

DOTA2 WINNING TEAM PREDICTION FROM CHOSEN HEROES BY MACHINE LEARNING METHODS

JIN HAICHEN 16T2006

CONTENT

- What is Dota2
- Get data
- Process data
- Models
- Result
- GUI

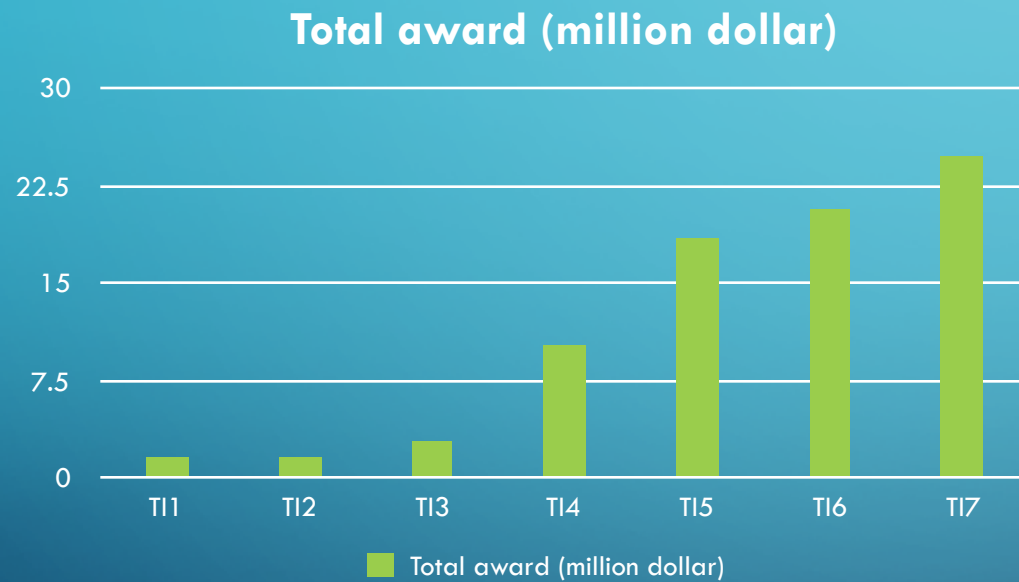
WHAT IS DOTA2 (PLAYERS)

- Dota2 is a free-to-play multiplayer online battle arena (MOBA) video game.

Owners: 118,526,603 ± 281,478
Players in the last 2 weeks: 9,423,417 ± 90,192 (7.95%)
Players total: 118,526,603 ± 281,478 (100%)
Peak concurrent players yesterday: 848,778
YouTube stats: 5,536,873 views and 31,519 comments for videos uploaded last week, over 50 new videos uploaded yesterday



WHAT IS DOTA2 (THE INTERNATIONAL)



WHAT IS DOTA2 (GAME CONTENT)

- Heroes choosing



Gameplay



DATA SOURCE (OPENDATA)

ID	DURATION	RADIANT	DIRE
3657358001 > Divine [5]	33:04 2 days ago		
3657371701 > Divine [5]	23:57 2 days ago		
3687372809 > Divine [5]	43:28 2 days ago		
3687379401 > Divine [5]	23:39 2 days ago		
3687383706 > Divine [5]	43:55 2 days ago		
3687388700 > Divine [5]	36:59 2 days ago		
3687421602 > Divine [5]	44:16 2 days ago		
3687426104 > Divine [5]	28:12 2 days ago		

GET DATA (MATCHES)

```
def mine_data(count=1000, timeout=15):
    matches = []
    less_than_match_id =
    less_than_match_str = ""
    while len(matches) < count:
        try:
            response = request(request_url + less_than_match_str, timeout=timeout)
        except:
            print('Failed to make a request, starting w. match ID: ', less_than_match_id)
            time.sleep(timeout)
            less_than_match_id -= 1
        else:
            response_json = json.loads(response)
            less_than_match_id = response_json[-1]['match_id']
            less_than_match_str = f'less than match id={str(less_than_match_id)}'
            matches.extend(response_json)
            print(len(matches), end=' ')
    return matches
```


