DOTA2 WINNING TEAM PREDICTION FROM CHOSEN HEROES BY MACHINE LEARNING METHODS

JIN HAICHEN 16T2006

CONTENT

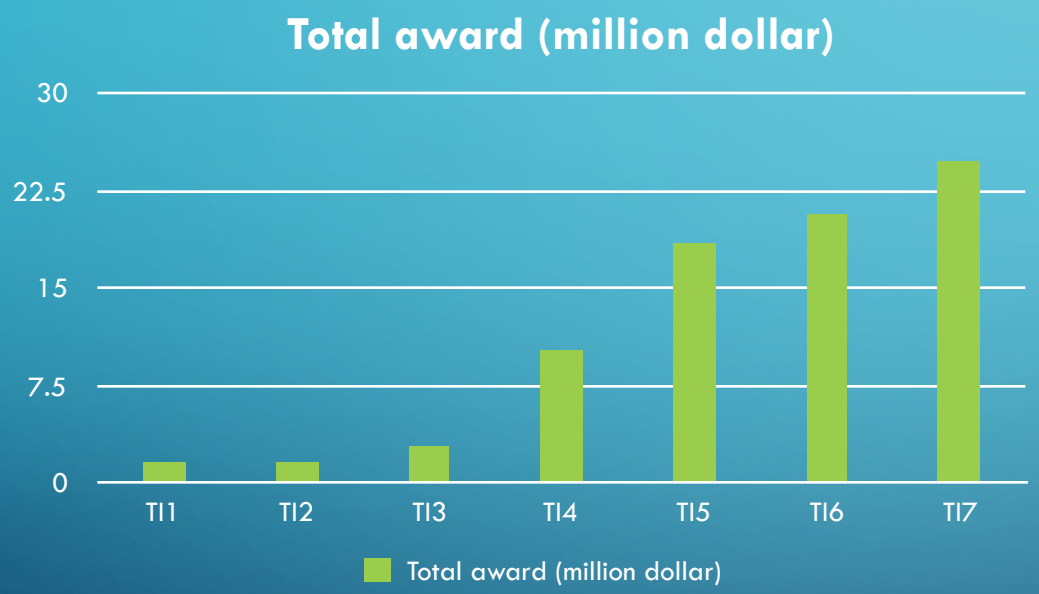
- What is Dota2
- Get data
- Process data
- Models
- Result
- GUI

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)

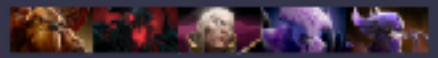
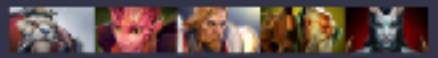
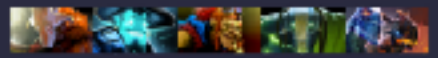
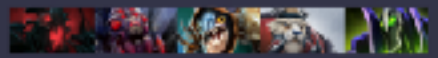
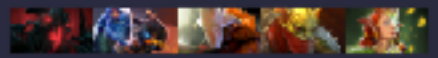
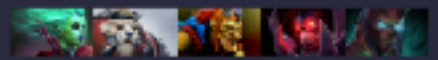
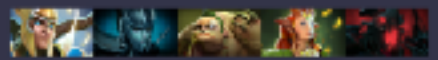
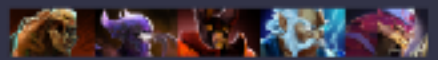
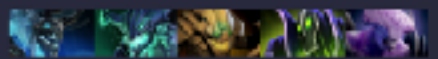


WHAT IS DOTA2 (GAME CONTENT)

