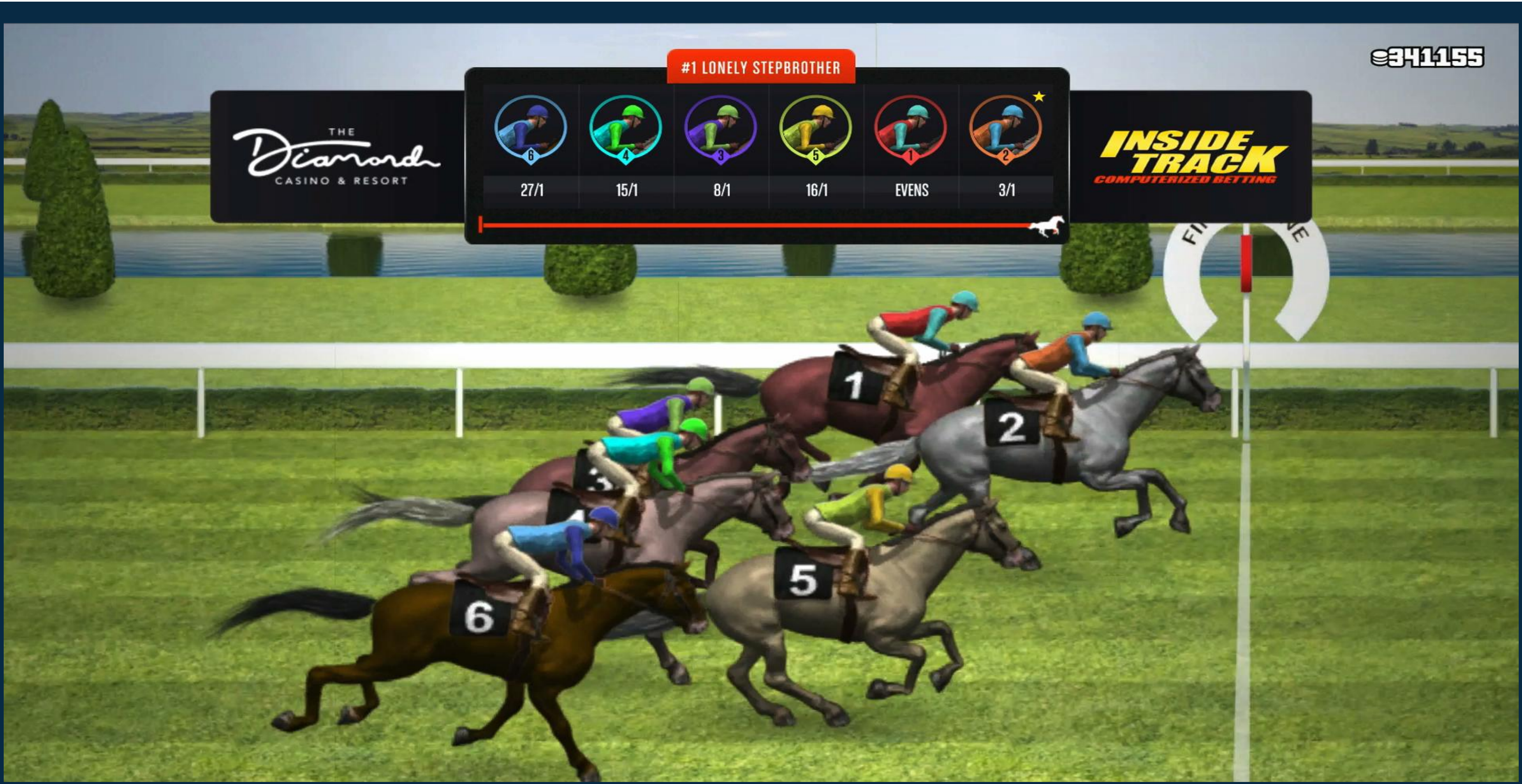
Proyek Kecerdasan Buatan

“Memprediksi berapa kemungkinan terbanyak bet tipe WIN yang bisa dimenangkan dari x percobaan “

KELOMPOK 12 :

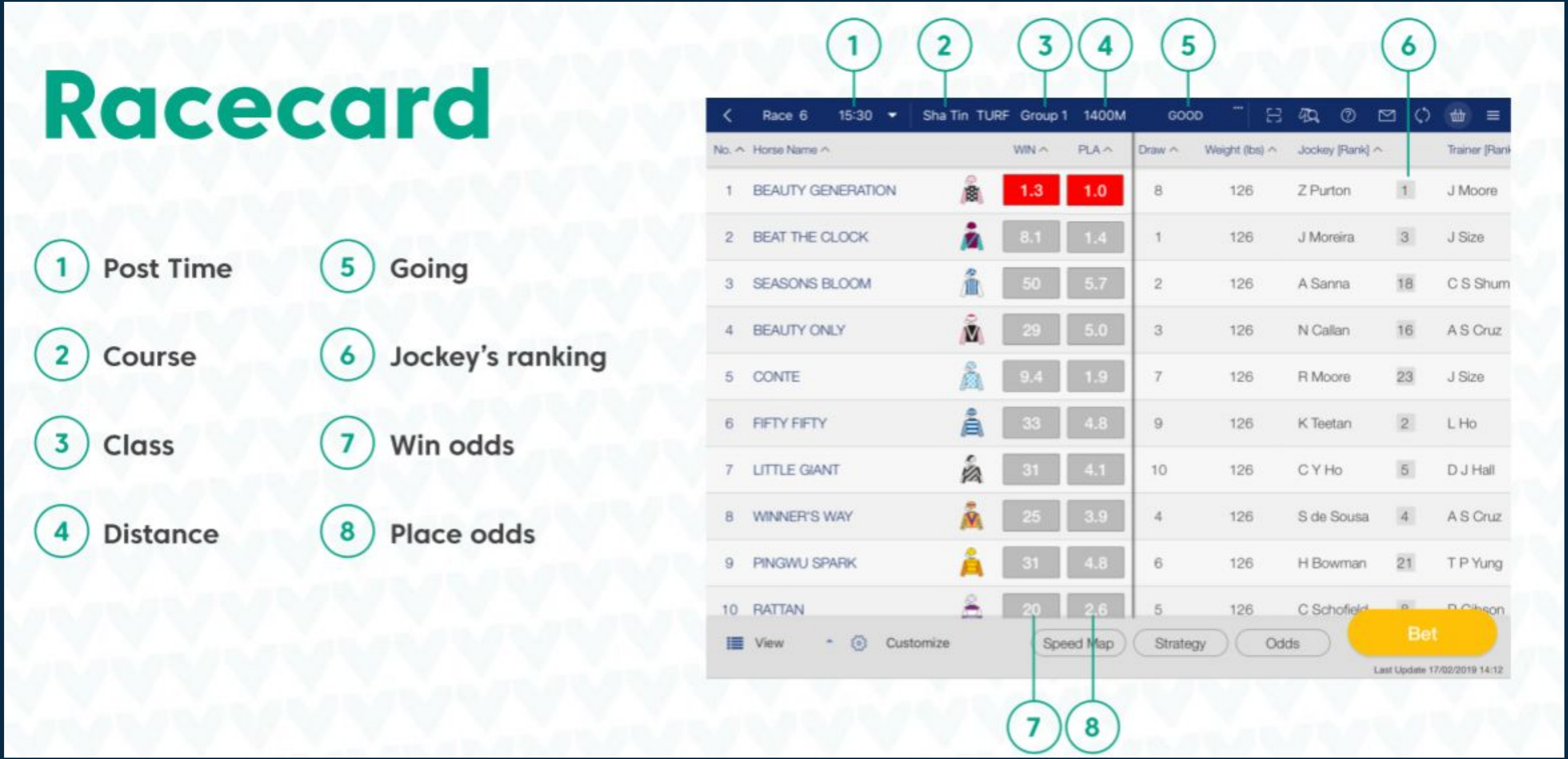
• ALEXANDER DEAN PANDREOU	(C14190132)
• ANDREW GERALDO	(C14190124)
• MARCELLINUS KELVIN T	(C14190132)
• ALBERTUS FARREL JUANDA	(C14190134)
• KEVIN CHANDRA H	(C14190136)

Ilustrasi


341155
#1 LONELY STEPBROTHER
THE Diamond CASINO & RESORT
6
4
3
5
1
2
27/1
15/1
8/1
16/1
EVENS
3/1
INSIDE TRACK
COMPUTERIZED BETTING
1
2
3
4
5
6

Penjelasan Horse Predict

Bagaimana betting dilakukan



Racecard

1. Post Time
2. Course
3. Class
4. Distance
5. Going
6. Jockey's ranking
7. Win odds
8. Place odds

Race 6 15:30 Sha Tin TURF Group 1 1400M GOOD

No.	Horse Name	WIN	PLA	Draw	Weight (lbs)	Jockey [Rank]		Trainer [Rank
1	BEAUTY GENERATION	1.3	1.0	8	126	Z Purton	1	J Moore
2	BEAT THE CLOCK	8.1	1.4	1	126	J Moreira	3	J Size
3	SEASONS BLOOM	50	5.7	2	126	A Sanna	18	C S Shum
4	BEAUTY ONLY	29	5.0	3	126	N Callan	16	A S Cruz
5	CONTE	9.4	1.9	7	126	R Moore	23	J Size
6	FIFTY FIFTY	33	4.8	9	126	K Teetan	2	L Ho
7	LITTLE GIANT	31	4.1	10	126	C Y Ho	5	D J Hall
8	WINNER'S WAY	25	3.9	4	126	S de Sousa	4	A S Cruz
9	PINGWU SPARK	31	4.8	6	126	H Bowman	21	T P Yung
10	RATTAN	20	2.6	5	126	C Schofield	[illegible]	P Gibson

