Predicting Horse Racing Result Using Tensorflow

Motivation

- Horse racing is different from games in a casino
- Publicly attended event -> no one controlling
- Dependent event
- Historical data are opened to public
- Predict horse racing result based on historical data
- Is it possible to generate profit?

Introduction

- Background information and terminology of horse racing
- Data collection, storage, extraction
- Simply data analysis
- Possible ways to model the problem
- Data preprocess and normalization
- Result from different learning algorithm
- Limitation and difficulties
- Conclusion and short demo

Background

- A sport that running horses at speed
- 8-14 horses in a race
- The fastest the winner
- Hong Kong Jockey Club (HKJC) providing pari-mutuel betting on horse racing
- Pari-mutuel betting is a betting system in which all bets of a particular type are placed together in a pool and taxes are removed, and payoff odds are calculated by sharing the pool among all winning bets
- There are many types of bet, we only focus on win bet

Data Collection

- ❌ Buy historical data from data company
- Expensive!!!

- ✓ Web crawling
- Collected 15-year historical data

Feature	Description
Date	-
Location	-
Race Number	-
Class	-
Distance	-
Going	Track condition
Course	Track
Pool	Prize pool
Place	-
Horse ID	-
Horse	-
Jockey	-
Trainer	-
Actual Weight	Carried weight
Declare Weight	Overall weight
Draw	-
LBW	Length behind winner
Running Position	-
Time	Finishing time
Win Odds	Closing odds

Data Storage

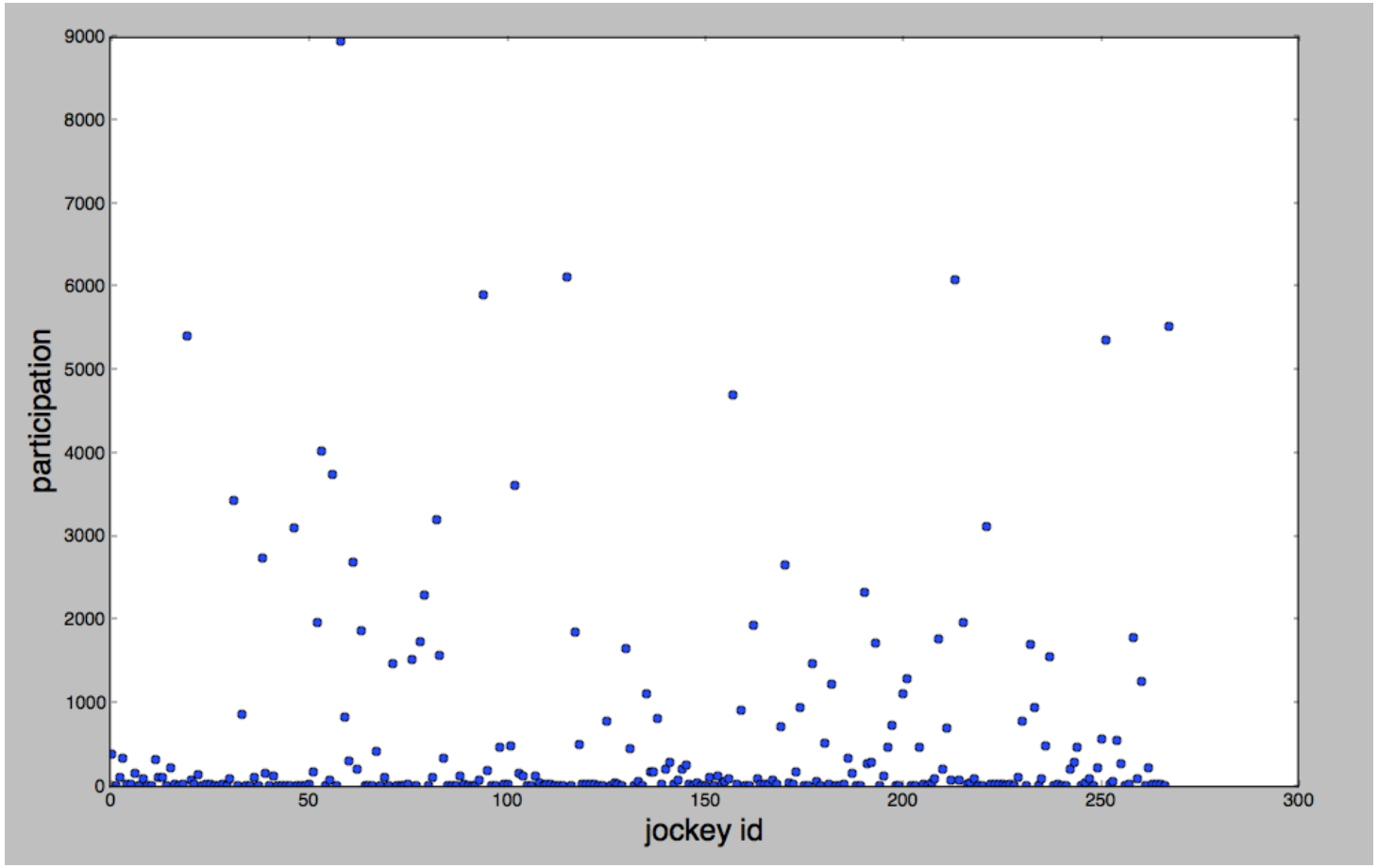
- Relational Database Management System
- Postgres vs MySQL
- Postgres has more useful built-in functions

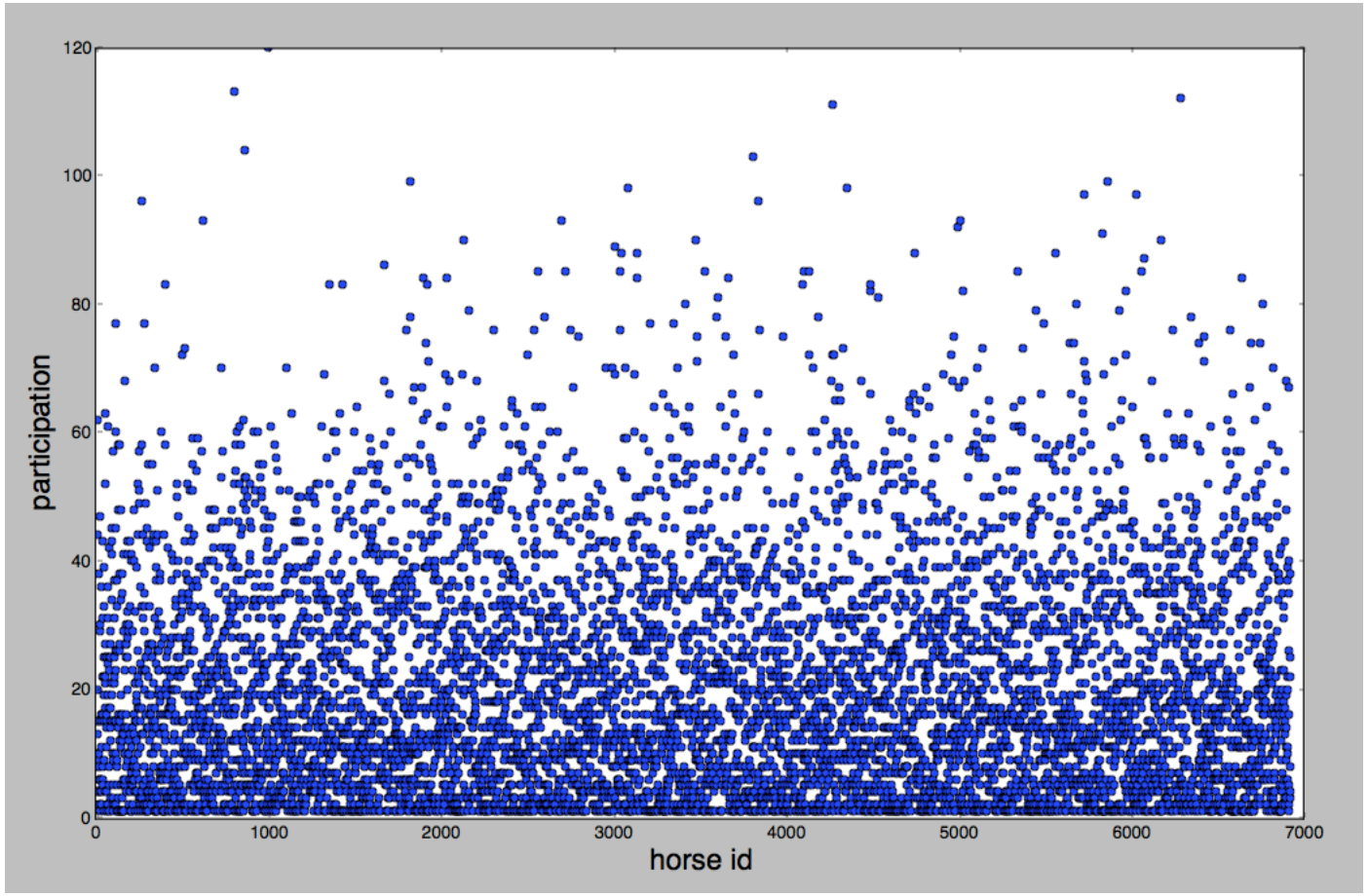
- GUI Software (Postico)
- Good for visualization