- Heroes choosing

Gameplay

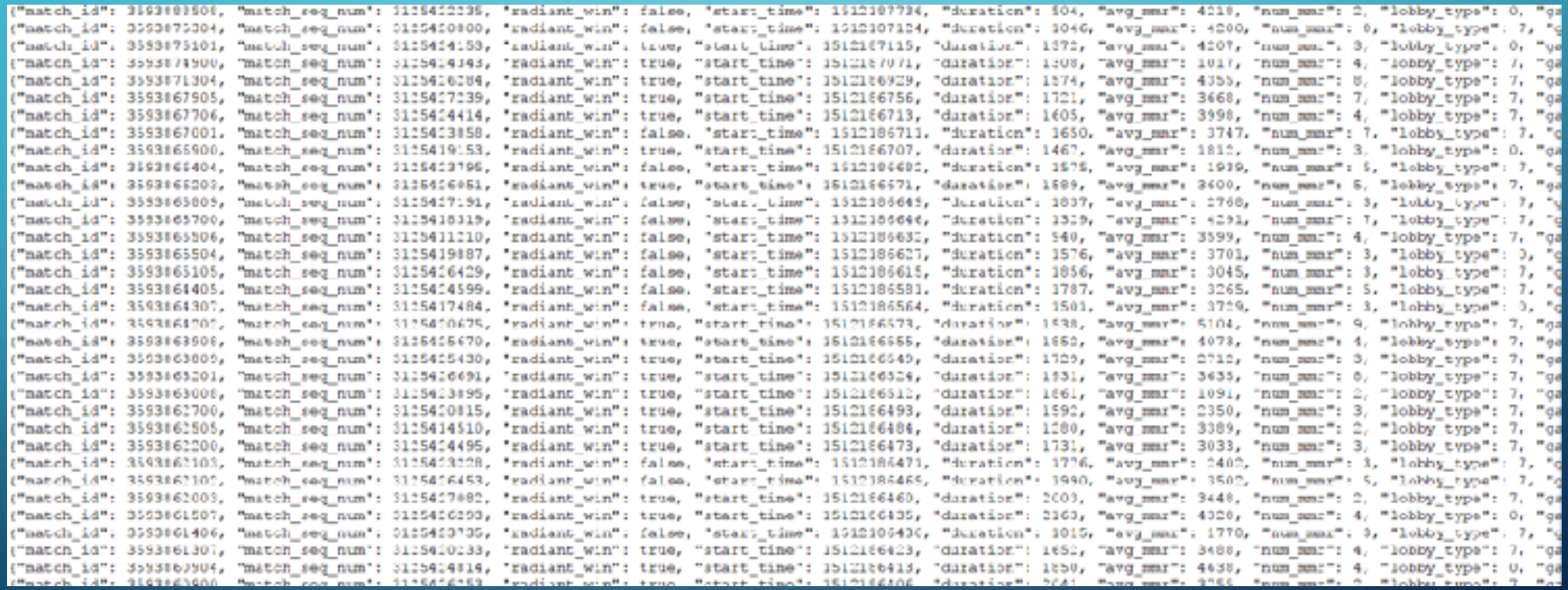
DATA SOURCE (OPENDATA)

