HORSE RACING PREDICTION

Evan And Anson

DS105 – Machine Learning Project Proposal

PROBLEM STATEMENT

‘Friend A’ interest to do betting for future horse racing in Singapore Turf Club. He seek for advice on how to do the bet for the horse racing.

- Betting Strategy:

We plan to split to bet on 3 horses to win the race. Every horses will bet $10 each. If 1 of the 3 horses that we bet win the race , we still can win the bet.

- For example:

  According to Singapore Pools, the odds for horse racing as below.

No	Horse Name	Odds for 1st Place	Bet Amount	Total
1	SuperBest	1:5	$10	$50
2	NorthStar	1:8	$10	$80
3	LittleBoy	1:10	$10	$100

Prize won for SuperBest to win the race = $50 - $30 (Cost to bet 3 horses) = $20 (Won)

Prize won for NorthStar to win the race = $80 - $30 (Cost to bet 3 horses) = $50 (Won)

Prize won for LittleBoy to win the race = $100 - $30 (Cost to bet 3 horses) = $70 (Won)

If we guess one of the 3 horses to win the race, we still can win the bet.

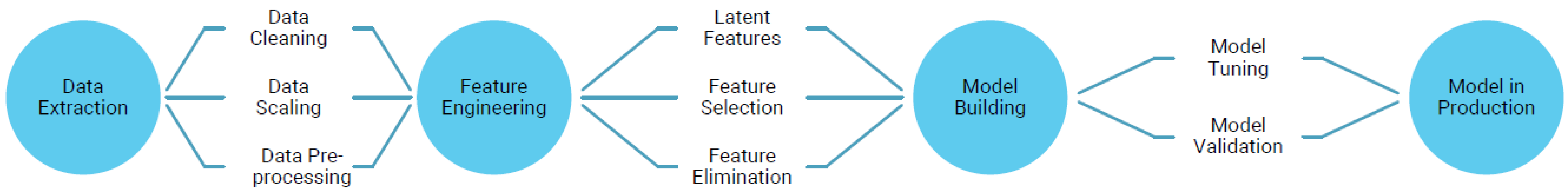
In the past record, it may happen to have two winners. If that’s the case, it will be a bonus for us to win the race.

OBJECTIVE

To develop a model that can predict which 3 horses can win the race (Top 3 placing) to help ‘Friend A’ to win the bet.

STAGES IN MACHINE LEARNING (ML)

STAGES IN MACHINE LEARNING



STAGE 1: DATA EXTRACTION

Source of the data:

https://racing.turfclub.com.sg/ (Turf Club Home Page)

- We choose Singapore turf club as it had wide range of data available regarding about horse racing since 1992.

- We choose the date from 01 January 2017 to 31 October 2022 (6 Years) as we believe that average racehorses can run at its peak 4 to 5 years.

- Scrap the data from website using selenium

- Final output as the result of scrapping with selenium.