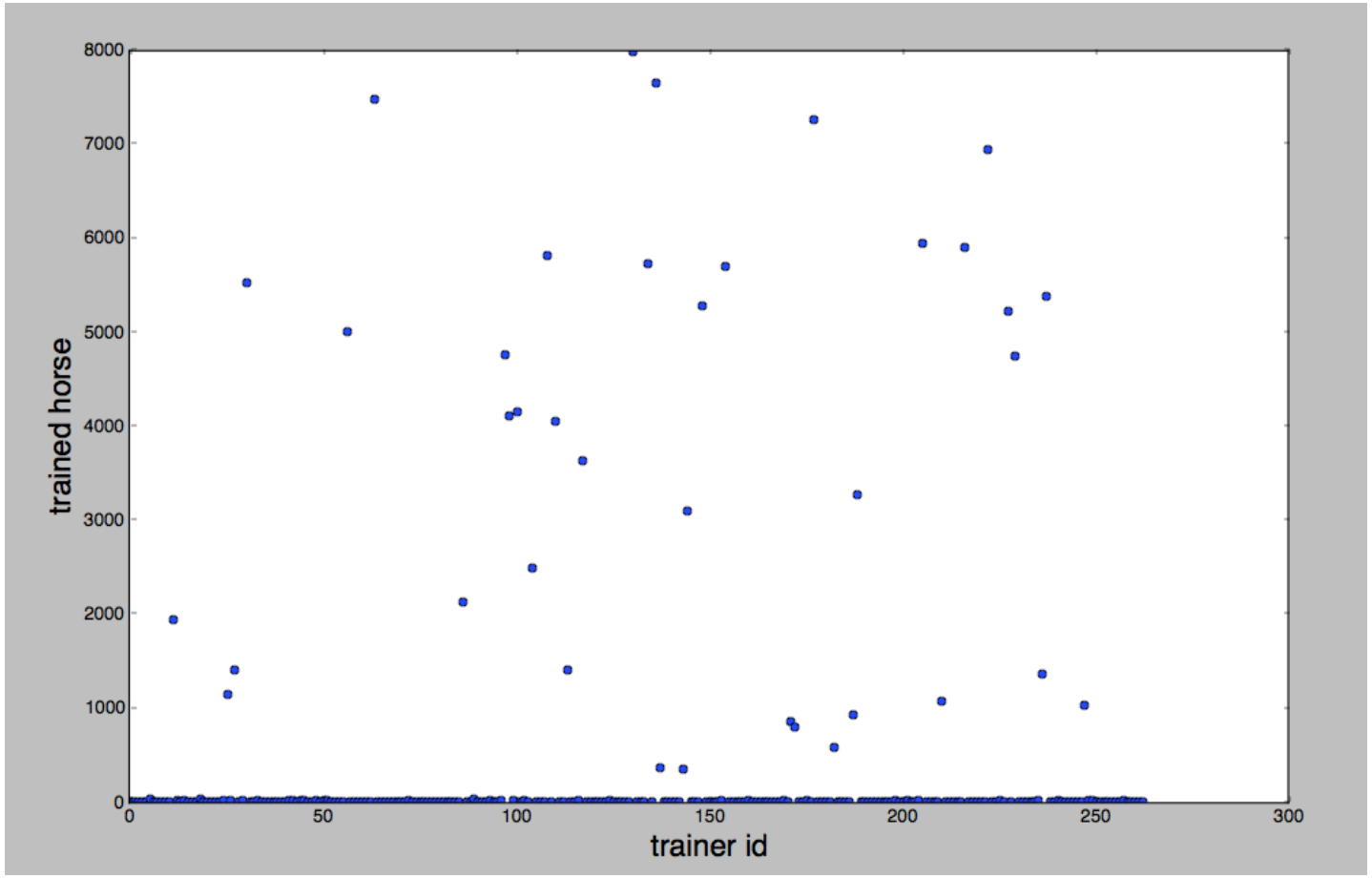
Extract more data

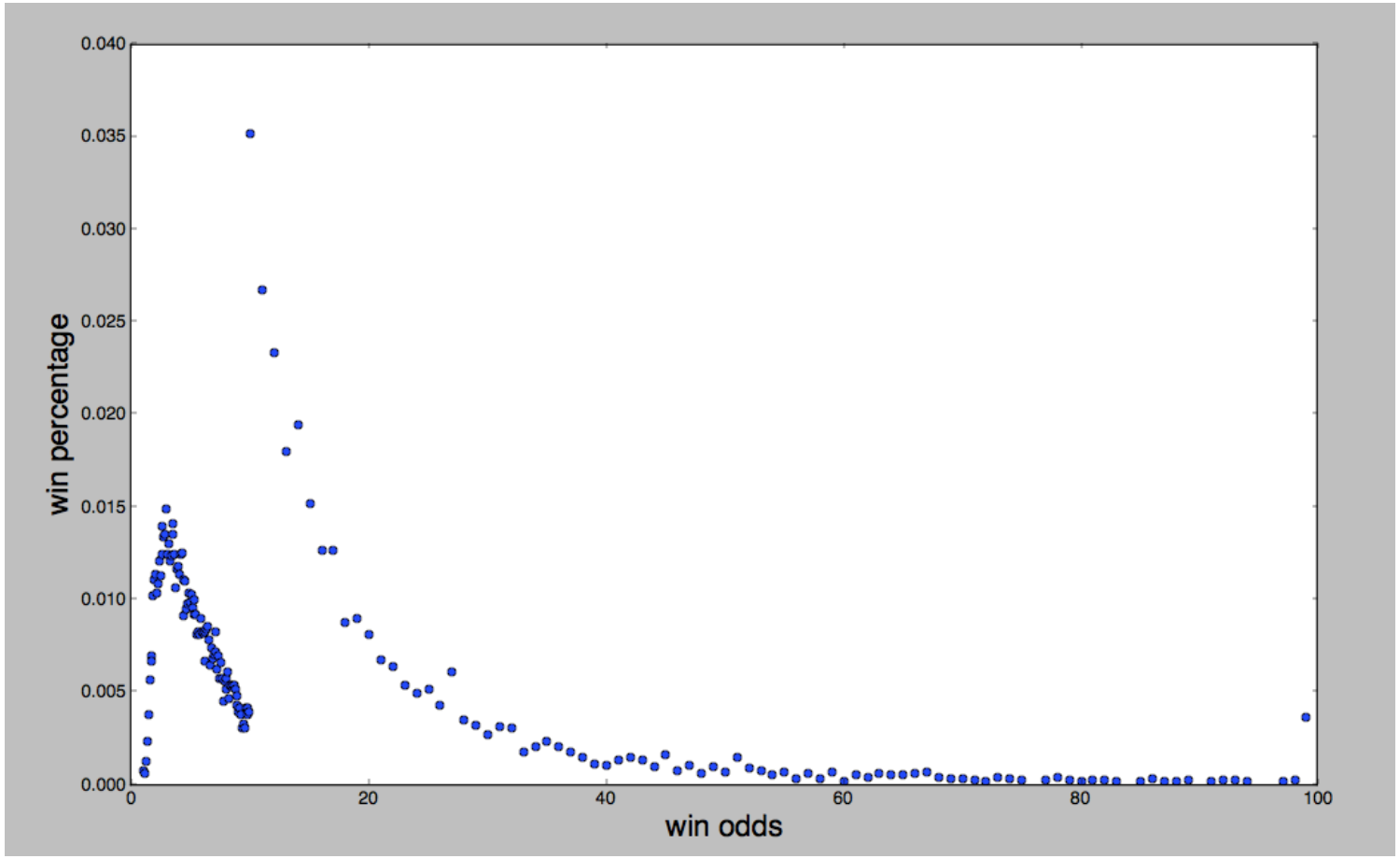
- Age
- Time since last race
- Weight different from last race
- Past performance on the same track
- Jockey win rate
- Trainer win rate
- Horse win rate

