players.filter(players("hero_id") === 108 ).count()
  player_hero108 shouldBe 0
}
```

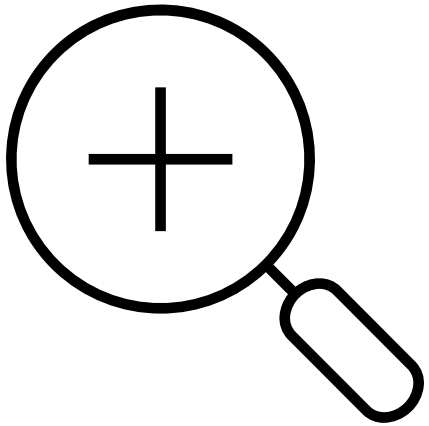
Use Cases

- Administrator:
  System will process the data and predict which side will win the game
  User:
- User can select the ten heroes' names from the list in a match
- User can see the Synergy or Countering between heroes in this match

Future Work



Optimize the model’s parameter and extract features to improve the model’s performance



Add more functionalities to current system, such as hero recommendation or predict the result during the game


Thanks
DOTA 2
VALVE