Last Update 17/02/2019 14:12

Penjelasan Aturan

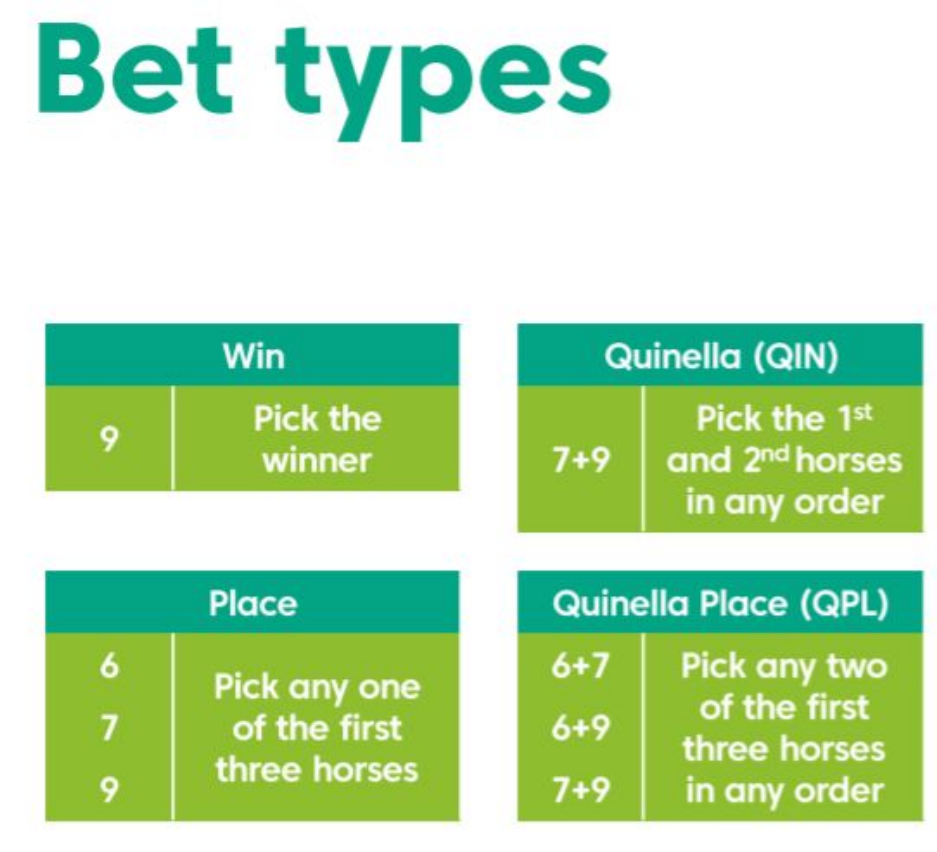
Terdapat 4 jenis bet:

1.Win

2.Place

3.Quinella

4.Quinella Place



Bet types

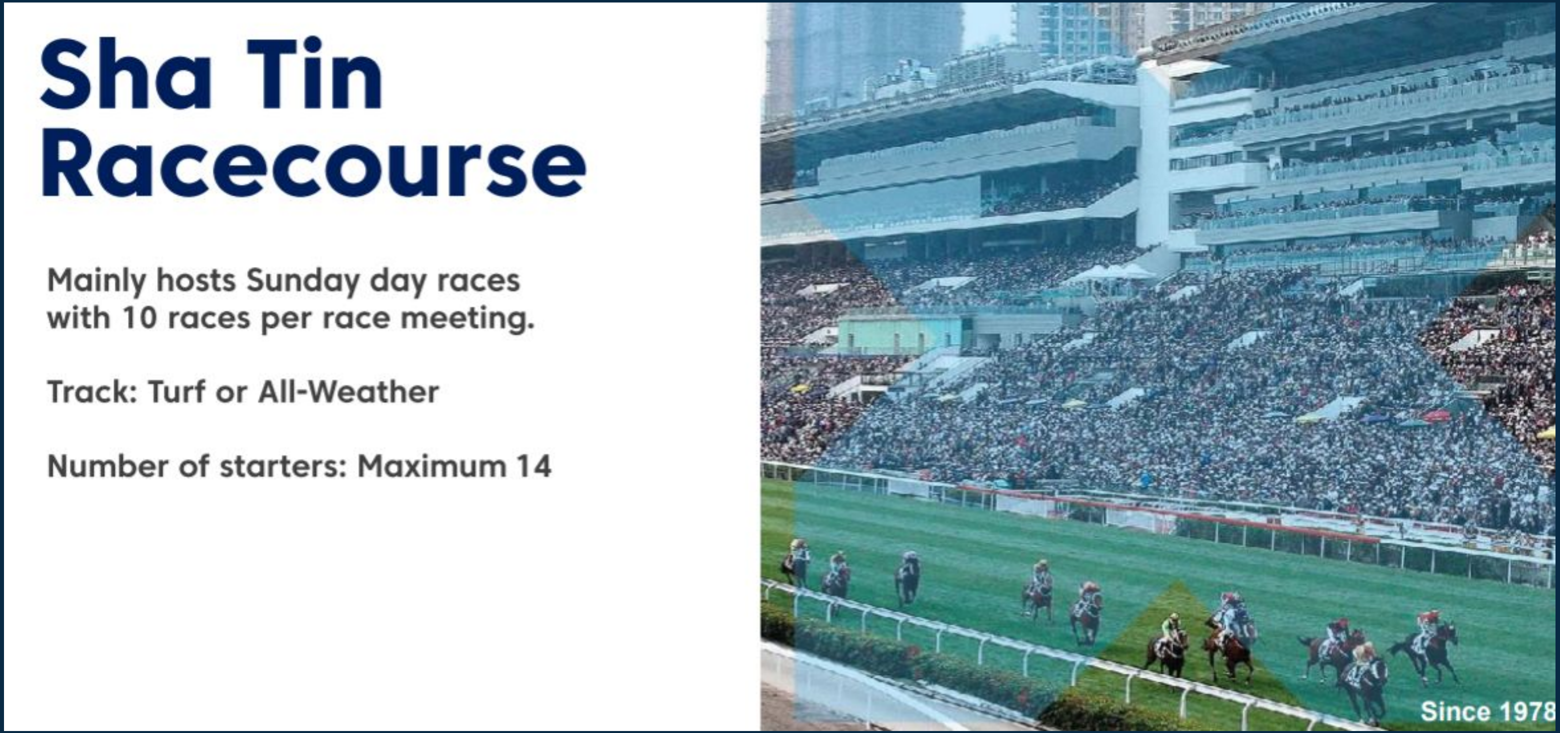
Win	
9	Pick the winner

Quinella (QIN)	
7+9	Pick the 1st and 2nd horses in any order

Place	
6 7 9	Pick any one of the first three horses

Quinella Place (QPL)	
6+7 6+9 7+9	Pick any two of the first three horses in any order

Jenis pertandingan



Sha Tin Racecourse

Mainly hosts Sunday day races with 10 races per race meeting.

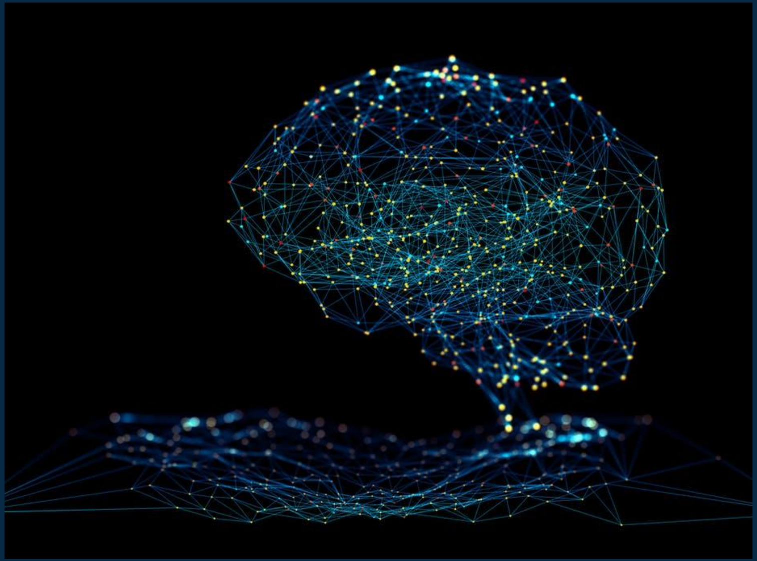
Track: Turf or All-Weather

Number of starters: Maximum 14

Since 1978

Metode Yang digunakan : Neural Network

Neural Network adalah bentuk pola yang mengadopsi layaknya kerja otak manusia dimana terdapat input data masuk,yang kemudian diproses,dan menghasilkan output.



Why neural network?

- Neural network memprediksi lebih baik, karena menggunakan hidden layer, dalam menguji dataset yang kita miliki.

- Dibanding menggunakan regresi linear, hanya menggunakan node input dan output dalam prediksi tanpa mengolah data lebih dalam.

Supervised or unsupervised?

menggunakan Supervised karena tujuan dari adanya training data adalah untuk menghasilkan hanya output yang kita kehendaki.