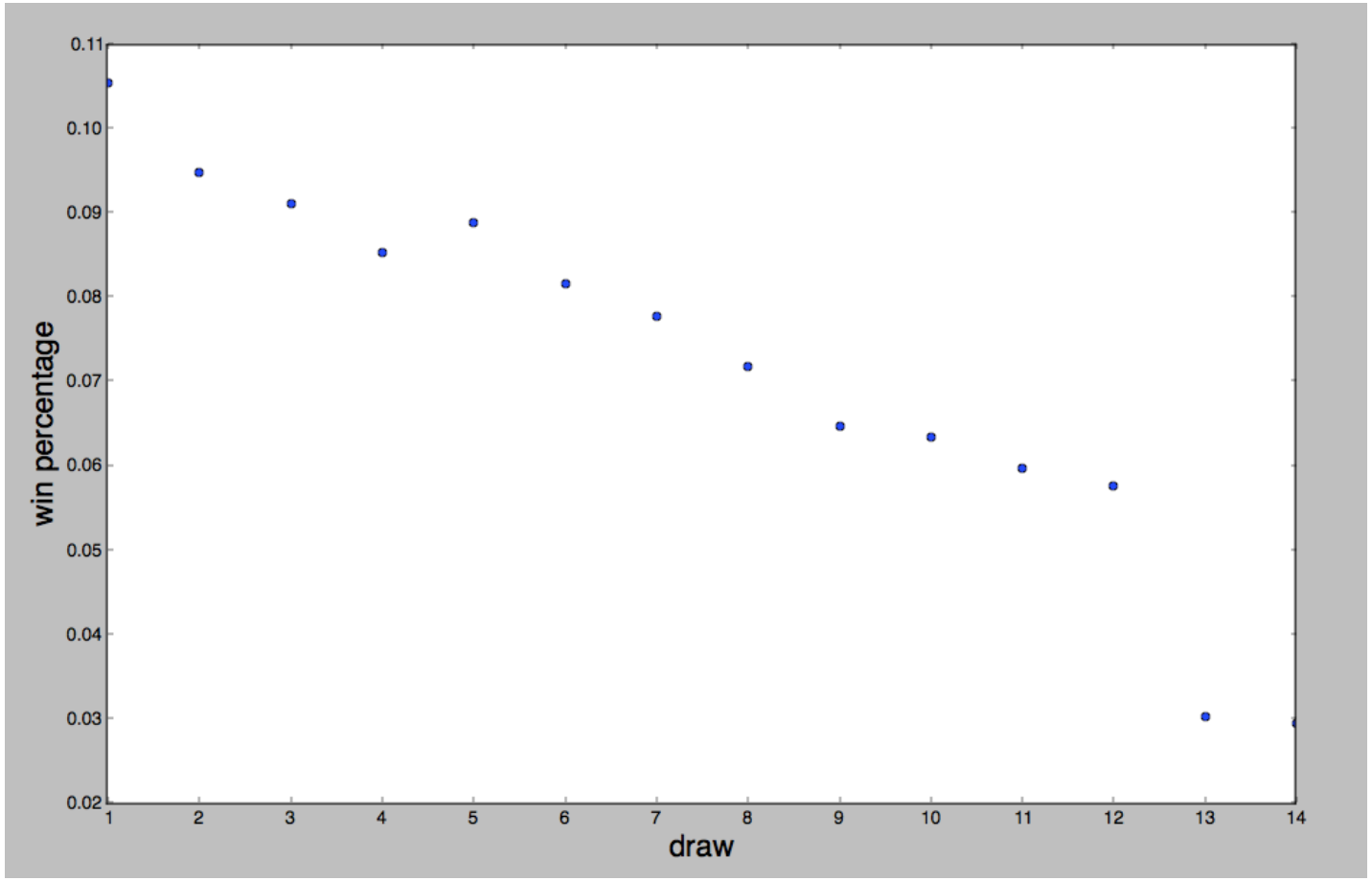
Data analysis

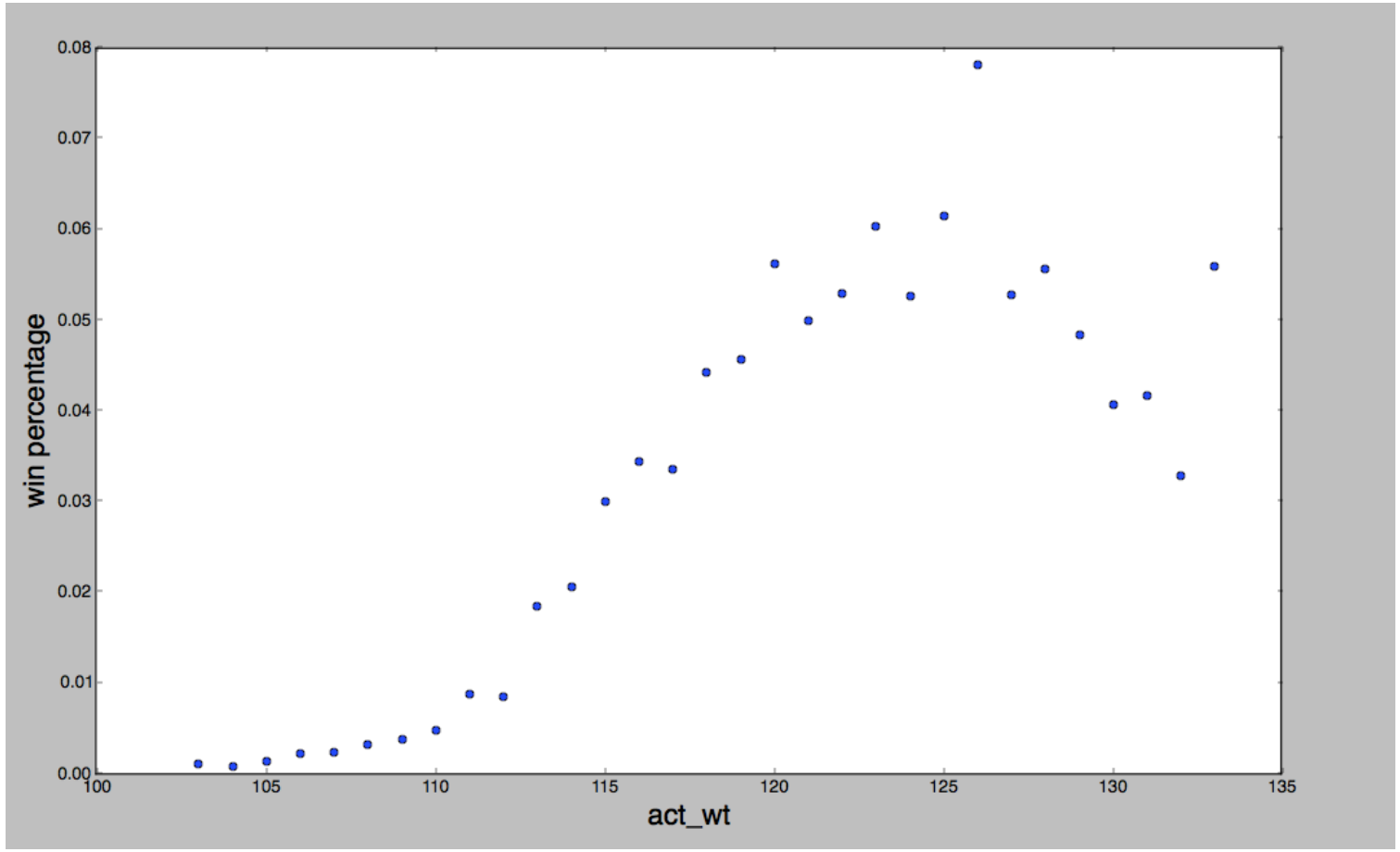


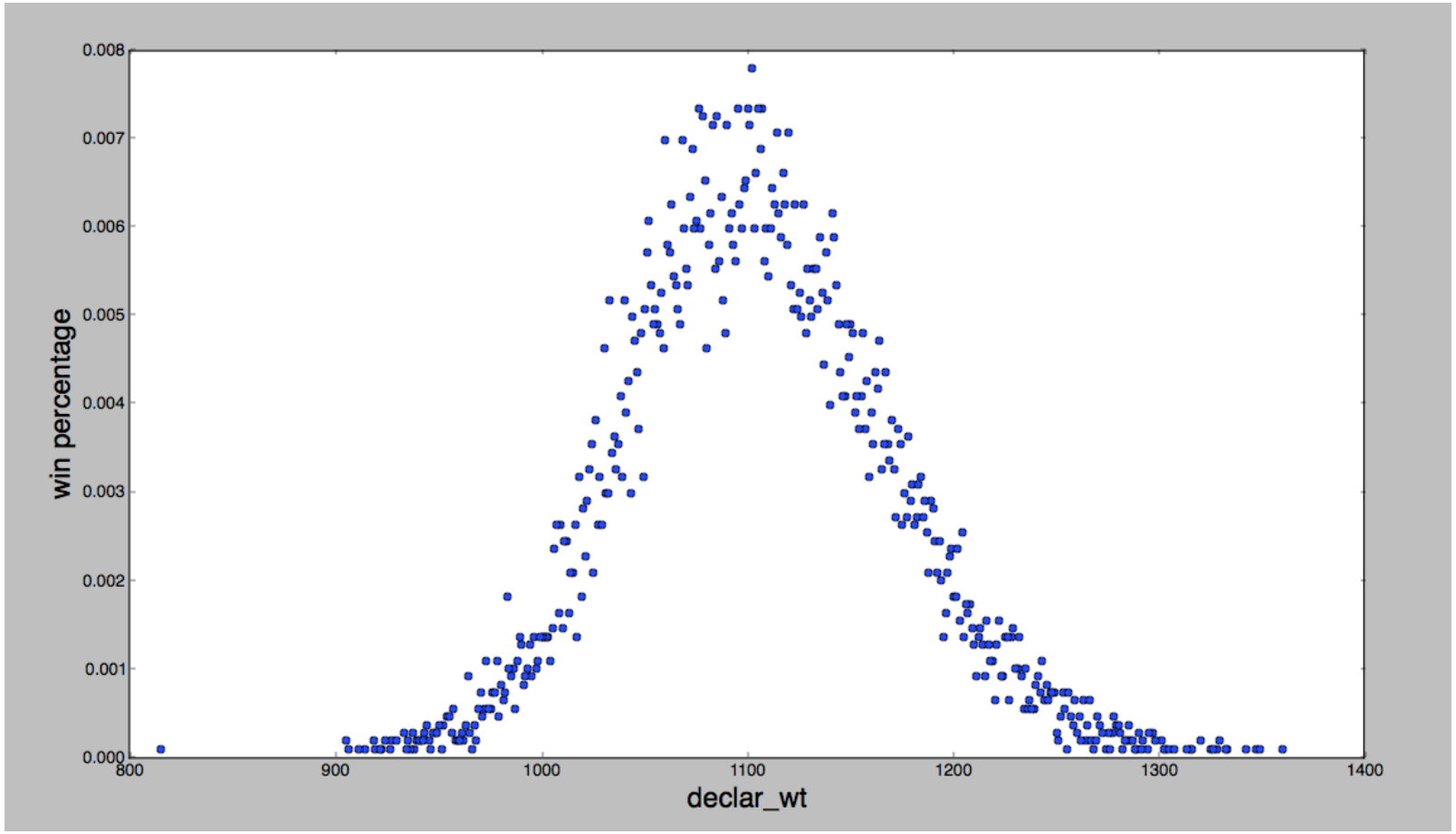

participation
horse id
0
20
40
60
80