GET DATA (10 MILLION MATCHES)

```
[{"match_id": 3593803506, "match_seq_num": 3125412135, "radiant_win": false, "start_time": 1512187734, "duration": 104, "avg_mmr": 4218, "num_mmr": 2, "lobby_type": 0, "ga"}, {"match_id": 3593875304, "match_seq_num": 3125410800, "radiant_win": false, "start_time": 1512187124, "duration": 1046, "avg_mmr": 4200, "num_mmr": 0, "lobby_type": 7, "ga"}, {"match_id": 3593875101, "match_seq_num": 3125414153, "radiant_win": true, "start_time": 1512187115, "duration": 1372, "avg_mmr": 4207, "num_mmr": 3, "lobby_type": 0, "ga"}, {"match_id": 3593871300, "match_seq_num": 3125414143, "radiant_win": true, "start_time": 1512187071, "duration": 1308, "avg_mmr": 4017, "num_mmr": 4, "lobby_type": 7, "ga"}, {"match_id": 3593871304, "match_seq_num": 3125416184, "radiant_win": true, "start_time": 1512186929, "duration": 1374, "avg_mmr": 4355, "num_mmr": 8, "lobby_type": 7, "ga"}, {"match_id": 3593867505, "match_seq_num": 3125417239, "radiant_win": true, "start_time": 1512186756, "duration": 1721, "avg_mmr": 3668, "num_mmr": 7, "lobby_type": 7, "ga"}, {"match_id": 3593867706, "match_seq_num": 3125414414, "radiant_win": true, "start_time": 1512186713, "duration": 1605, "avg_mmr": 3998, "num_mmr": 4, "lobby_type": 7, "ga"}, {"match_id": 3593867001, "match_seq_num": 3125413858, "radiant_win": false, "start_time": 1512186711, "duration": 1650, "avg_mmr": 3747, "num_mmr": 7, "lobby_type": 7, "ga"}, {"match_id": 3593865900, "match_seq_num": 3125419153, "radiant_win": true, "start_time": 1512186707, "duration": 1467, "avg_mmr": 1812, "num_mmr": 3, "lobby_type": 0, "ga"}, {"match_id": 3593865404, "match_seq_num": 3125412786, "radiant_win": false, "start_time": 1512186692, "duration": 1575, "avg_mmr": 1919, "num_mmr": 5, "lobby_type": 7, "ga"}, {"match_id": 3593865203, "match_seq_num": 3125416851, "radiant_win": true, "start_time": 1512186571, "duration": 1589, "avg_mmr": 3600, "num_mmr": 5, "lobby_type": 7, "ga"}, {"match_id": 3593865805, "match_seq_num": 3125417151, "radiant_win": false, "start_time": 1512186645, "duration": 1837, "avg_mmr": 1768, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593863700, "match_seq_num": 3125418319, "radiant_win": false, "start_time": 1512186644, "duration": 1319, "avg_mmr": 4291, "num_mmr": 7, "lobby_type": 7, "ga"}, {"match_id": 3593863806, "match_seq_num": 3125411110, "radiant_win": false, "start_time": 1512186632, "duration": 540, "avg_mmr": 3599, "num_mmr": 4, "lobby_type": 7, "ga"}, {"match_id": 3593865504, "match_seq_num": 3125419887, "radiant_win": false, "start_time": 1512186627, "duration": 1576, "avg_mmr": 3701, "num_mmr": 3, "lobby_type": 0, "ga"}, {"match_id": 3593865105, "match_seq_num": 3125416429, "radiant_win": false, "start_time": 1512186615, "duration": 1856, "avg_mmr": 3045, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593864405, "match_seq_num": 3125414599, "radiant_win": false, "start_time": 1512186581, "duration": 1787, "avg_mmr": 3265, "num_mmr": 5, "lobby_type": 