Kategori:

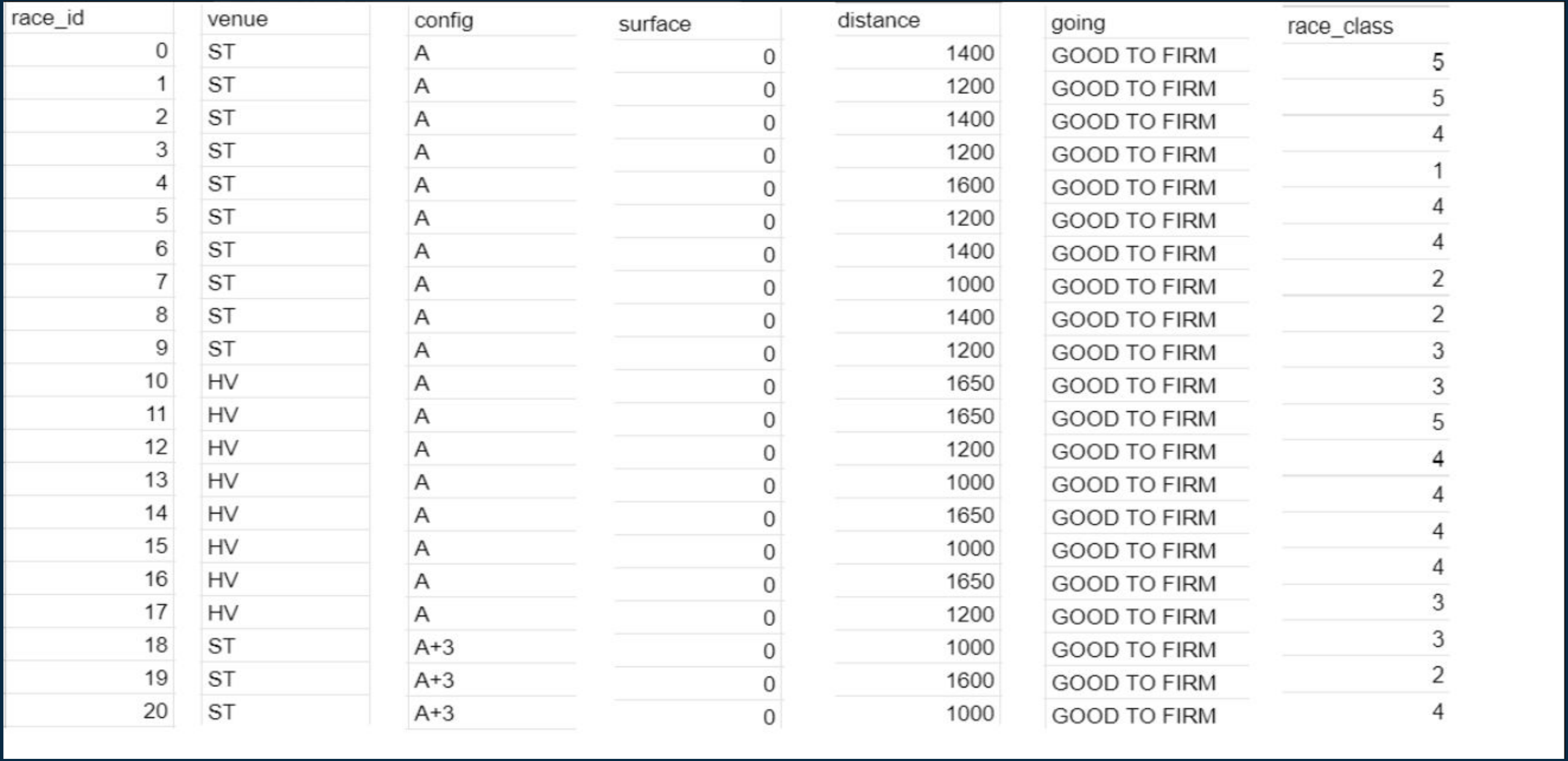
horse age, horse country, horse type, horse rating, actual weight.

Bila menggunakan unsupervised, output yang dihasilkan masih tidak diketahui polanya/abstrak, jadi bisa tidak sesuai harapan, dan dapat membuang waktu.

Data pre-processing

- Terdapat 2 data yang akan diproses
- Data-data ini berasal dari 2 file csv, yaitu races.csv dan runs.csv
- 2 tabel ini saling berkaitan pada kolom race_id dan diperlukan join dalam memproses data

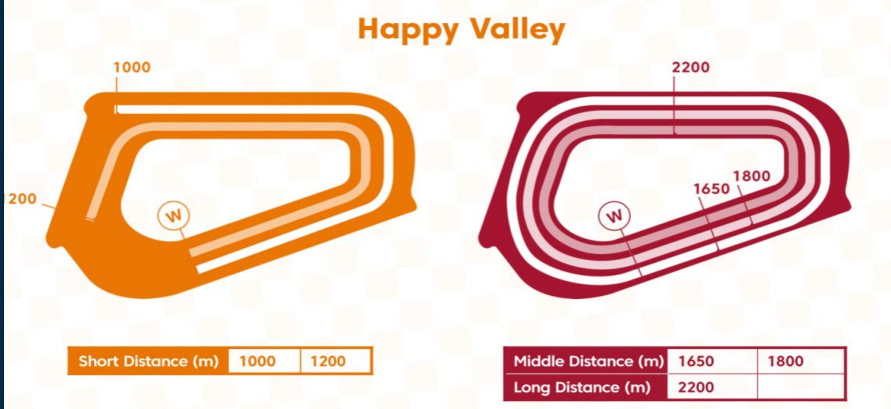
Data tabel Race.csv



race_id	venue	config	surface	distance	going	race_class
0	ST	A	0	1400	GOOD TO FIRM	5
1	ST	A	0	1200	GOOD TO FIRM	5
2	ST	A	0	1400	GOOD TO FIRM	4
3	ST	A	0	1200	GOOD TO FIRM	1
4	ST	A	0	1600	GOOD TO FIRM	4
5	ST	A	0	1200	GOOD TO FIRM	4
6	ST	A	0	1400	GOOD TO FIRM	2
7	ST	A	0	1000	GOOD TO FIRM	2
8	ST	A	0	1400	GOOD TO FIRM	3
9	ST	A	0	1200	GOOD TO FIRM	3
10	HV	A	0	1650	GOOD TO FIRM	5
11	HV	A	0	1650	GOOD TO FIRM	4
12	HV	A	0	1200	GOOD TO FIRM	4
13	HV	A	0	1000	GOOD TO FIRM	4
14	HV	A	0	1650	GOOD TO FIRM	4
15	HV	A	0	1000	GOOD TO FIRM	3
16	HV	A	0	1650	GOOD TO FIRM	3
17	HV	A	0	1200	GOOD TO FIRM	2
18	ST	A+3	0	1000	GOOD TO FIRM	4
19	ST	A+3	0	1600	GOOD TO FIRM	
20	ST	A+3	0	1000	GOOD TO FIRM	

Venue dan Surface

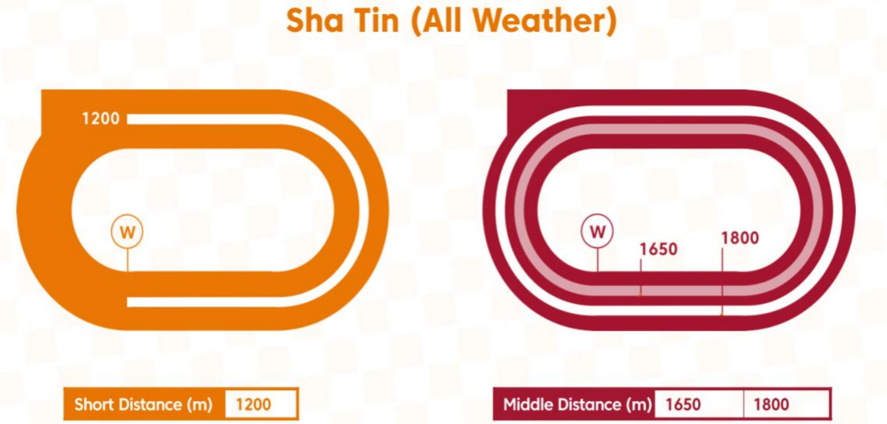
Surface 1=dirt,0=turf
Config kategori ,semakin turun kategori
width area semakin kecil
Venue ST=Shatin
HV=Happy Valley
Distance dalam meter
Going kondisi track



Happy Valley

Short Distance (m)	1000	1200

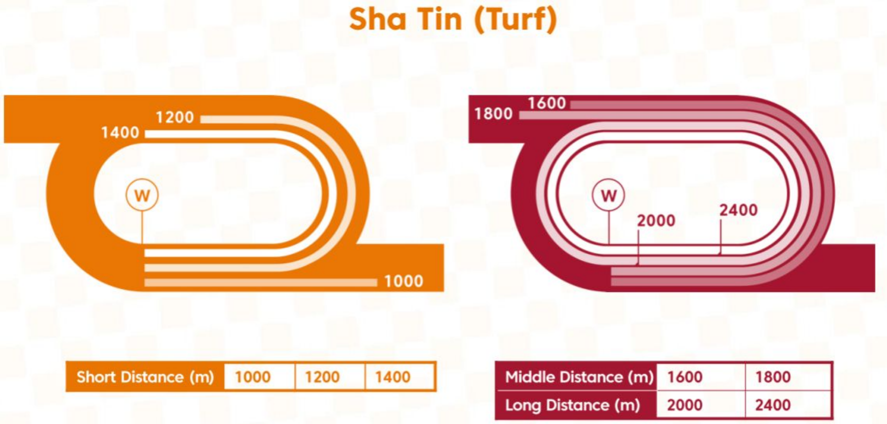
Middle Distance (m)	1650	1800
Long Distance (m)	2200	



Sha Tin (All Weather)