100
120
0
1000
2000
3000
4000
5000
6000
7000

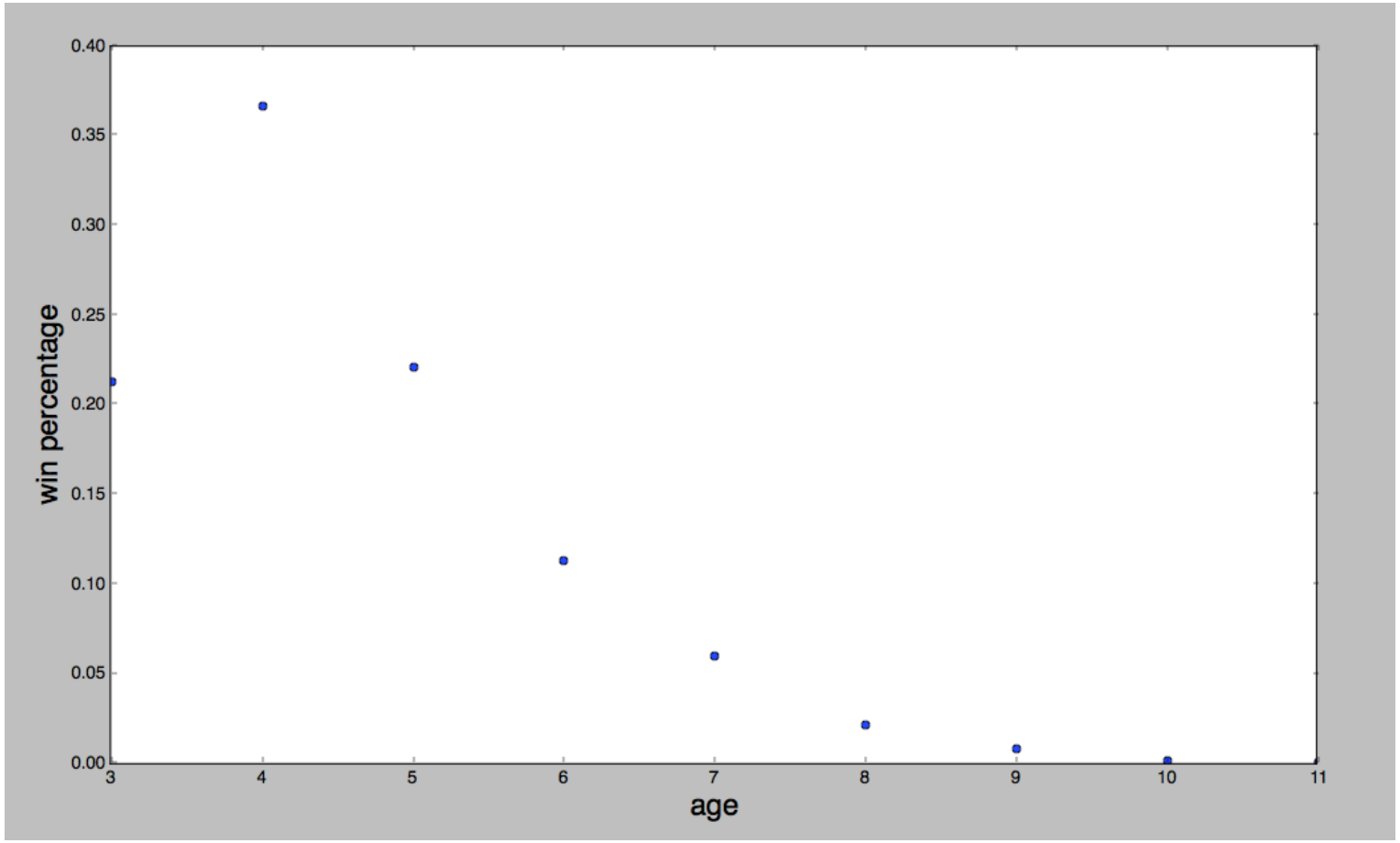

trained horse
trainer id
8000
7000
6000
5000
4000
3000
2000
1000
0
0
50
100
150
200
250
300