7, "ga"}, {"match_id": 3593864307, "match_seq_num": 3125417484, "radiant_win": false, "start_time": 1512186564, "duration": 1501, "avg_mmr": 3729, "num_mmr": 3, "lobby_type": 0, "ga"}, {"match_id": 3593864707, "match_seq_num": 3125410675, "radiant_win": true, "start_time": 1512186573, "duration": 1838, "avg_mmr": 5104, "num_mmr": 9, "lobby_type": 7, "ga"}, {"match_id": 3593863908, "match_seq_num": 3125415670, "radiant_win": true, "start_time": 1512186555, "duration": 1652, "avg_mmr": 4073, "num_mmr": 4, "lobby_type": 7, "ga"}, {"match_id": 3593863809, "match_seq_num": 3125415430, "radiant_win": true, "start_time": 1512186549, "duration": 1729, "avg_mmr": 2712, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593863201, "match_seq_num": 3125416691, "radiant_win": true, "start_time": 1512186524, "duration": 1931, "avg_mmr": 3635, "num_mmr": 0, "lobby_type": 7, "ga"}, {"match_id": 3593863008, "match_seq_num": 3125413895, "radiant_win": true, "start_time": 1512186512, "duration": 1661, "avg_mmr": 1091, "num_mmr": 2, "lobby_type": 7, "ga"}, {"match_id": 3593862700, "match_seq_num": 3125410815, "radiant_win": true, "start_time": 1512186493, "duration": 1592, "avg_mmr": 2350, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593862505, "match_seq_num": 3125414510, "radiant_win": true, "start_time": 1512186486, "duration": 1280, "avg_mmr": 3389, "num_mmr": 2, "lobby_type": 7, "ga"}, {"match_id": 3593862200, "match_seq_num": 3125414495, "radiant_win": true, "start_time": 1512186473, "duration": 1731, "avg_mmr": 3033, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593862103, "match_seq_num": 3125413128, "radiant_win": false, "start_time": 1512186471, "duration": 1776, "avg_mmr": 3402, "num_mmr": 1, "lobby_type": 3, "ga"}, {"match_id": 3593862102, "match_seq_num": 3125416453, "radiant_win": false, "start_time": 1512186465, "duration": 1940, "avg_mmr": 3502, "num_mmr": 5, "lobby_type": 7, "ga"}, {"match_id": 3593862003, "match_seq_num": 3125417882, "radiant_win": true, "start_time": 1512186460, "duration": 2003, "avg_mmr": 3448, "num_mmr": 2, "lobby_type": 7, "ga"}, {"match_id": 3593861507, "match_seq_num": 3125416293, "radiant_win": true, "start_time": 1512186435, "duration": 2163, "avg_mmr": 4326, "num_mmr": 4, "lobby_type": 0, "ga"}, {"match_id": 3593861406, "match_seq_num": 3125413735, "radiant_win": false, "start_time": 1512186426, "duration": 1015, "avg_mmr": 1770, "num_mmr": 3, "lobby_type": 7, "ga"}, {"match_id": 3593861307, "match_seq_num": 3125410133, "radiant_win": true, "start_time": 1512186423, "duration": 1452, "avg_mmr": 3488, "num_mmr": 4, "lobby_type": 7, "ga"}, {"match_id": 3593860904, "match_seq_num": 3125414814, "radiant_win": true, "start_time": 1512186413, "duration": 1850, "avg_mmr": 4638, "num_mmr": 4, "lobby_type": 0, "ga"}, {"match_id": 3593860800, "match_seq_num": 3125416153, "radiant_win": true, "start_time": 1512186406, "duration": 2641, "avg_mmr": 3755, "num_mmr": 2, "lobby_type": 7, "ga"}]
```