Short Distance (m)	1200

Middle Distance (m)	1650	1800



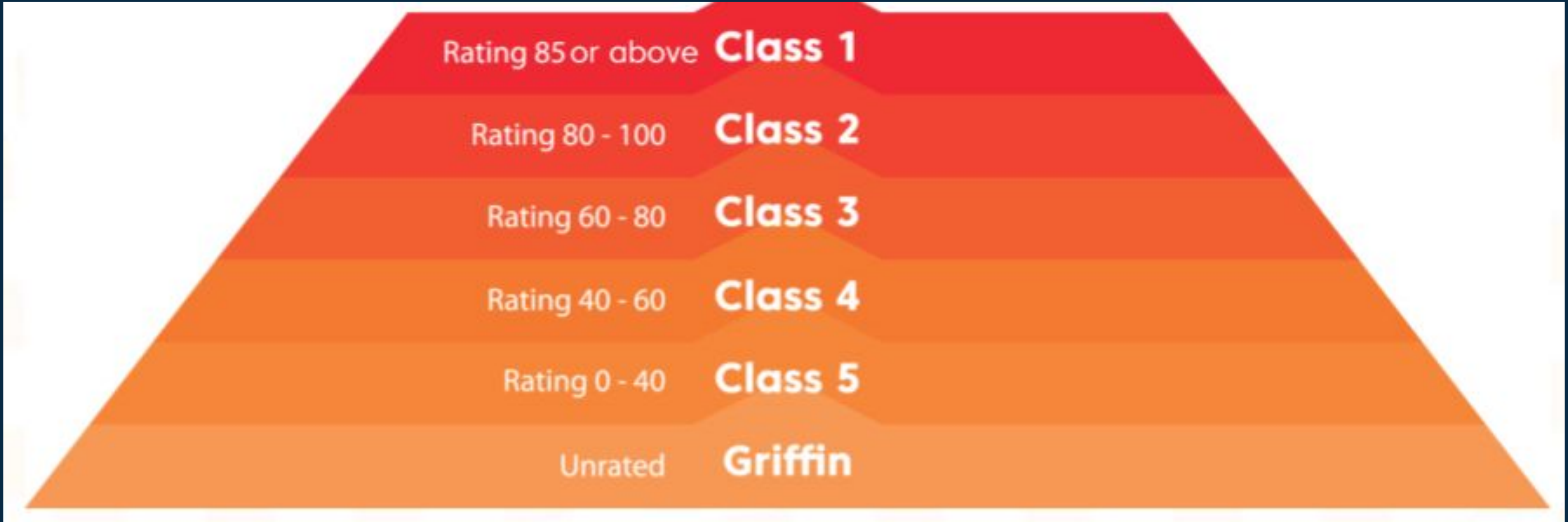
Sha Tin (Turf)

Short Distance (m)	1000	1200	1400

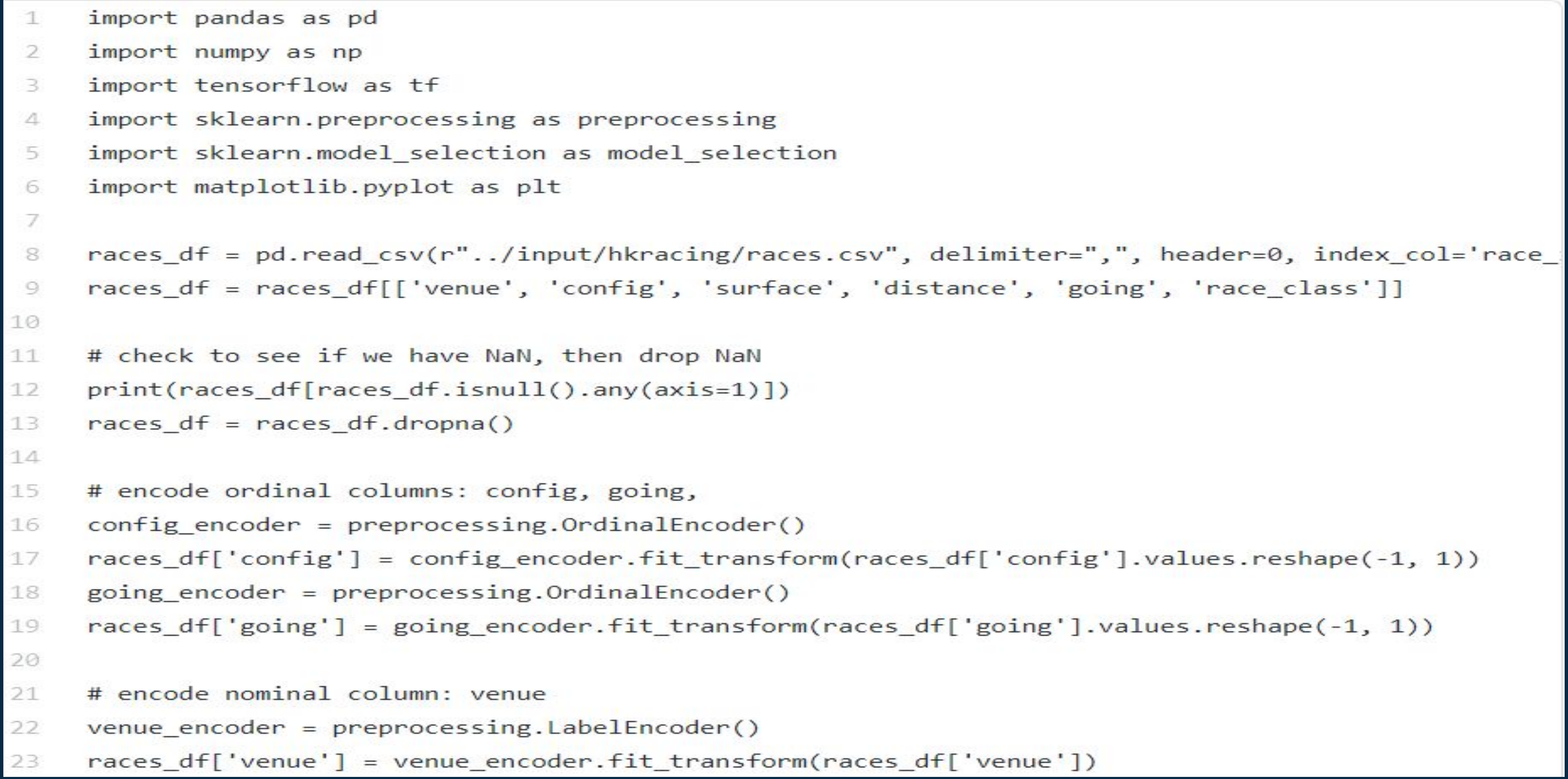
Middle Distance (m)	1600	1800
Long Distance (m)	2000	2400

Class_race

semakin kuat kuda class semakin tinggi,berhubungan dengan kolom rating

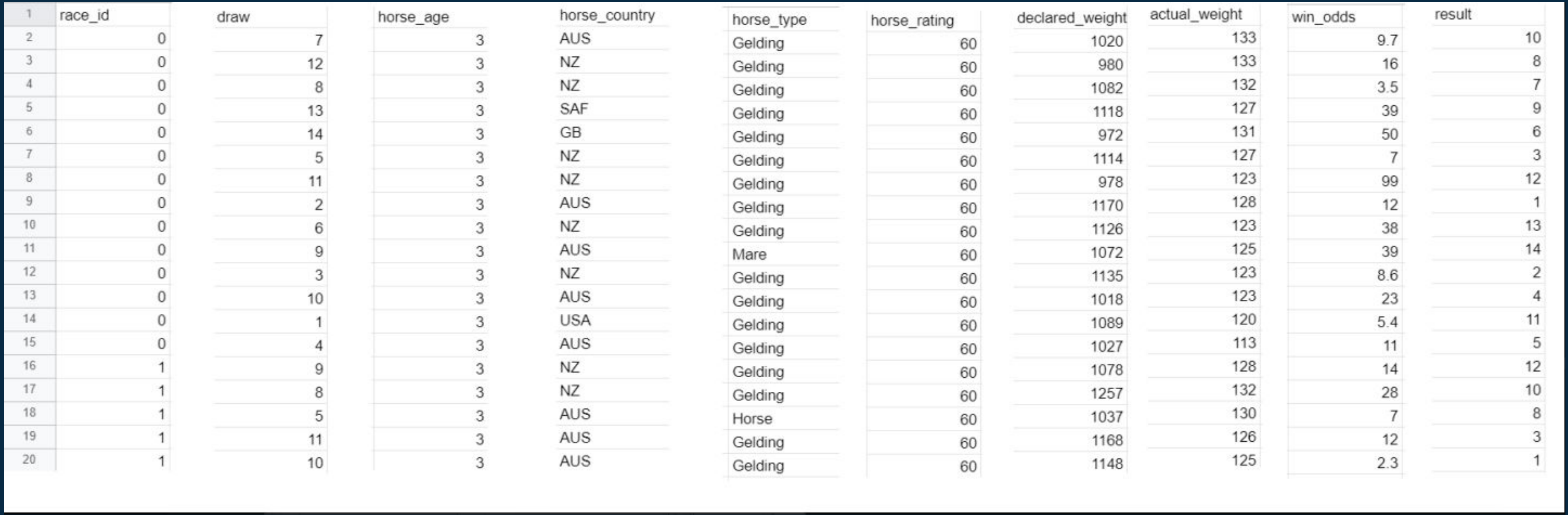


Memasukan data



```
import pandas as pd
import numpy as np
import tensorflow as tf
import sklearn.preprocessing as preprocessing
import sklearn.model_selection as model_selection
import matplotlib.pyplot as plt

races_df = pd.read_csv(r"../input/hkracing/races.csv", delimiter=",", header=0, index_col='race_
races_df = races_df[['venue', 'config', 'surface', 'distance', 'going', 'race_class']]

# check to see if we have NaN, then drop NaN
print(races_df[races_df.isnull().any(axis=1)])
races_df = races_df.dropna()

# encode ordinal columns: config, going,
config_encoder = preprocessing.OrdinalEncoder()
races_df['config'] = config_encoder.fit_transform(races_df['config'].values.reshape(-1, 1))
going_encoder = preprocessing.OrdinalEncoder()
races_df['going'] = going_encoder.fit_transform(races_df['going'].values.reshape(-1, 1))

# encode nominal column: venue
venue_encoder = preprocessing.LabelEncoder()
races_df['venue'] = venue_encoder.fit_transform(races_df['venue'])
```