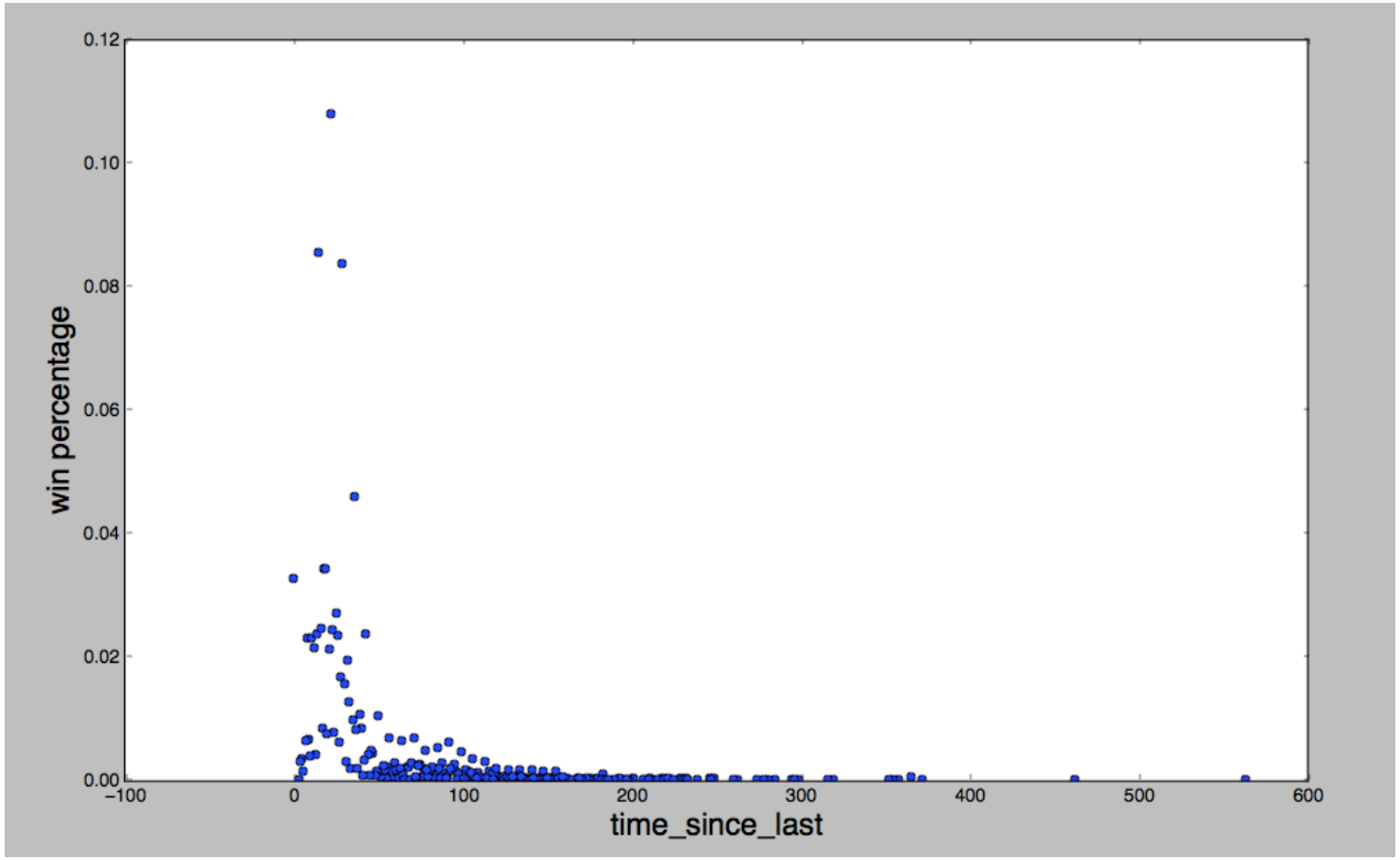

win percentage
win odds
0.040
0.035
0.030
0.025
0.020
0.015
0.010
0.005
0.000
0
20
40
60
80
100

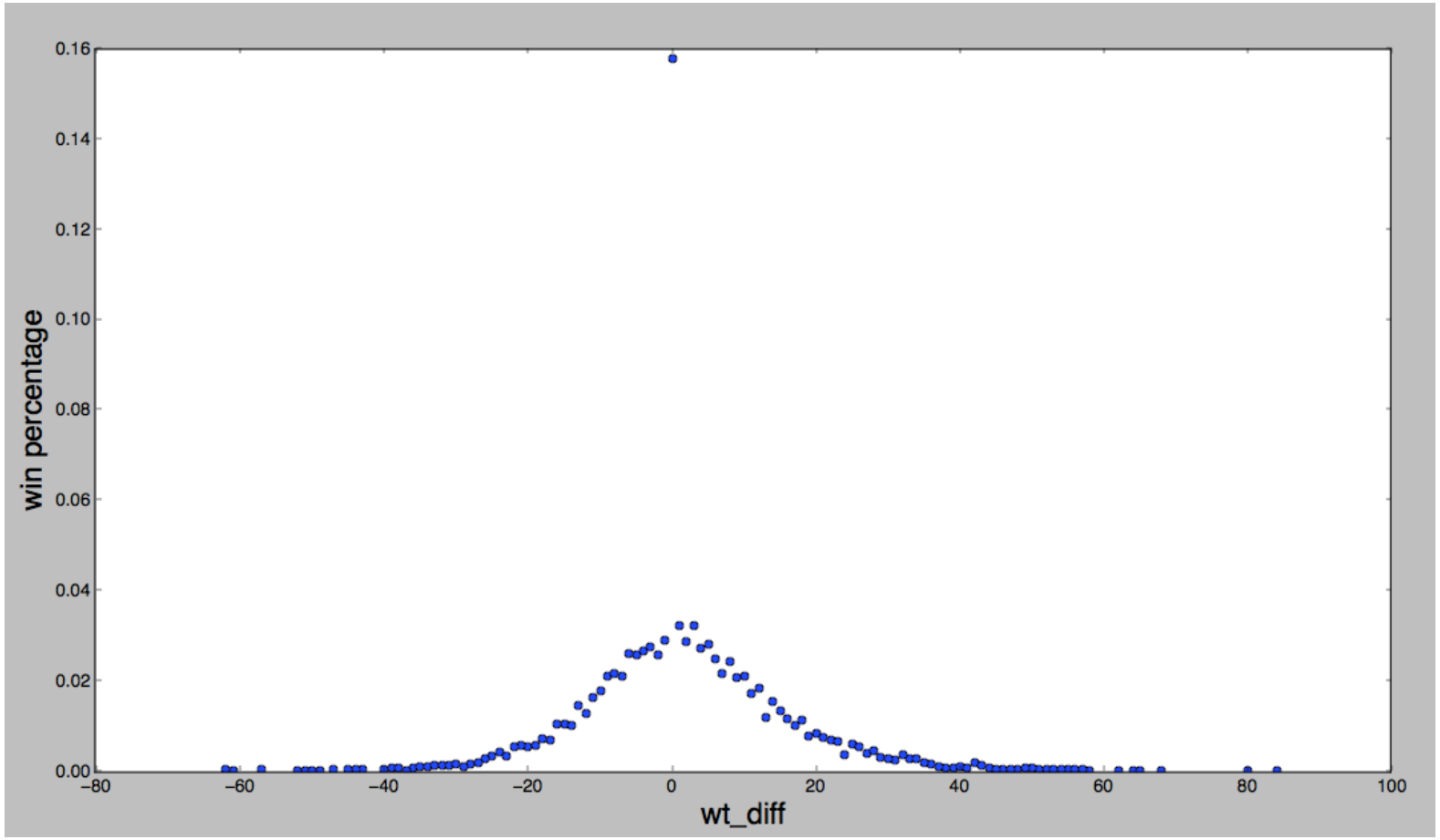

win percentage
draw
0.11
0.10
0.09
0.08
0.07
0.06
0.05
0.04
0.03
0.02
1
2
3
4
5
6
7
8
9
10
11
12
13
14