A	B	C	D	E	F	G	H	I	J	K	L	M	N
Pl	horse_name	bar	Track	Gear	Rtg	jockey	trainer	Owner	Last 800m	Last 400m	Final	Second	total second to finish
10	A LA VICTORY	7	1200	B, TT	79	K A'ISISUH	J PETERS	CHINA HO	8	9	10	8.6	71
10	A LA VICTORY	7	1200	B, TT	79	K A'ISISUH	J PETERS	CHINA HO	8	9	10	8.6	71
5	A LA VICTORY	6	1200	B, TT	77	SY MOON	J PETERS	CHINA HO	8	7	5	2.7	70
3	A LOT IN HAND	4	1100	B	43	APP CK NG	HK TAN	HAPPY LIFI	3	3	3	2.9	67
8	A LOT IN HAND	3	1200	B	43	TH KOH	HK TAN	HAPPY LIFI	2	2	8	8.1	74
4	A LOT IN HAND	11	1200	B	41	M EWE	HK TAN	HAPPY LIFI	2	2	4	5.1	73
3	A LOT IN HAND	2	1200	B	39	M EWE	HK TAN	HAPPY LIFI	1	1	3	2.1	73
5	A LOT IN HAND	11	1200	B	39	APP CK NG	HK TAN	HAPPY LIFI	2	3	5	5.1	73
8	A LOT IN HAND	4	1100	B	38	D DAVID	HK TAN	HAPPY LIFI	1	2	8	6	66
9	AABIR	3	1000	TT		V DURIC	M WALKEF	JOHN ERIC	7	8	9	10.2	62
2	AABIR	6	1000	TT		R WOODW	M WALKEF	JOHN ERIC	1	1	2	0.3	119
8	AABIR	7	1200	TT		R WOODW	M WALKEF	JOHN ERIC	5	6	8	3.1	72
2	AABIR	10	1100	TT	48	R ZAWARI	M WALKEF	JOHN ERIC	2	2	2	0.8	65
3	AABIR	3	1000	TT	48	APP AB RII	M WALKEF	JOHN ERIC	2	2	3	1.3	60
3	AABIR	12	1000	TT	48	R ZAWARI	M WALKEF	JOHN ERIC	5	5	3	2.1	119
11	AABIR	3	1000	TT	48	APP AB RII	M WALKEF	JOHN ERIC	5	6	11	9.5	61
11	AABIR	9	1100	TT	48	R ZAWARI	M WALKEF	JOHN ERIC	3	3	11	8.9	67
6	AASKAR	10	1000	PP, TT	60	V DURIC	D LOGAN	AJ'S STABL	1	2	6	3.5	61
1	AASKAR	4	1000	P, TT	60	B THOMPS	D LOGAN	AJ'S STABL	1	1	1	0	60
9	AASKAR	6	1000	P, TT	61	B THOMPS	D LOGAN	AJ'S STABL	2	1	9	5.3	61
9	AASKAR	6	1000	P, TT	61 (+1)	S JOHN	D LOGAN	AJ'S STABL	7	4	9	15.2	62
12	AASKAR	2	1000	BB, TT	60 (-1)	I AZHAR	D LOGAN	AJ'S STABL	3	2	12	15.7	62
3	ABEBE	3	1400	BB	56	J POWELL	S BAERTSC	LIM CHUN	4	6	3	2.8	84
13	ABEBE	9	1600	B	56	J POWELL	S BAERTSC	LIM CHUN	12	14	13	38.7	104

COLUMN ATTRIBUTES (FEATURES)

Total 14 Features we scrap from the Singapore Turf Club website:

1. Placing (Pl) – Final placing in the race
2. Horse Name
3. Bar – Gate for the horse to race
4. Track – Race Distance
5. Gear – Things like reins, whips, crops and hoods allow jockey to get the horse run faster
6. Rating (Rtg) – A horse assign to determine the type of the race to be compete
7. Jockey – Person who rides the horse in horse racing
8. Trainer – Person who train the horses for the racing
9. Owner – Person who owns the stable & the horse
10. Last 800m – Horse Placing for final 800m distance towards the end
11. Last 400m – Horse Placing for final 400m distance towards the end
12. Final – Final placing in the race
13. Second – Time different in seconds between the winner of the race
14. Total second to finish (time-sec) – Time to finish the race