Data tabel Runs.csv



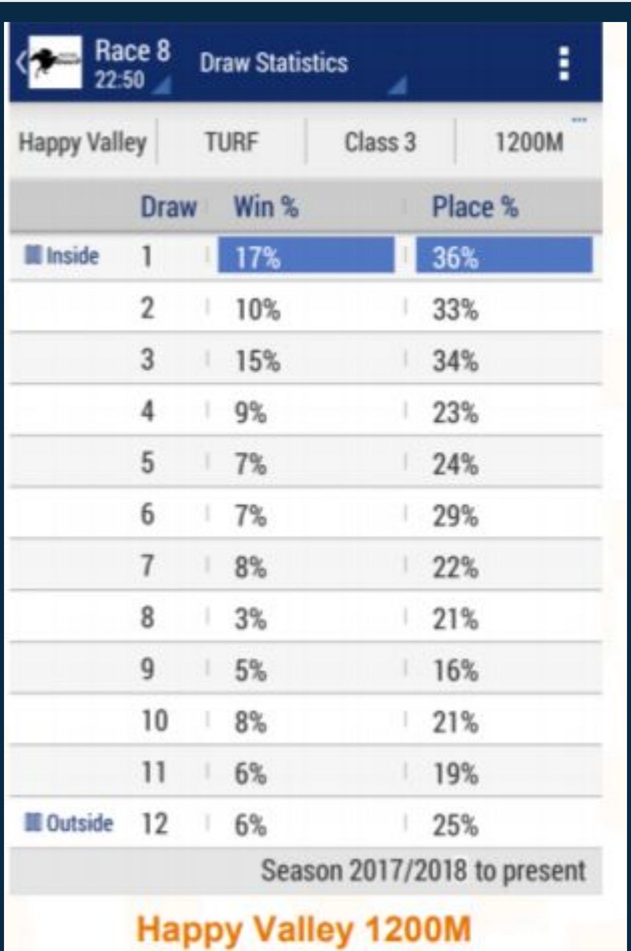
1	race_id	draw	horse_age	horse_country	horse_type	horse_rating	declared_weight	actual_weight	win_odds	result
2	0	7	3	AUS	Gelding	60	1020	133	9.7	10
3	0	12	3	NZ	Gelding	60	980	133	16	8
4	0	8	3	NZ	Gelding	60	1082	132	3.5	7
5	0	13	3	SAF	Gelding	60	1118	127	39	9
6	0	14	3	GB	Gelding	60	972	131	50	6
7	0	5	3	NZ	Gelding	60	1114	127	7	3
8	0	11	3	NZ	Gelding	60	978	123	99	12
9	0	2	3	AUS	Gelding	60	1170	128	12	1
10	0	6	3	NZ	Gelding	60	1126	123	38	13
11	0	9	3	AUS	Mare	60	1072	125	39	14
12	0	3	3	NZ	Gelding	60	1135	123	8.6	2
13	0	10	3	AUS	Gelding	60	1018	123	23	4
14	0	1	3	USA	Gelding	60	1089	120	5.4	11
15	0	4	3	AUS	Gelding	60	1027	113	11	5
16	1	9	3	NZ	Gelding	60	1078	128	14	12
17	1	8	3	NZ	Gelding	60	1257	132	28	10
18	1	5	3	AUS	Horse	60	1037	130	7	8
19	1	11	3	AUS	Gelding	60	1168	126	12	3
20	1	10	3	AUS	Gelding	60	1148	125	2.3	1

Draw

Draw adalah posisi kuda di starting gate

Semakin kecil angka draw,semakin dekat dengan pembatas,semakin kecil jarak belokan

Artinya,semakin diuntungkan



Race 8
22:50

Draw Statistics

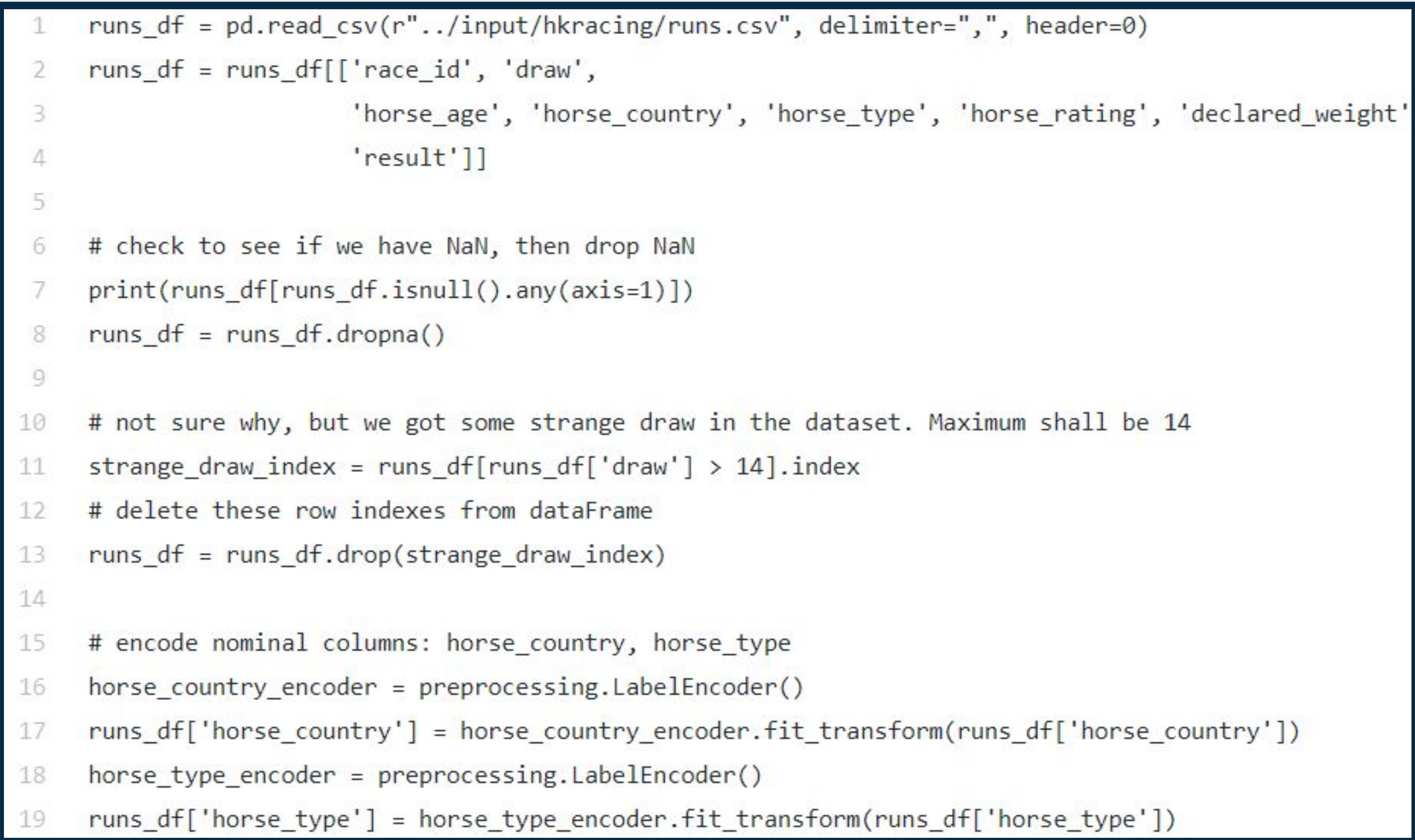
Happy Valley | TURF | Class 3 | 1200M

	Draw	Win %	Place %
Inside	1	17%	36%
	2	10%	33%
	3	15%	34%
	4	9%	23%
	5	7%	24%
	6	7%	29%
	7	8%	22%
	8	3%	21%
	9	5%	16%
	10	8%	21%
	11	6%	19%
Outside	12	6%	25%