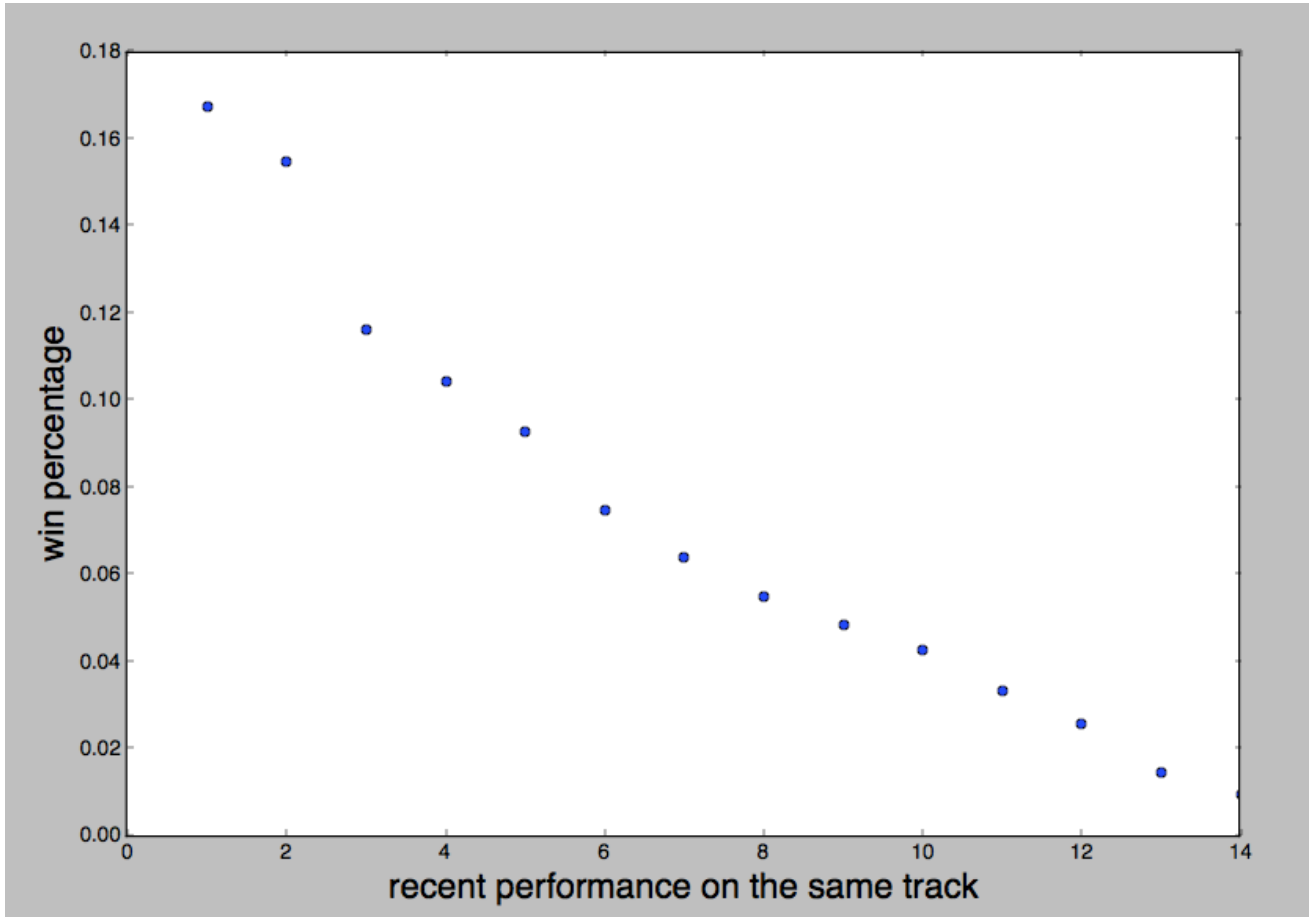

win percentage
act_wt
0.00
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
100
105
110
115
120
125
130
135


win percentage
declar_wt
0.008
0.007
0.006
0.005
0.004
0.003
0.002
0.001
0.000
800
900
1000
1100
1200
1300
1400


win percentage
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0.05
0.00
3
4
5
6
7
8
9
10
11
age


win percentage
time_since_last
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−100
0
100
200
300
400
500
600

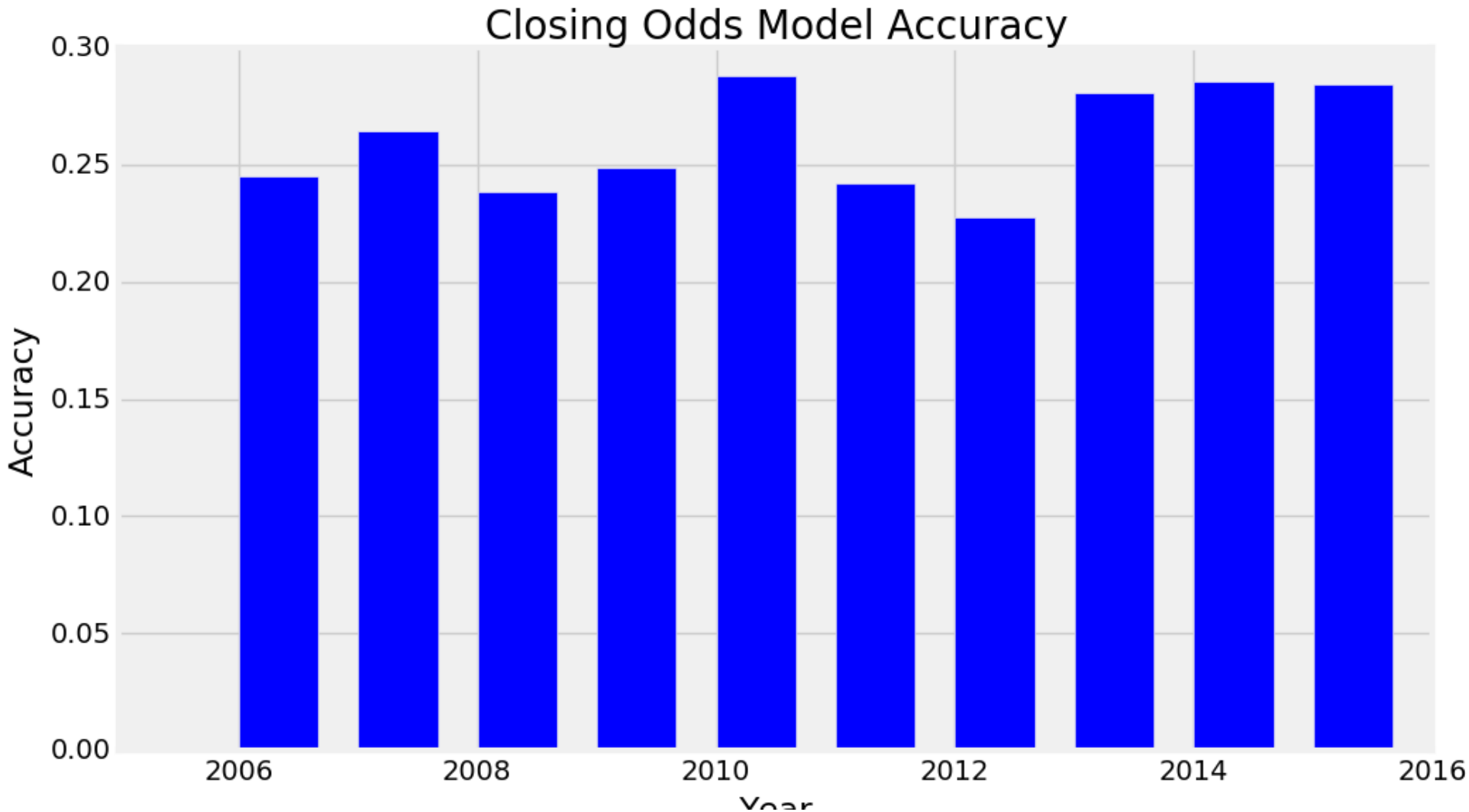

win percentage
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
−80
−60
−40
−20
0
20
40
60
80
100
wt_diff


win percentage
0.18
0.16
0.14
0.12
0.10
0.08
0.06
0.04
0.02
0.00
0
2
4
6
8
10
12
14
recent performance on the same track

Public Intelligence

- Odds are driven by public
- Lower odds mean higher expectation
- Using final win odds for prediction
- Prediction:
  - Choose the horse with the lowest odds

Public Intelligence



ELO Rating System

- $R'_x = R_x + K(S_x - E_x)$
- $E_x = \frac{\sum_{1<i<N, i\neq x} \frac{1}{1+10^{R_i - R_x}}}{\frac{N(N-1)}{2}}$
- $S_p = \frac{N-p}{\frac{N(N-1)}{2}}\ for\ 1 \leq p \leq 14$
- $S_p = \frac{N-(p+0.5)}{\frac{N(N-1)}{2}}, for\ p\ is\ double\ head, 1 \leq p \leq 13$

ELO Rating System

We have set the initial ELO to 1500,
then we followed the equations above to compute ELO for jockeys, horses and trainers

First pick a random K, then binary search to find a good K

	K	Win percentage
Horse elo	6000	0.1697
Jockey elo	50	0.1717
Trainer elo	50	0.1356

Possible ways to model the problem

- Strength of a horse
- Probability of a horse to win the race
- Finishing Time
- Which horse will win in a race?

- We decided to model the problem as the probability of a horse to win the race.