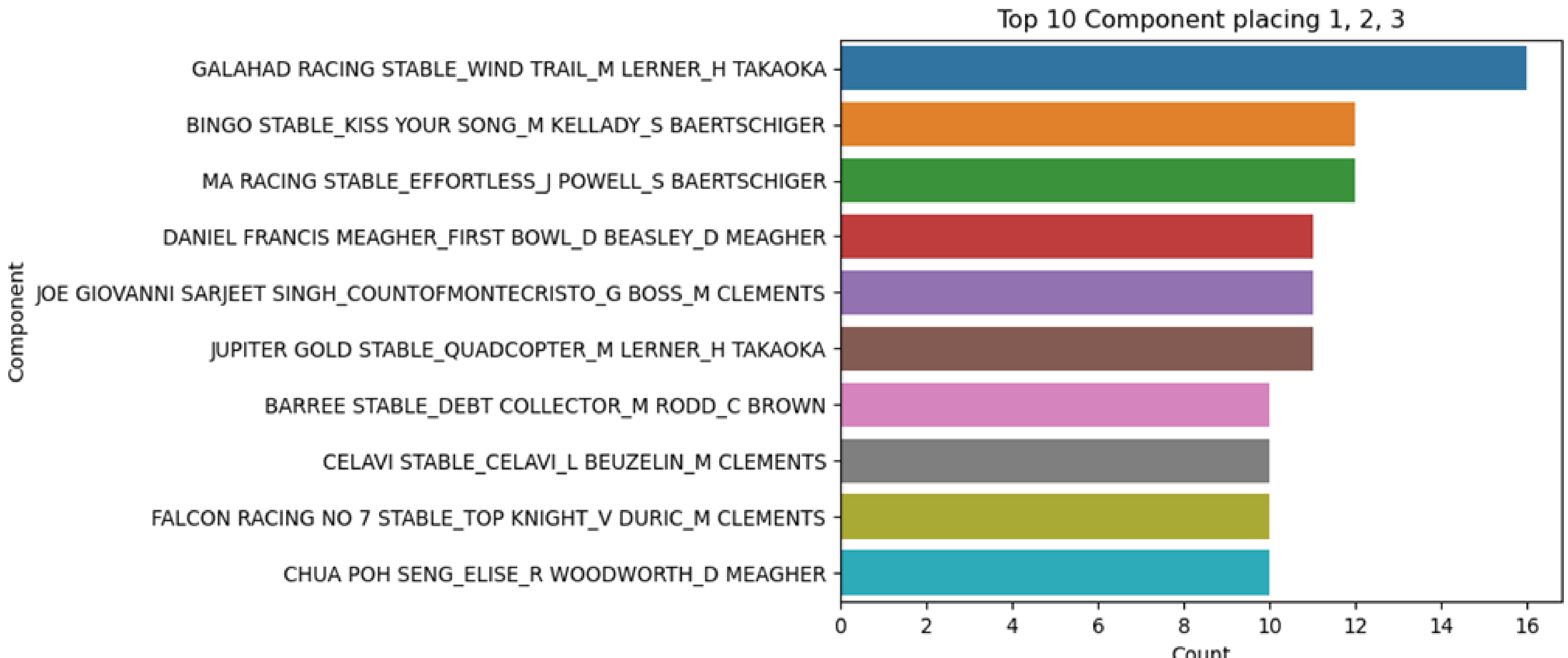
* yellow in highlight will be selected for analysis and ML purpose.

STAGE 2: FEATURE ENGINEERING

PART 1: EXPLORATORY DATA ANALYSIS (EDA)

Component = Owner + Horse Name + Jockey + Trainer

From the bar chart, although the component is different where they win the top 3 places, there is some similarity between trainer.



STAGE 2: FEATURE ENGINEERING

PART 1: EXPLORATORY DATA ANALYSIS (EDA)