Season 2017/2018 to present

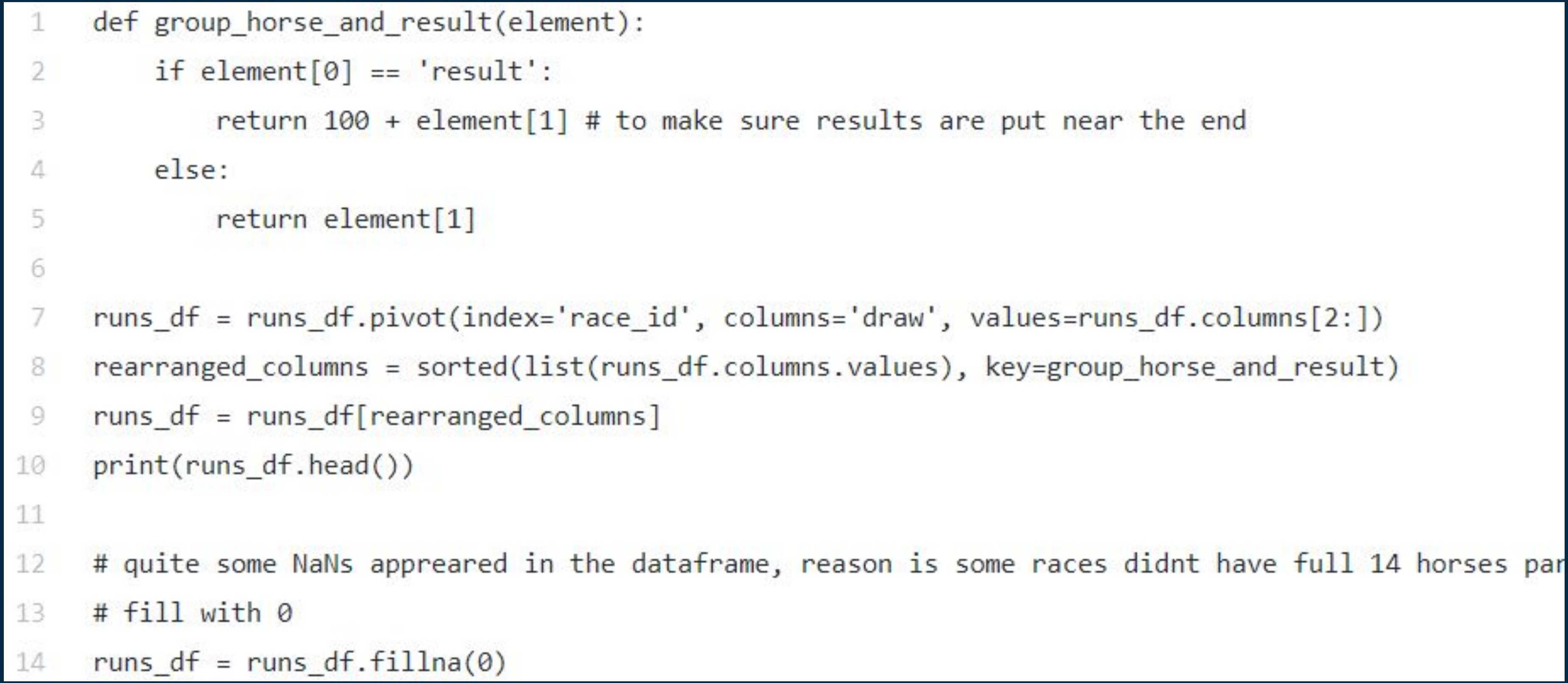
Happy Valley 1200M

Memasukan data



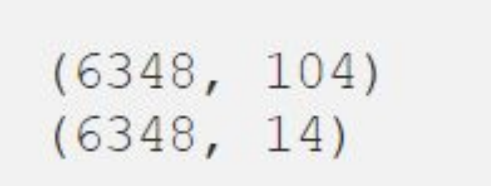
```
runs_df = pd.read_csv(r"../input/hkracing/runs.csv", delimiter=",", header=0)
runs_df = runs_df[['race_id', 'draw',
                   'horse_age', 'horse_country', 'horse_type', 'horse_rating', 'declared_weight'
                   'result']]

# check to see if we have NaN, then drop NaN
print(runs_df[runs_df.isnull().any(axis=1)])
runs_df = runs_df.dropna()

# not sure why, but we got some strange draw in the dataset. Maximum shall be 14
strange_draw_index = runs_df[runs_df['draw'] > 14].index
# delete these row indexes from dataFrame
runs_df = runs_df.drop(strange_draw_index)

# encode nominal columns: horse_country, horse_type
horse_country_encoder = preprocessing.LabelEncoder()
runs_df['horse_country'] = horse_country_encoder.fit_transform(runs_df['horse_country'])
horse_type_encoder = preprocessing.LabelEncoder()
runs_df['horse_type'] = horse_type_encoder.fit_transform(runs_df['horse_type'])
```

Mengubah urutan data

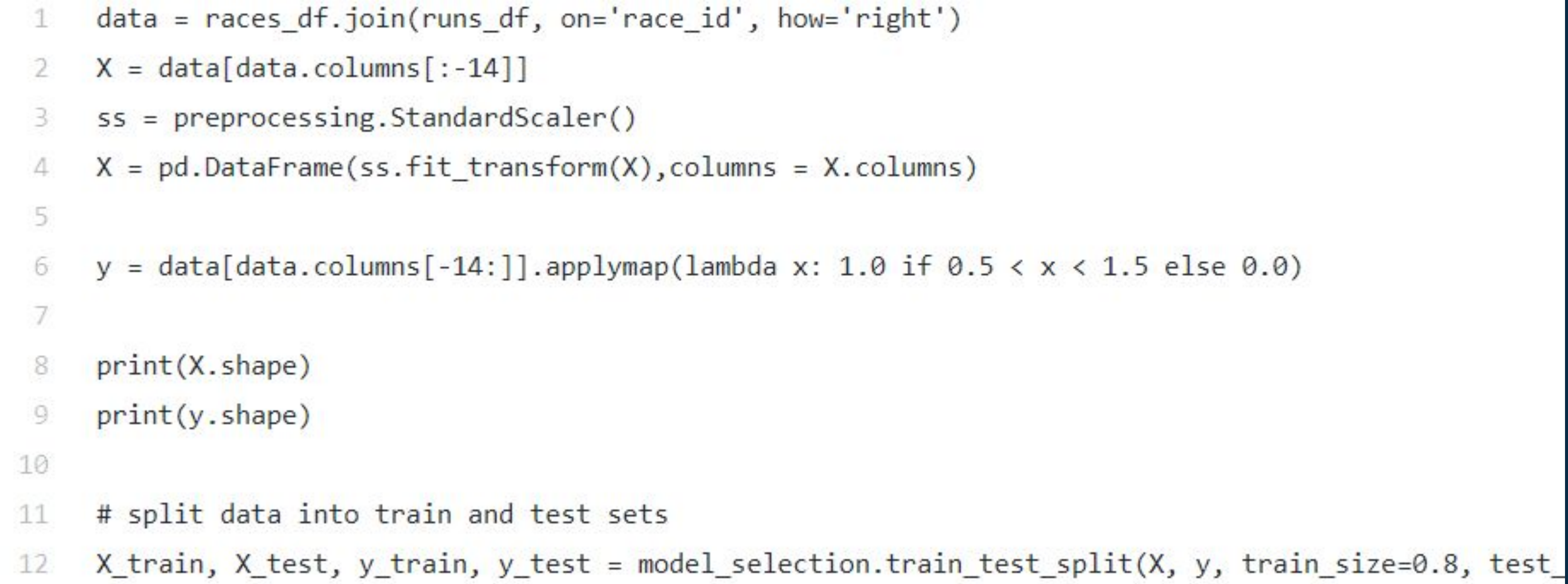


```
def group_horse_and_result(element):
    if element[0] == 'result':
        return 100 + element[1] # to make sure results are put near the end
    else:
        return element[1]

runs_df = runs_df.pivot(index='race_id', columns='draw', values=runs_df.columns[2:])
rearranged_columns = sorted(list(runs_df.columns.values), key=group_horse_and_result)
runs_df = runs_df[rearranged_columns]
print(runs_df.head())

# quite some NaNs appreared in the dataframe, reason is some races didnt have full 14 horses par
# fill with 0
runs_df = runs_df.fillna(0)
```

Join data Split data train dengan test

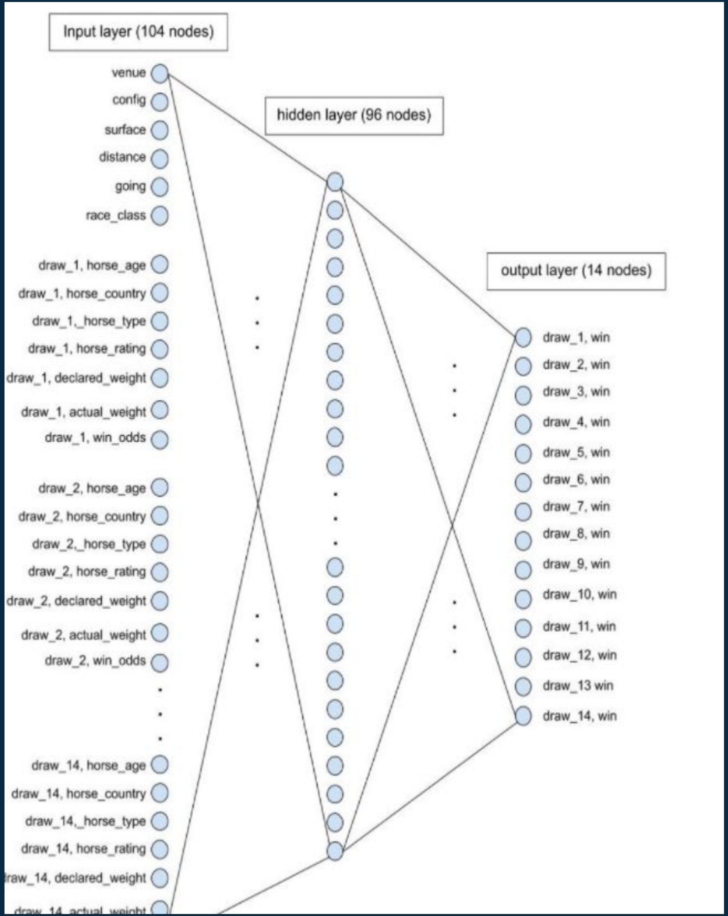


```
(6348, 104)
(6348, 14)
```



```python
data = races_df.join(runs_df, on='race_id', how='right')
X = data[data.columns[:-14]]
ss = preprocessing.StandardScaler()
X = pd.DataFrame(ss.fit_transform(X),columns = X.columns)

y = data[data.columns[-14:]].applymap(lambda x: 1.0 if 0.5 < x < 1.5 else 0.0)

print(X.shape)
print(y.shape)

# split data into train and test sets
X_train, X_test, y_train, y_test = model_selection.train_test_split(X, y, train_size=0.8, test_
```

Make the neural network model



Ada 3 layer:

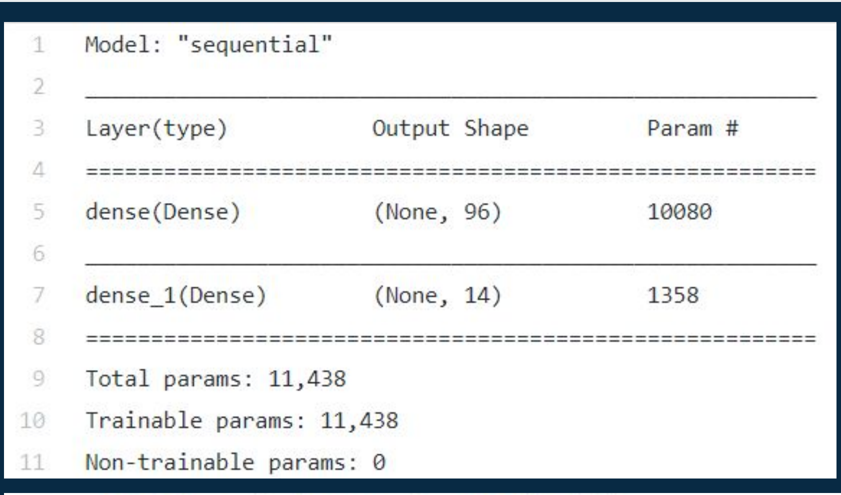
1. Input layer
   Ada 104 node
   6 dari race.csv (cuma 1 match saja jadi 6*1)
   7 *14 peserta ->karena dari 14 peserta masing-masing punya 7 data jadi 7*14
   Total 104 node