GET DATA (MATCHES)

- {
 - ...
 - "radiant_win": false, ← winning team (radiant or dire)
 - "duration": 504, ← duration time (seconds)
 - "avg_mmr": 4218, ← average match rating (player levels)
 - "radiant_team": "64,97,104,93,51", ← chosen heroes id
 - "dire_team": "82,14,1,42,35"
- }

GET DATA (HEROES)

Hero id

Hero's Chinese name

Hero's name

```
1, 敌法师, Anti-Mage  
2, 斧头, Axe  
3, 祸乱之源, Bane  
4, 血之狂魔, Bloodseeker  
5, 水晶室女, Crystal Maiden  
6, 卓尔游侠, Drow Ranger  
7, 撼地者, Earthshaker  
8, 主宰, Juggernaut  
9, 米拉娜, Mirana  
11, 影魔, Shadow Fiend  
10, 克林姆火灵, Morphling  
12, 幻影长矛手, Phantom Lancer  
13, 卜豆, Puck  
14, 帕吉, Pudge  
15, 剃刀, Razor  
16, 沙王, Sand King  
17, 风暴精灵, Storm Spirit  
18, 斯温, Sven  
19, 小小, Tiny  
20, 复仇之魂, Vengeful Spirit  
21, 风行侠, Windranger  
22, 宙斯, Zeus
```

GET DATA (SAVE AND LOAD MATCHES DATA)

```
def save_original_data(matches, file_name='origin.txt'):
    with open(file_name, 'w') as f:
        for match in matches:
            f.write(json.dumps(match) + '\n')
```

```
def load_original_data(file_name='origin.txt'):
    matches = []
    with open(file_name, 'r') as f:
        for line in f.readlines():
            json_str = line.strip()
            matches.append(json.loads(json_str))
    return matches
```

origin.txt 150,524 KB

GET DATA (PROCESS DATA)

```
def process_data(matches, duration_min=0, duration_max=3600, nmr_min=0, nmr_max=10000):
    teams_win = []
    matches_heroes = []
    for match in matches:
        duration = match['duration']
        nmr = match['avg_nmr']
        if isinstance(duration, int) is False or isinstance(nmr, int) is False:
            continue
        if duration < duration_min or duration > duration_max:
            continue
        if nmr > nmr_max or nmr < nmr_min or match['lobby_type'] is not 7:
            continue
        match_heroes = [0 for i in range(120)]
        match_heroes2 = [0 for i in range(120)]
        radiant_team = match['radiant_team'].split(',')
        dire_team = match['dire_team'].split(',')
        if match['radiant_win']:
            team_win = 1
        else:
            team_win = -1
        if len(radiant_team) is not 5 or len(dire_team) is not 5:
            continue
        for hero_id in radiant_team:
            match_heroes[int(hero_id)-1] = 1
            match_heroes2[int(hero_id)-1] = -1
        for hero_id in dire_team:
            match_heroes[int(hero_id)-1] = -1
            match_heroes2[int(hero_id)-1] = 1
        teams_win.append(team_win)
        matches_heroes.append(match_heroes)
        teams_win.append(-team_win)
        matches_heroes.append(match_heroes2)
    return sklearn_dataset(matches_heroes, teams_win)
```


PROCESS DATA (ONE-HOT ARRAY)

- "radiant_team": "64,97,104,93,51",
- "dire_team": "82,14,1,42,35"

- →

ally team hero

enemy team hero

not used hero

PROCESS DATA (SPLIT FOR TRAINING AND TESTING)

```
matches_train, matches_test = train_test_split(matches, test_size=0.2)
matches_data_train = process_data(matches_train)
matches_data_test = process_data(matches_test)
XM, XT, YM, YT = matches_data_train.data, matches_data_test.data, matches_data_train.target, matches_data_test.target
```

- Input:
 - Heroes choosing status (one-hot array): [120 elements of 1, 0, -1]
 - Winning status (a number): 1 means team wins and -1 means team loses