Data preprocess and normalization

- Remove records with empty data
- Real value data -> subtract min/(max-min)
- Categorical data -> Tensorflow handle for us

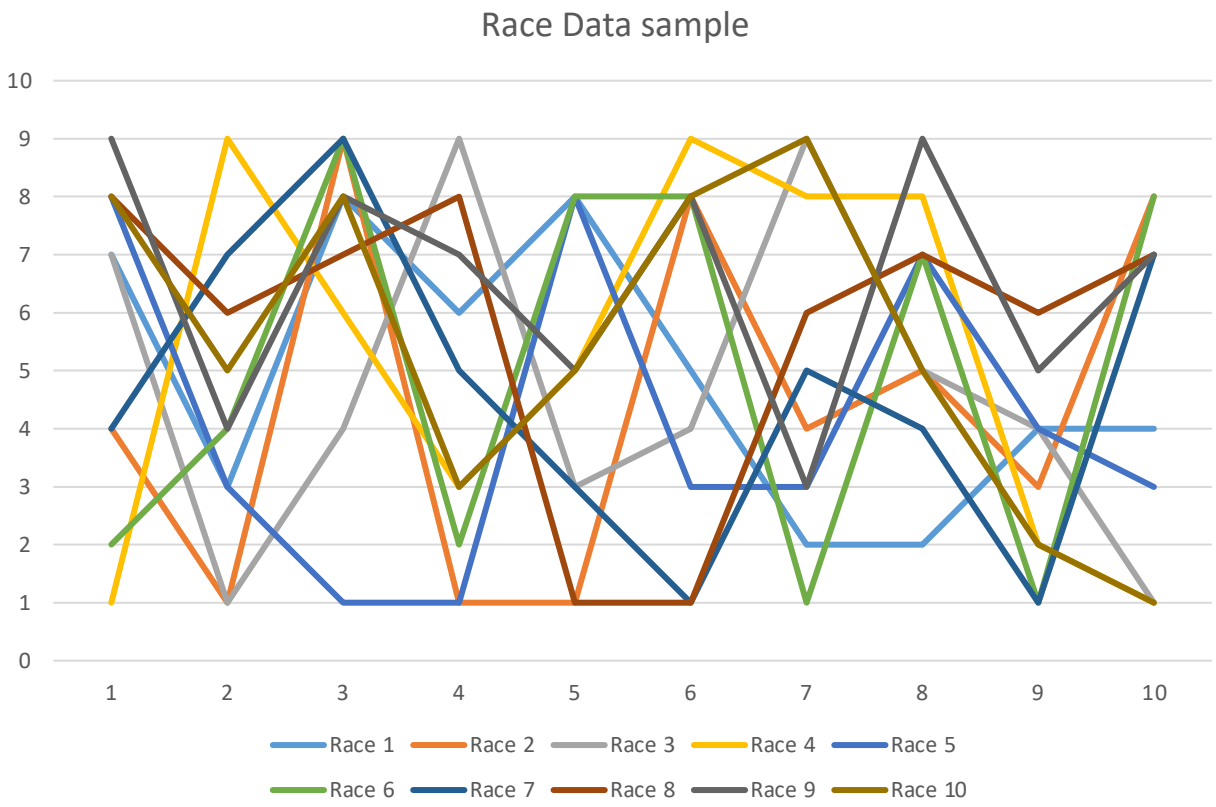
- Crossed categorical data
  - Draw x location x course

Model Training

- Pattern Matching (Not Tensorflow)
- Linear Model
- Deep Neural Network

Pattern Matching

- Find k similar races for prediction
- $R = \{h_i, j_i, t_i\}\ for\ 1 \leq i \leq n$
- $Index = \{R_i\}\ for\ 1 \leq i \leq m$
- $similarity(R_i, R_j) = \frac{R_i \cdot R_j}{\left||R_i||R_j|\right|}$



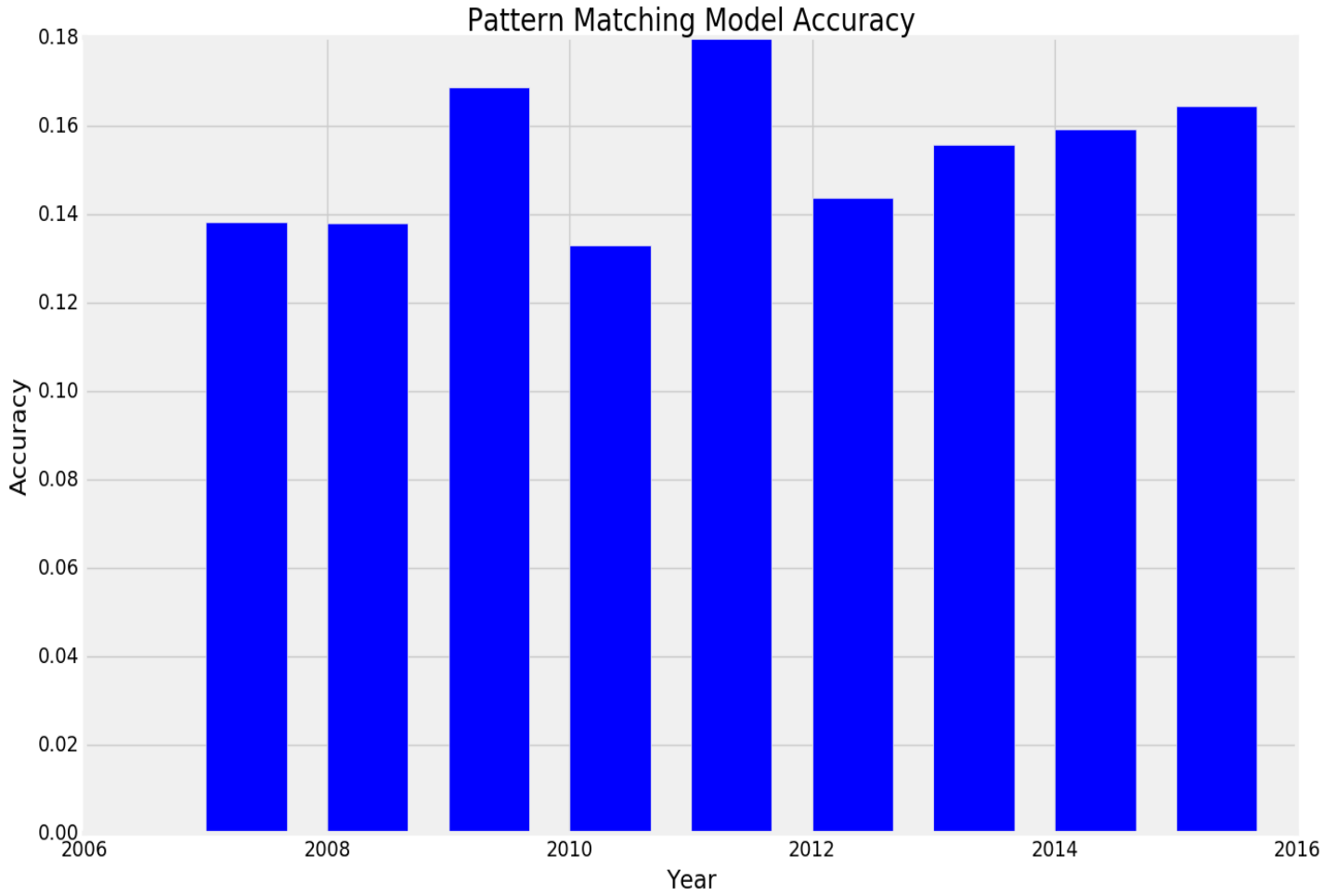
Pattern Matching – Value of k

- k too high
  - Too many irrelevant data
  - Poor prediction result
- k too slow
  - Not enough data for prediction

Number of races	Value of k
~700	<105
~1400	<135
~2100	<254
~2800	<261
~3500	<289

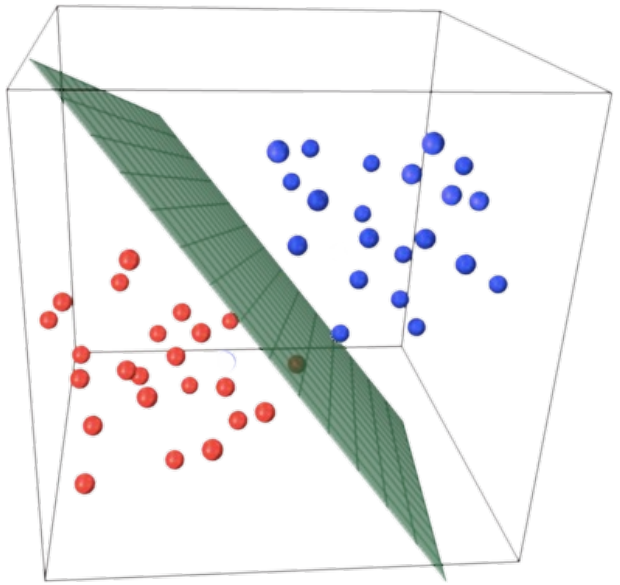
Pattern Matching

- Index's size: ~700 record
- k = 12



Linear Model

- Also called Logistic regression model
- $P(Y = 1|X) = \frac{1}{1+\exp(-(w^T x+b))}$
- Y=1, the horse win; Y=0, the horse not win
- X is a set of features
- We are training w