2. Hidden Layer
   Jumlah node pada hidden layer tidak ditentukan,jumlahnya bebas seusai kebutuhan
   Tapi ada beberapa kriteria
   1.Harus di antara input dan output,misal diantara 14-104
   2.Untuk menghitung estimasi node yang dibutuhkan menggunakan rumus sqrt(input node *output node)

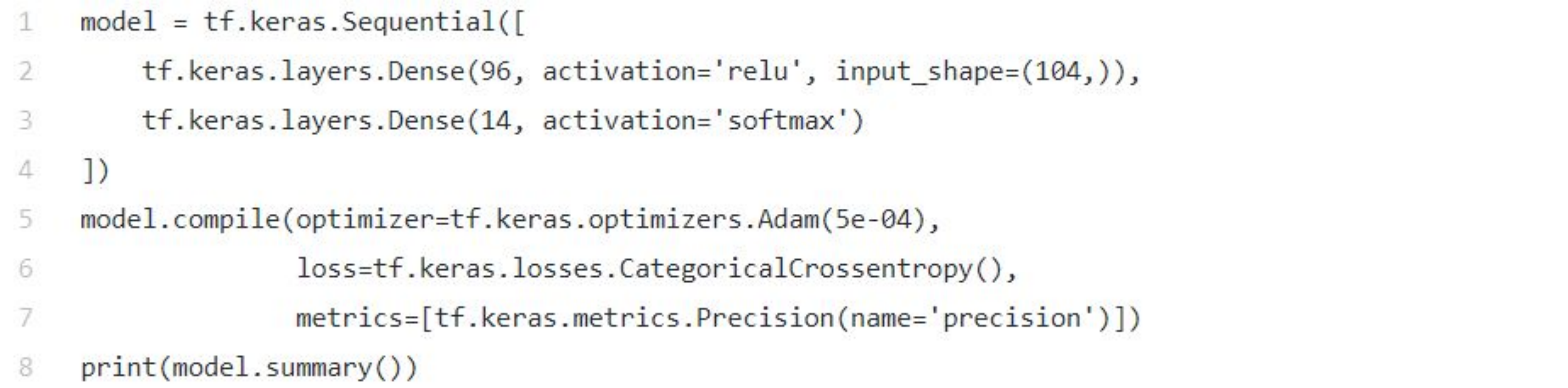
3. Output Layer
   Jumlah output layer yang ada sebanyak 14 node,karena terdapat 14 peserta ,
   Dimana tiap node mengindikasikan apakah seekor kuda memenangkan Pertandingan

   Penjelasan code
   Bisa disimpulkan banyak parameter yang bisa di train adalah 11,438

Memasukan Ke Neural Network

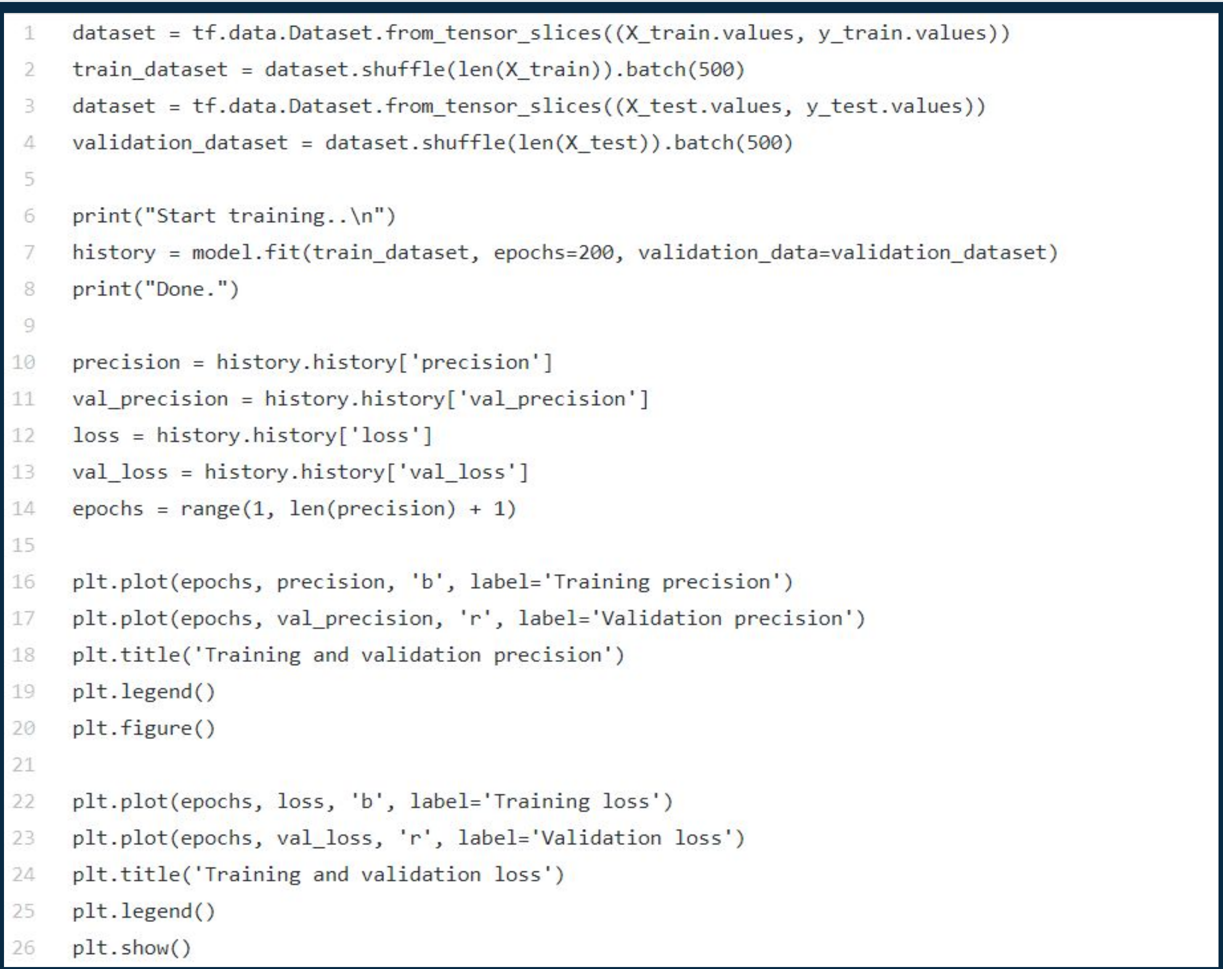


```
Model: "sequential"
________________________________________________________
Layer(type)           Output Shape          Param #
========================================================
dense(Dense)          (None, 96)            10080
________________________________________________________
dense_1(Dense)        (None, 14)            1358
========================================================
Total params: 11,438
Trainable params: 11,438
Non-trainable params: 0
```



```
model = tf.keras.Sequential([
    tf.keras.layers.Dense(96, activation='relu', input_shape=(104,)),
    tf.keras.layers.Dense(14, activation='softmax')
])
model.compile(optimizer=tf.keras.optimizers.Adam(5e-04),
              loss=tf.keras.losses.CategoricalCrossentropy(),
              metrics=[tf.keras.metrics.Precision(name='precision')])
print(model.summary())
```

Train data



```
dataset = tf.data.Dataset.from_tensor_slices((X_train.values, y_train.values))
train_dataset = dataset.shuffle(len(X_train)).batch(500)
dataset = tf.data.Dataset.from_tensor_slices((X_test.values, y_test.values))
validation_dataset = dataset.shuffle(len(X_test)).batch(500)

print("Start training..\n")
history = model.fit(train_dataset, epochs=200, validation_data=validation_dataset)
print("Done.")

precision = history.history['precision']
val_precision = history.history['val_precision']
loss = history.history['loss']
val_loss = history.history['val_loss']
epochs = range(1, len(precision) + 1)

plt.plot(epochs, precision, 'b', label='Training precision')
plt.plot(epochs, val_precision, 'r', label='Validation precision')
plt.title('Training and validation precision')
plt.legend()
plt.figure()

plt.plot(epochs, loss, 'b', label='Training loss')
plt.plot(epochs, val_loss, 'r', label='Validation loss')
plt.title('Training and validation loss')
plt.legend()
plt.show()
```