ID	DURATION	RADIANT	DIRE
3667358001 › Divine [5]	33:04 2 days ago		
3667371701 › Divine [5]	23:57 2 days ago		
3667372609 › Divine [5]	43:26 2 days ago		
3667379401 › Divine [5]	23:39 2 days ago		
3667383706 › Divine [5]	43:55 2 days ago		
3667386700 › Divine [5]	36:59 2 days ago		
3667421602 › Divine [5]	44:16 2 days ago		
3667426104 › 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 at 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 = '?less_than_match_id=' + str(less_than_match_id)
            matches.extend(response_json)
            print(len(matches), end=' ')
    return matches
```

GET DATA (10 MILLION MATCHES)



GET DATA (MATCHES)

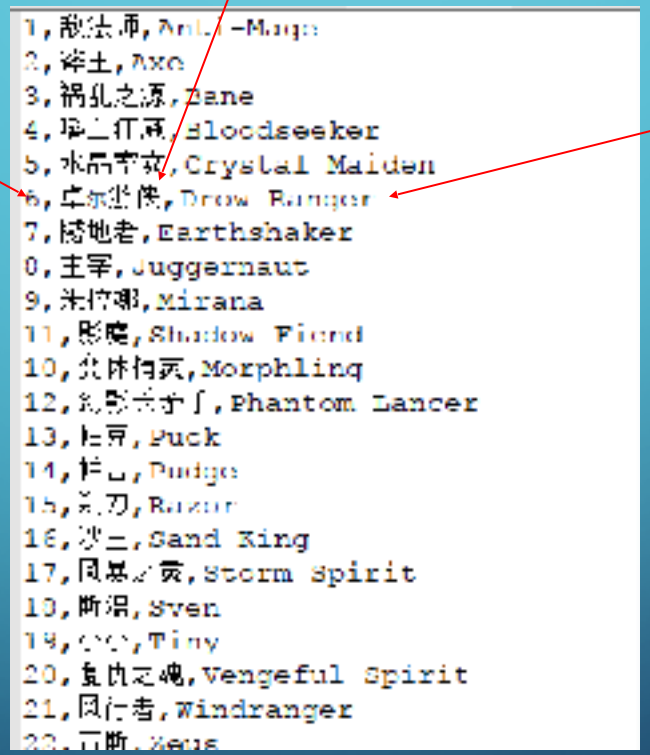
- {
  - …
  - "radiant_win": false, ← winning team (radiant or dire)
  - "duration": 504, ← duration time (seconds)
  - "avg_mmr": 4218, ← average match rating (player levels)
  - "radiant_team": "64,97,104,93,51", ← chosen heroes id
  - “dire_team”: “82,14,1,42,35”
- }

GET DATA (HEROES)

Hero's Chinese name

Hero id

Hero's name



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
```