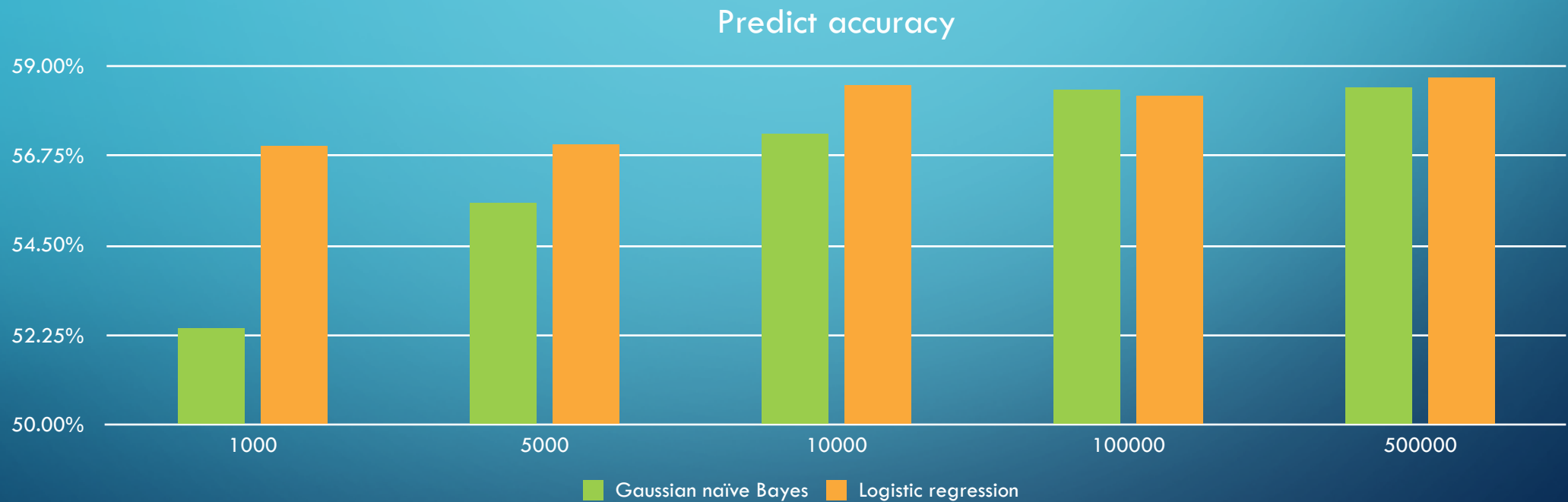
RESULTS (DIFFERENT ALGORITHMS)

Algorithms	Predict accuracy
Gaussian naïve Bayes	58.45%
Perceptron	52.62%
Logistic regression	58.73%
Decision tree	51.70%
Random forest	54.04%
Ada boost	57.02%

RESULTS (DIFFERENT TOTAL MATCH COUNT)

Total match count	Gaussian naïve Bayes	Logistic regression
1,000	52.40%	57.00%
5,000	55.56%	57.02%
10,000	57.30%	58.54%
100,000	58.40%	58.26%
500,000	58.45%	58.73%