Linear Model - Problem

- Unbalanced dataset
  - Only one winner in a race
  - Around 10:1 for Y=0:Y=1
- Solutions
  - Duplicating data
  - Assigning weight to record

Linear Model - Train

- Training dataset: 2001-2014
- Testing dataset: 2015-2016
- The way we test the model is to pick the horse with the highest probability to win among horses in a race, the win rate is referring to the number of race the model correctly predicted in 2015-2016

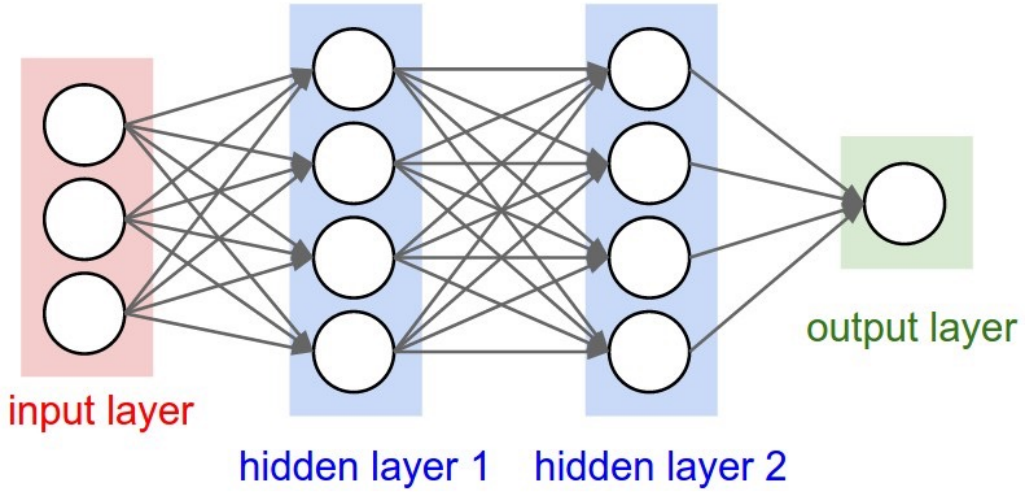
Linear Model - Result

Epoch	Win	Win rate
1000	182	0.232143
2000	196	0.25
3000	200	0.255102
3500	208	0.265306
4000	208	0.265306

Deep Neural Network

- Tensorflow handles the detail
- Number of hidden layer
- Number of node in each layer
- How many epochs to train

- X and Y same as Linear Model



Deep Neural Network - Problem

- DNN don’t accept categorical data
  - Tensorflow provides function to convert categorical column to embedding column
- Unbalanced dataset
  - Same as Linear Model

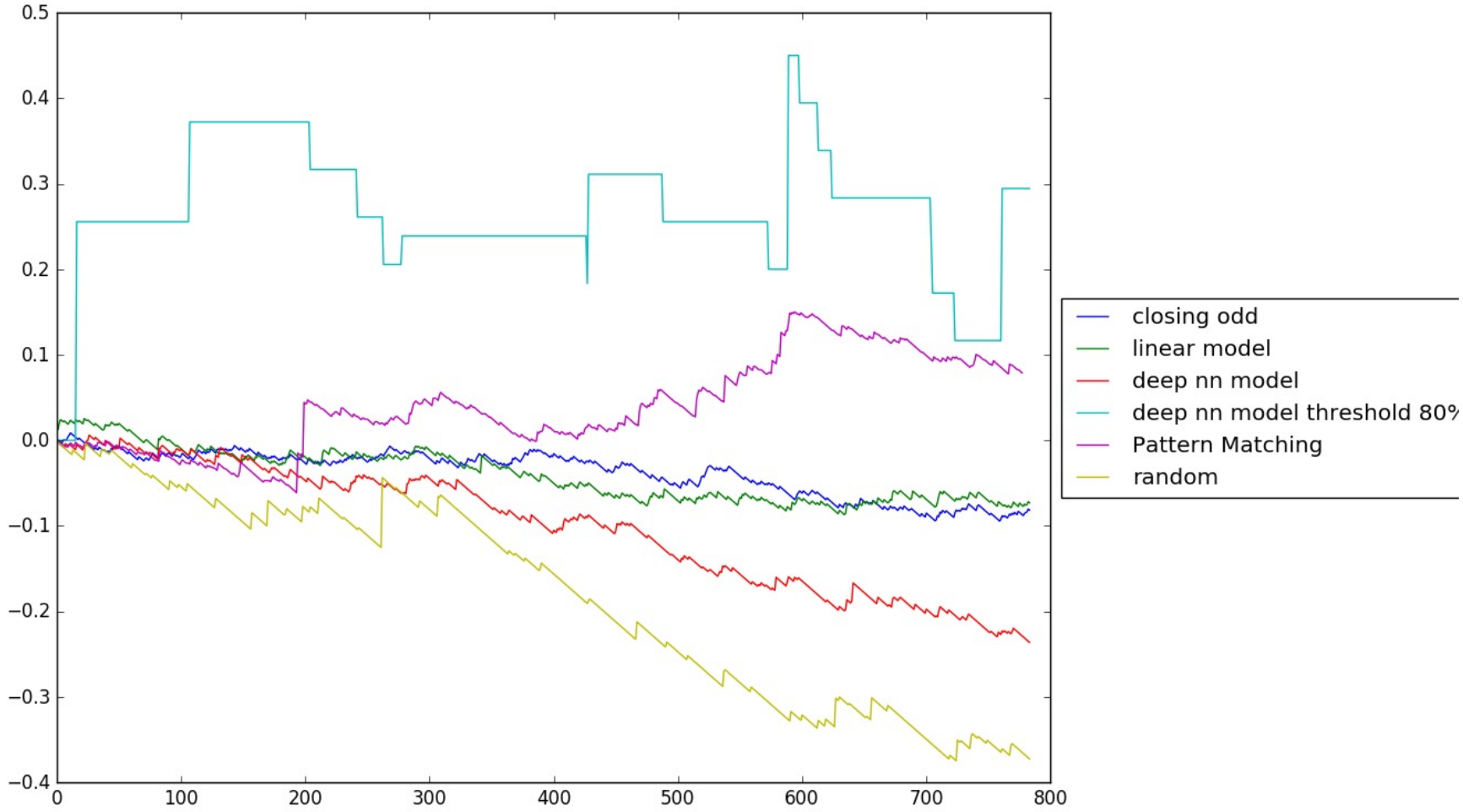
Deep Neural Network - Train

- Training dataset: 2001-2014
- Testing dataset: 2015-2016
- The way we test the model is to pick the horse with the highest probability to win among horses in a race, the win rate is referring to the number of race the model correctly predicted in 2015-2016

Deep Neural Network - Result

Epoch	Hidden Layer	Win	Accuracy
2000	[100,50]	132	0.168367
3000	[100,50]	143	0.182398
4000	[100,50]	151	0.192602
5000	[100,50]	158	0.201531
6000	[100,50]	166	0.211735
7000	[100,50]	168	0.214286
8000	[100,50]	167	0.213010

Models Evaluation



Limitation and difficulties

- Web crawling
- Preparing data
- Lack of data

Future work

- Improve models in terms of accuracy(win rate) and profit earning
- Automate process of updating data from HKJC
- Integral the trained model to an application

Conclusion

- In terms of win rate,

	Beat win odd model
Linear Model	no
Deep Neural Network Model	no
Pattern Matching	no

- In terms of profit earning,

	Beat win odd model	Positive return
Linear Model	Yes	No
Deep Neural Network Model	No	No
Pattern Matching	Yes	Yes
Deep Neural Network Model with Threshold	Yes	Yes

Demo

Q & A