GET DATA (PROCESS DATA)

```
def process_data(matches, duration_min=0, duration_max=3600, mmr_min=0, mmr_max=10000):
    teams_win = []
    matches_heroes = []
    for match in matches:
        duration = match['duration']
        mmr = match['avg_mmr']
        if isinstance(duration, int) is False or isinstance(mmr, int) is False:
            continue
        if duration < duration_min or duration > duration_max:
            continue
        if mmr > mmr_max or mmr < mmr_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
- [0, 1, 0, 0, -1, 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, -1, 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, 0, -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, 1, 0, 0, 0, 0, 0, 0]

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 -1means team loses

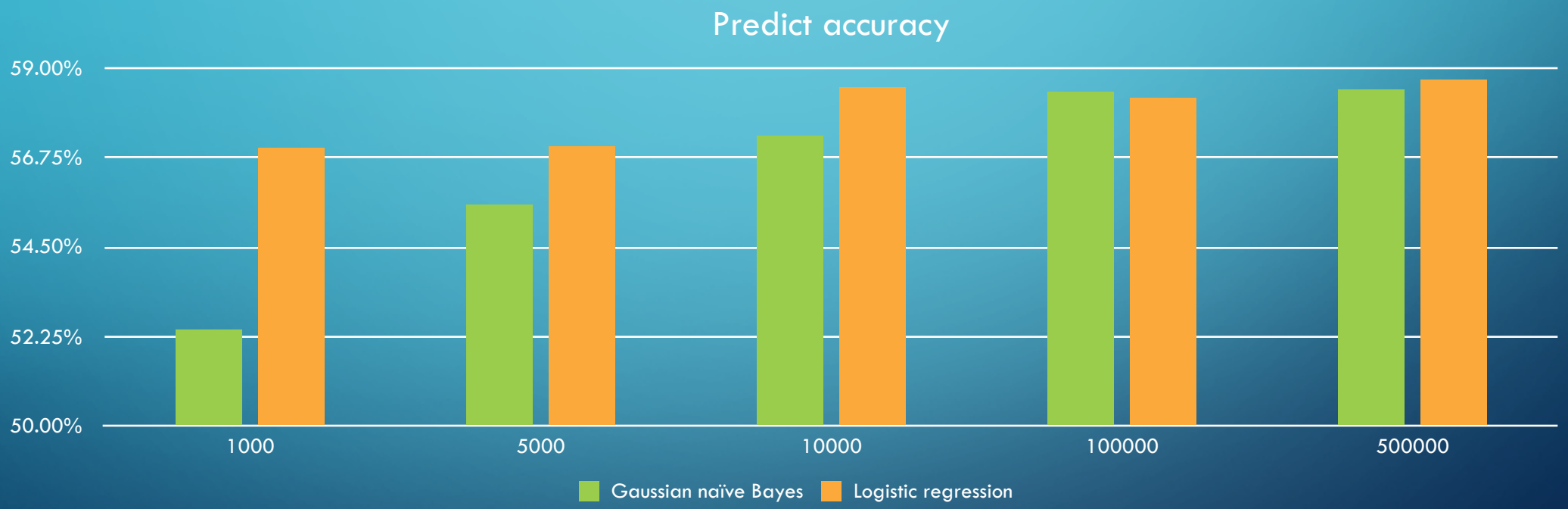
RESULTS (DIFFERENT ALGORITHMS)

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%

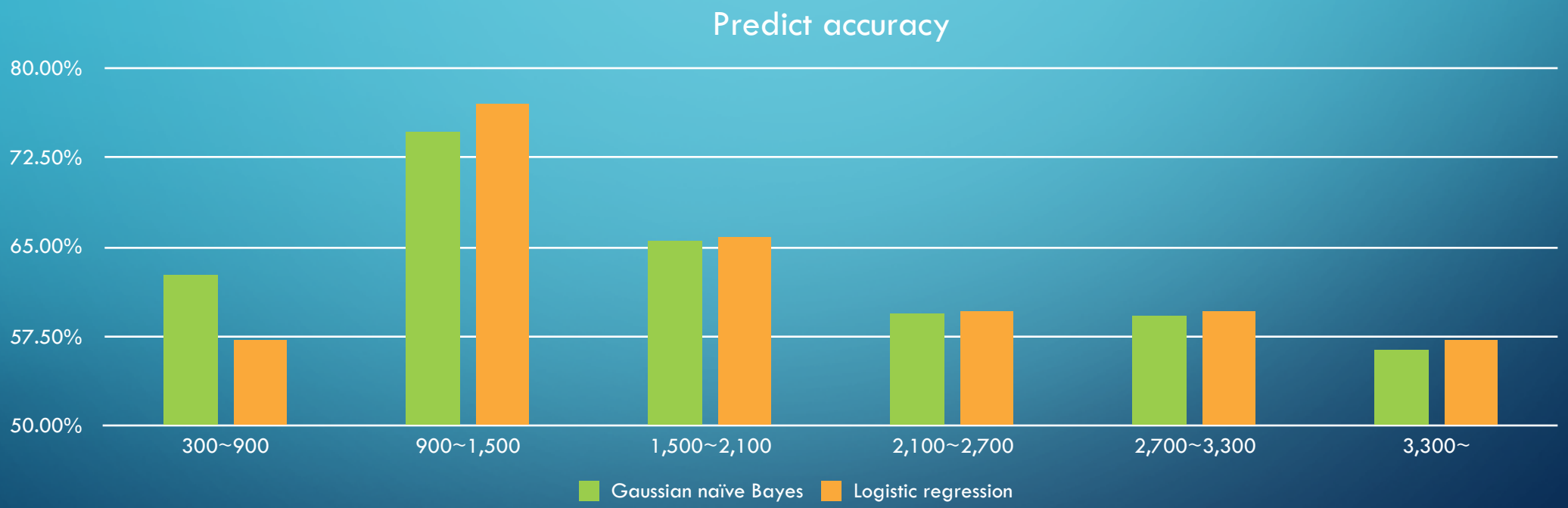
RESULTS (DIFFERENT TOTAL MATCH COUNT)



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%

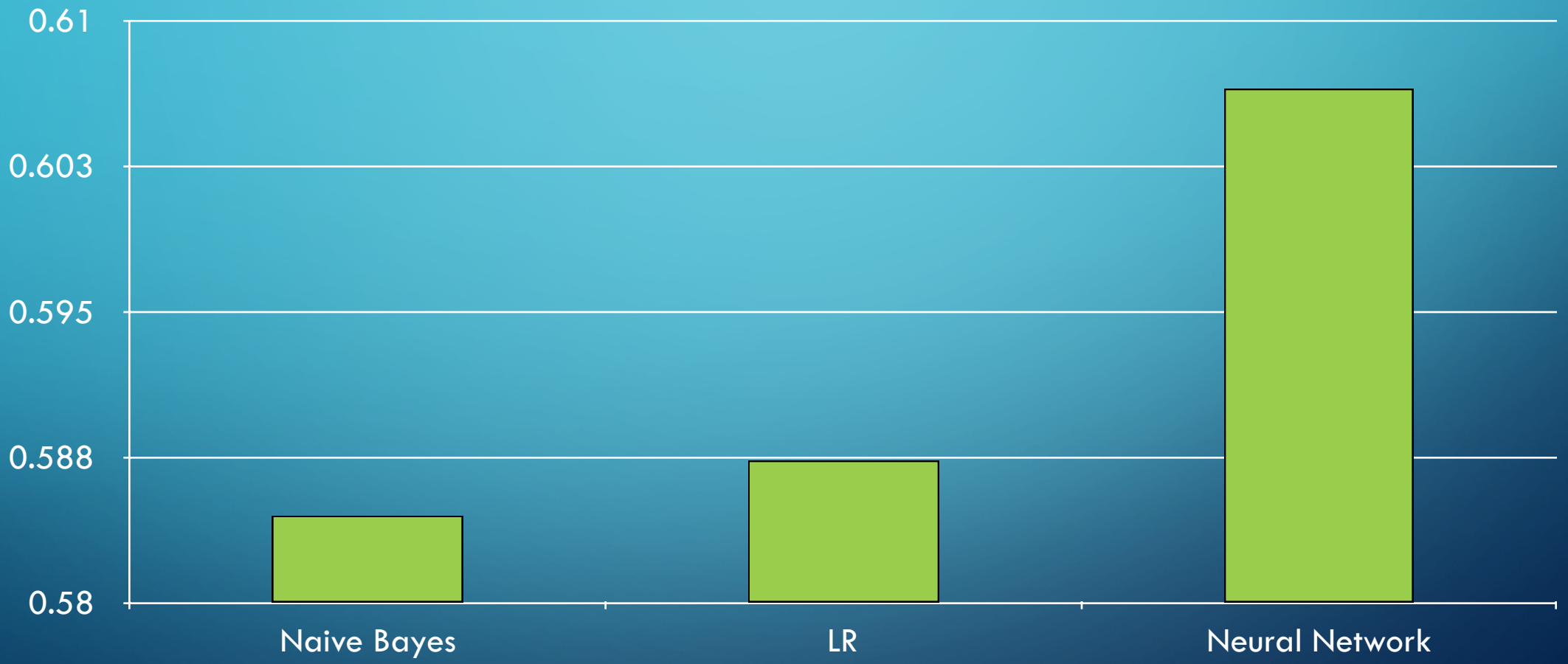
RESULTS (DIFFERENT DURATION TIME)



NEURAL NETWORK MODEL