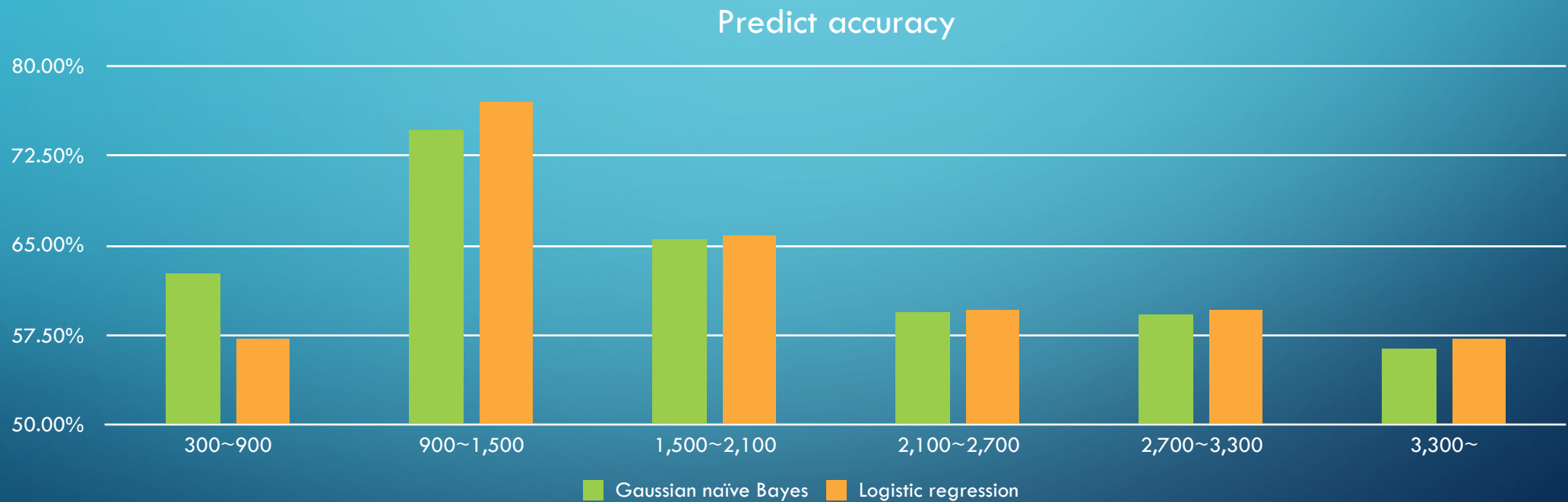
RESULTS (DIFFERENT TOTAL MATCH COUNT)



RESULTS (DIFFERENT DURATION TIME)

Duration time (s)	Gaussian naïve Bayes	Logistic regression
300~900	62.70%	57.14%
900~1,500	74.64%	77.06%
1,500~2,100	65.53%	65.82%
2,100~2,700	59.44%	59.52%
2,700~3,300	59.23%	59.56%
3,300~	56.34%	57.12%

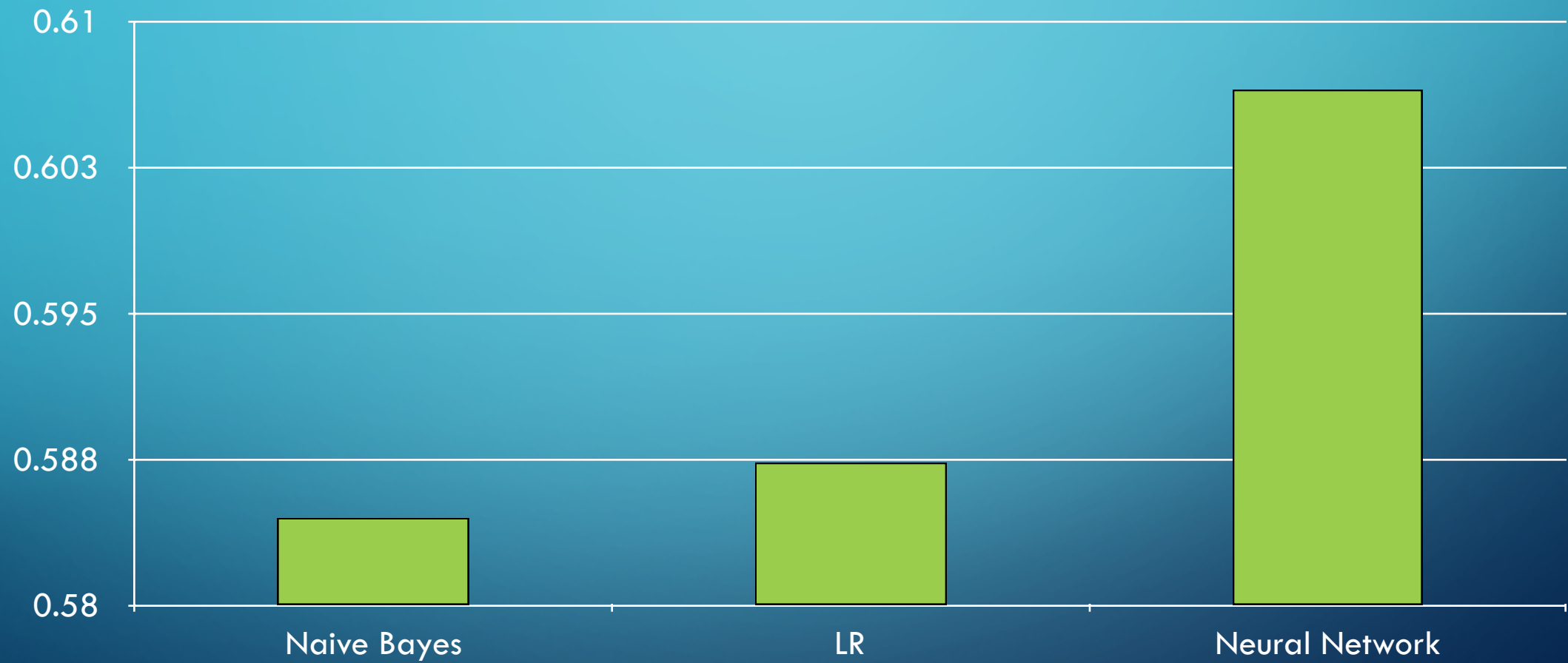
RESULTS (DIFFERENT DURATION TIME)