Cara Kerja model.fit

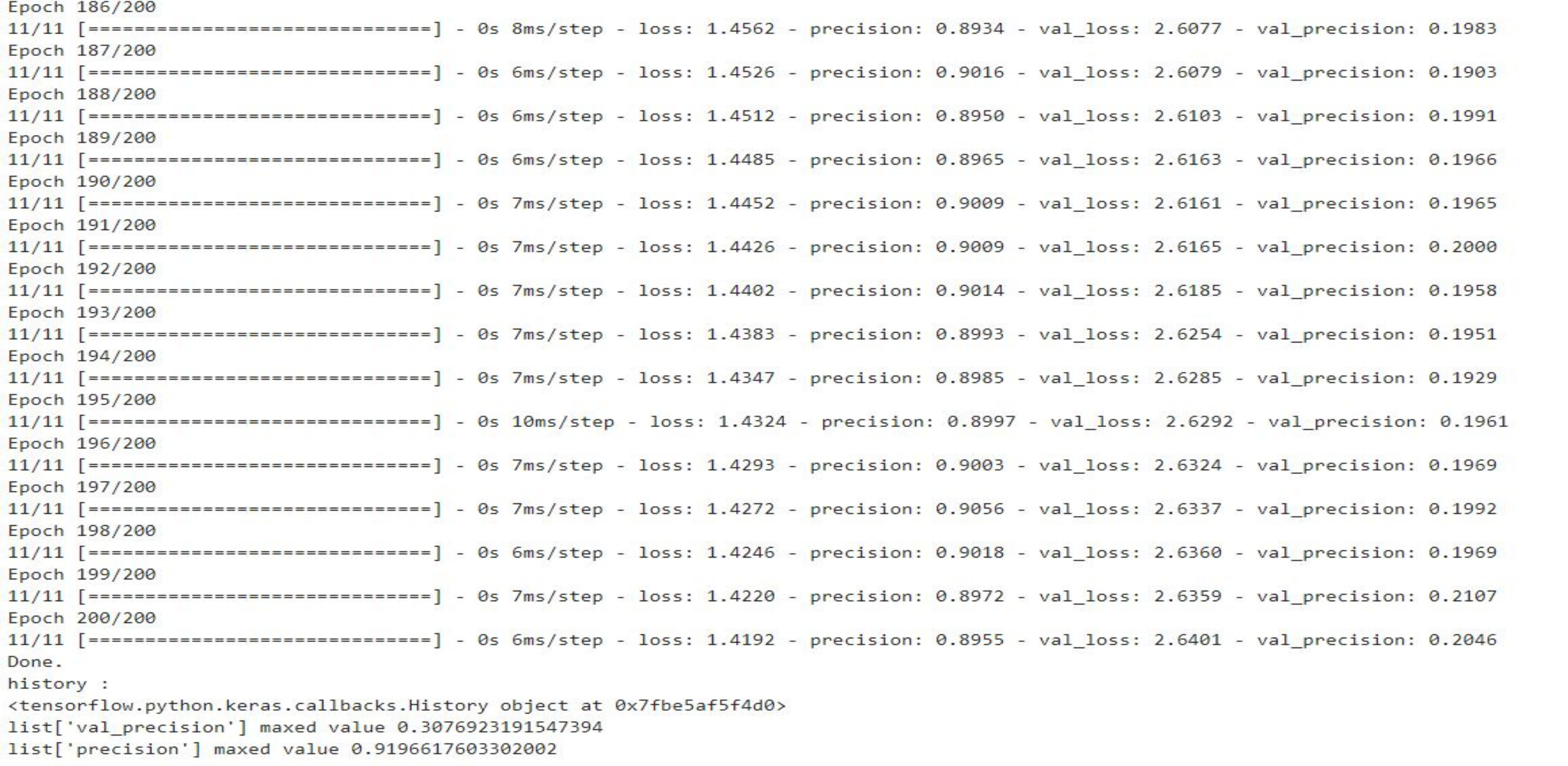
```
Model.fit(train_dataset,epochs=200,validation_data=validation_dataset)
```

- train _dataset sebagai input
- Epochs adalah berapa kali data akan ditrain, 200 berarti 200 kali training
- validation_data = data yang digunakan untuk men-train dataset yang ada,

  Hasil disimpan di History

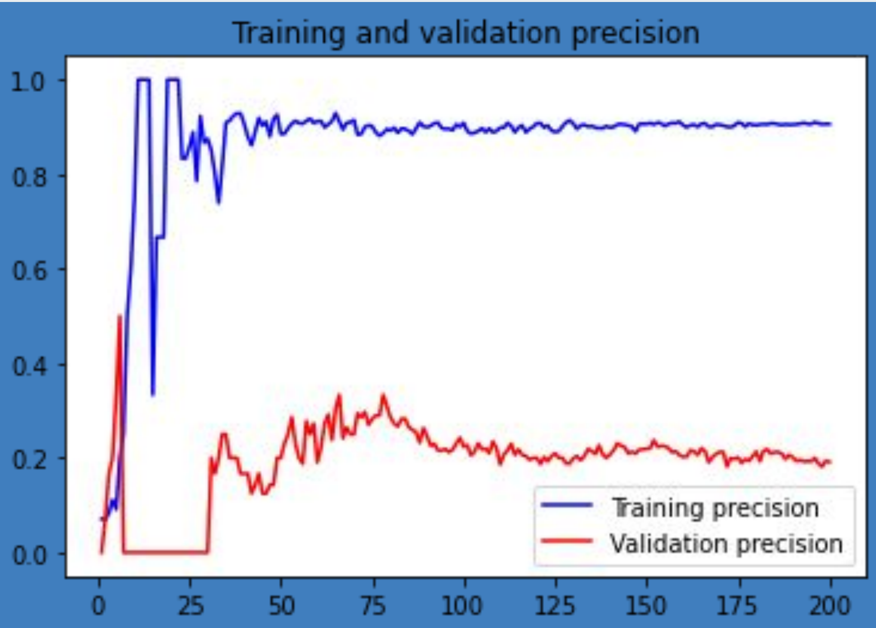
  Tujuan menunjukan tingkat kepresisian data yang ada,dan seberapa presisi kemenangan terjadi bila melakukan bet

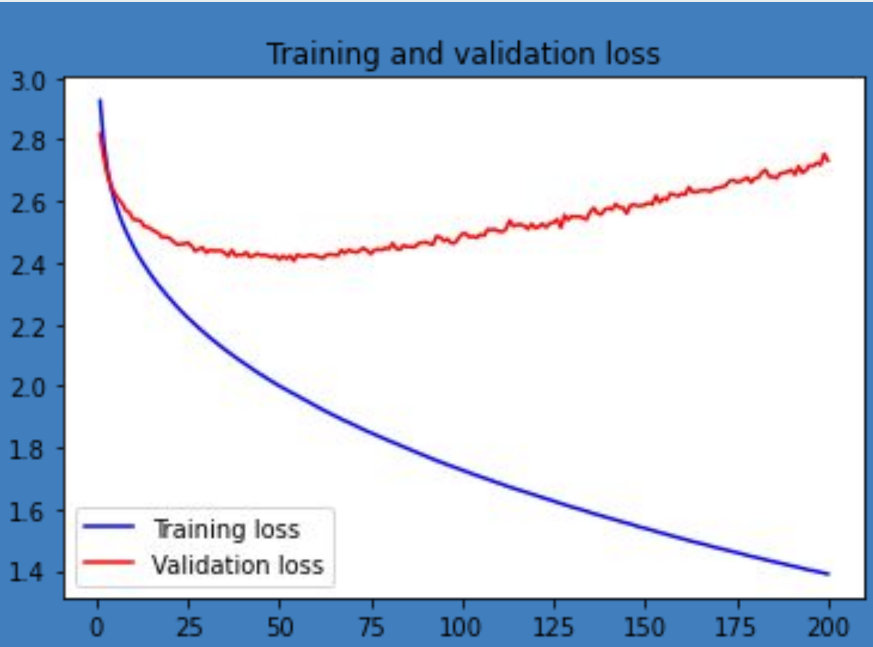
output



```
Epoch 186/200
11/11 [==============================] - 0s 8ms/step - loss: 1.4562 - precision: 0.8934 - val_loss: 2.6077 - val_precision: 0.1983
Epoch 187/200
11/11 [==============================] - 0s 6ms/step - loss: 1.4526 - precision: 0.9016 - val_loss: 2.6079 - val_precision: 0.1903
Epoch 188/200
11/11 [==============================] - 0s 6ms/step - loss: 1.4512 - precision: 0.8950 - val_loss: 2.6103 - val_precision: 0.1991
Epoch 189/200
11/11 [==============================] - 0s 6ms/step - loss: 1.4485 - precision: 0.8965 - val_loss: 2.6163 - val_precision: 0.1966
Epoch 190/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4452 - precision: 0.9009 - val_loss: 2.6161 - val_precision: 0.1965
Epoch 191/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4426 - precision: 0.9009 - val_loss: 2.6165 - val_precision: 0.2000
Epoch 192/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4402 - precision: 0.9014 - val_loss: 2.6185 - val_precision: 0.1958
Epoch 193/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4383 - precision: 0.8993 - val_loss: 2.6254 - val_precision: 0.1951
Epoch 194/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4347 - precision: 0.8985 - val_loss: 2.6285 - val_precision: 0.1929
Epoch 195/200
11/11 [==============================] - 0s 10ms/step - loss: 1.4324 - precision: 0.8997 - val_loss: 2.6292 - val_precision: 0.1961
Epoch 196/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4293 - precision: 0.9003 - val_loss: 2.6324 - val_precision: 0.1969
Epoch 197/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4272 - precision: 0.9056 - val_loss: 2.6337 - val_precision: 0.1992
Epoch 198/200
11/11 [==============================] - 0s 6ms/step - loss: 1.4246 - precision: 0.9018 - val_loss: 2.6360 - val_precision: 0.1969
Epoch 199/200
11/11 [==============================] - 0s 7ms/step - loss: 1.4220 - precision: 0.8972 - val_loss: 2.6359 - val_precision: 0.2107
Epoch 200/200
11/11 [==============================] - 0s 6ms/step - loss: 1.4192 - precision: 0.8955 - val_loss: 2.6401 - val_precision: 0.2046
Done.
history :
<tensorflow.python.keras.callbacks.History object at 0x7fbe5af5f4d0>
list['val_precision'] maxed value 0.3076923191547394
list['precision'] maxed value 0.9196617603302002
```

Hasil dan graph





Disini precision, adalah seberapa akurat hasil training data

Val_precision = Output seberapa presisi kemenangan bet yang kita dapatkan, misal 0.3 jadi 30% win

Loss = Data tidak akurat

Kesimpulan

Hasil yang dihasilkan

Sebuah nilai precision tertinggi dari berapa banyak epochs training yang dilakukan
Misal precision tertinggi adalah 0.3 maka jika memprediksi win sebanyak 10 kali,hanya 3 yang benar

Dari graph
Hasil terbaik yang bisa didapat adalah 0,3,dan setelah itu terjadi overfitting, data yang dihasilkan tidak akurat

Jadi Bisa disimpulkan,kemungkinan terbaik untuk memenangkan bet berdasarkan data yang ada adalah sebanyak 3 kali dari 10 kali bet