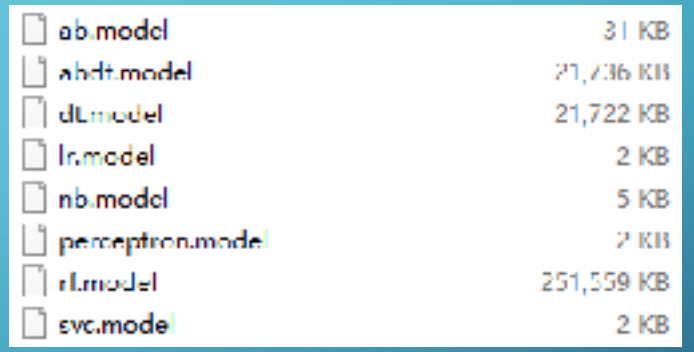
- Layers: 5 fully connected layers, each of them has 120 hidden nodes
- Activation: ReLu
- Input: 1x120 array with -1, 0, 1
- Output: win or lose (probability of winning and losing condition)

RESULT (NEURAL NETWORK)

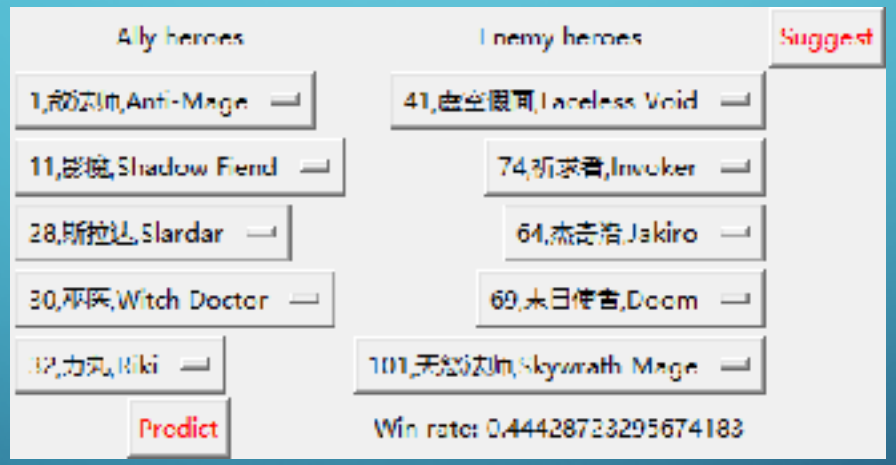


GUI (SAVE AND LOAD MODELS)

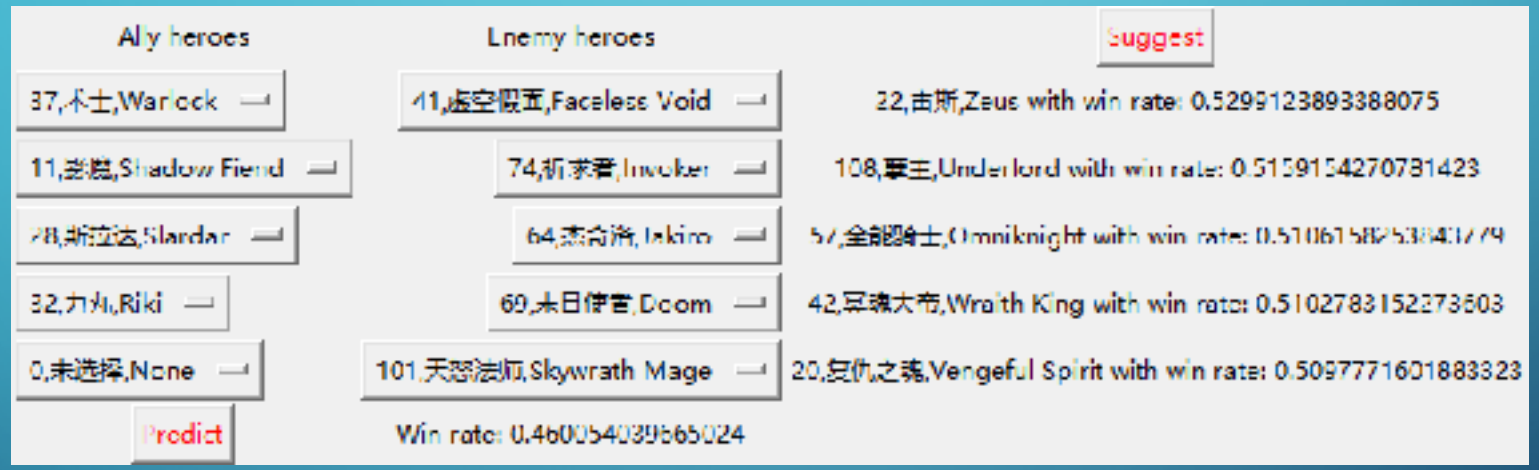
```
from sklearn.linear_model import  ogisticregression as CLF
clf = CLF(n_jobs=-1).fit(xM, yY)
from sklearn.externals import joblib
joblib.dump(clf, 'lr.model')
```



GUI (GIVING PREDICTION RESULT)



GUI (SUGGESTING HEROES)



THANK YOU VERY MUCH!