NEURAL NETWORK MODEL

- Layers: 5 fully connected layers, each of them has 120 hidden nodes
- Activation: ReLu
- Input: 1x120 array with -1, 0, 1
- Output: win or lose (probability of winning and losing condition)

RESULT (NEURAL NETWORK)



GUI (SAVE AND LOAD MODELS)

```
from sklearn.linear_model import LogisticRegression as CLF
clf = CLF(n_jobs=-1).fit(XM, yM)
from sklearn.externals import joblib
joblib.dump(clf, 'lr.model')
```

ab.model	31 KB
abdt.model	27,736 KB
dt.model	21,722 KB
lr.model	2 KB
nb.model	5 KB
perceptron.model	2 KB
rf.model	251,559 KB
svc.model	2 KB

GUI (GIVING PREDICTION RESULT)

Aly heroes		Enemy heroes	Suggest
1, 敌法师, Anti-Mage		41, 虚空假面, Lancelot Void	
11, 影魔, Shadow Fiend		74, 祈求者, Invoker	
28, 斯拉达, Slardar		64, 杰奇洛, Jakiro	
30, 巫医, Witch Doctor		69, 末日使者, Doom	
32, 力丸, Riki		101, 天怒法师, Skywrath Mage	
	Predict	Win rate: 0.44428733295674183	

GUI (SUGGESTING HEROES)

Aly heroes	Enemy heroes	
37, 术士, Warlock	41, 虚空假面, Faceless Void	22, 宙斯, Zeus with win rate: 0.5299123893388075
11, 影魔, Shadow Fiend	74, 祈求者, Invoker	108, 孽主, Underlord with win rate: 0.5159154270781423
28, 斯拉达, Slardar	64, 杰奇洛, Jekino	57, 全能骑士, Omninknight with win rate: 0.5106158253041779
32, 力丸, Riki	69, 末日使者, Doom	42, 冥魂大帝, Wraith King with win rate: 0.5102783152273803
0, 未选择, None	101, 天怒法师, Skywrath Mage	20, 复仇之魂, Vengeful Spirit with win rate: 0.5097771601883323
Predict	Win rate: 0.460054039565024	

The background is a blue gradient. In the corners, there are decorative white lines resembling circuit traces or a stylized tree structure, with small circles at the end of the lines.

THANK YOU VERY MUCH!