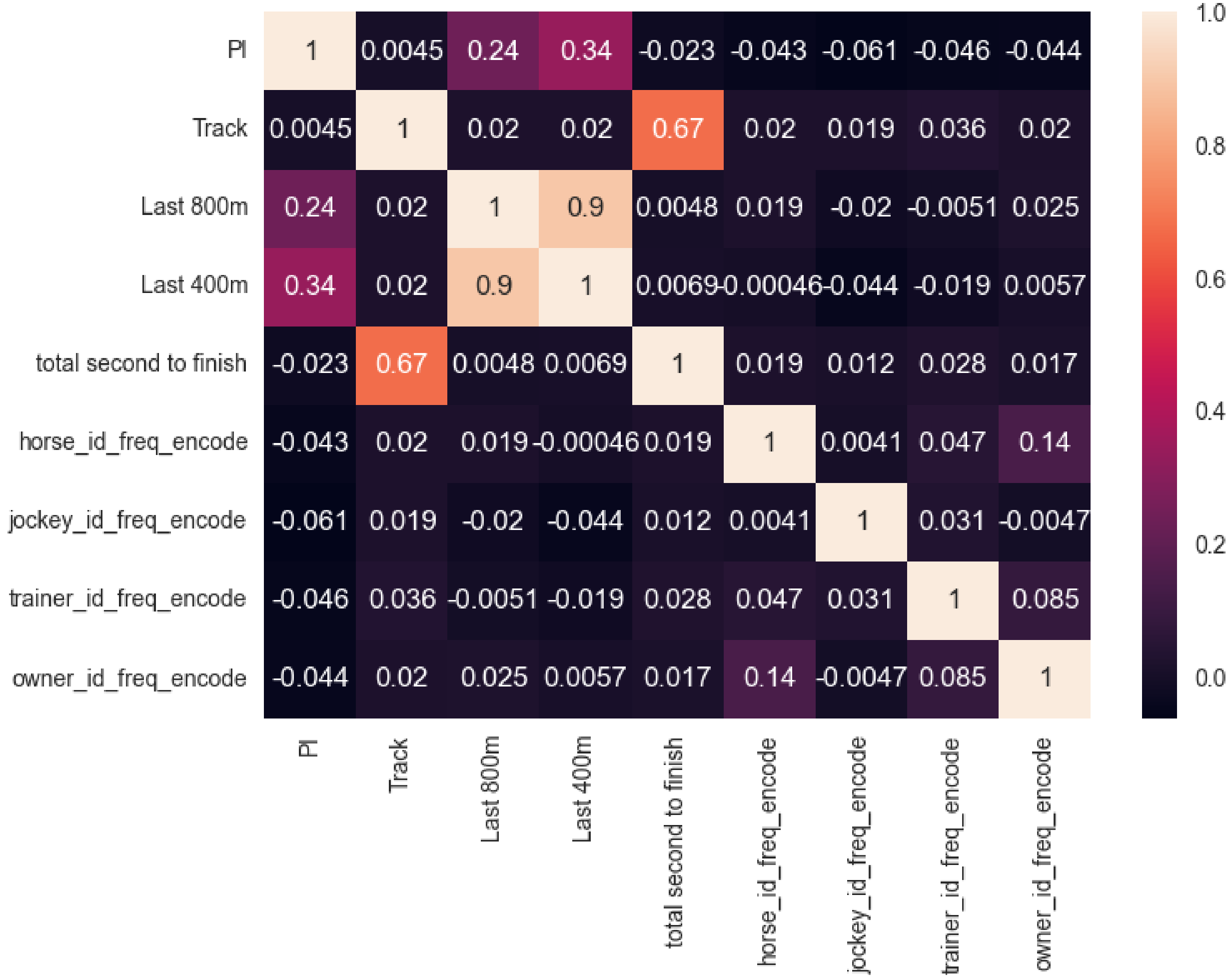
Component = Owner + Horse Name + Jockey + Trainer

Relationship between 4 components:

1. Horse not only have 1 owner only as they might be trade for breeding
2. Trainer follow the horses when they change the owner.
3. Owner is correlated to trainer as they hire trainer to train the horse.
4. Jockey is the independent features that not correlated to each components as they can ride any horses during the race from different owners.

STAGE 2: FEATURE ENGINEERING

PART 1: EXPLORATORY DATA ANALYSIS (EDA)



STAGE 2: FEATURE ENGINEERING

PART 2: LABEL ENCODING

Based on the strategy spoken previous slides, target ('Pl') change to binary labels encoding.

Split to 2 class

1. Top 3 ranking (Placing 1, 2 and 3)
2. Other (Placing after 4)

Change the target features ('Pl') to Binary Label Encoding

Top 3 Placing = 0
Placing 4 above = 1

```
df.loc[df.Pl <= 3,'Pl']= 0 #( top 3 ranking )
df.loc[df.Pl >= 4,'Pl']= 1 #( Other )
df.head(10)
```

✓ 0.2s

	Pl	horse_name	Track	jockey	trainer	Owner	Last 800m	Last 400m	total second to finish
0	1	A LA VICTORY	1200	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0
1	1	A LA VICTORY	1200	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0
2	1	A LA VICTORY	1200	SY MOON	J PETERS	CHINA HORSE CLUB STABLE	8.0	7.0	70.0
3	0	A LOT IN HAND	1100	APP CK NG	HK TAN	HAPPY LIFE STABLE	3.0	3.0	67.0
4	1	A LOT IN HAND	1200	TH KOH	HK TAN	HAPPY LIFE STABLE	2.0	2.0	74.0
5	1	A LOT IN HAND	1200	M EWE	HK TAN	HAPPY LIFE STABLE	2.0	2.0	73.0
6	0	A LOT IN HAND	1200	M EWE	HK TAN	HAPPY LIFE STABLE	1.0	1.0	73.0
7	1	A LOT IN HAND	1200	APP CK NG	HK TAN	HAPPY LIFE STABLE	2.0	3.0	73.0
8	1	A LOT IN HAND	1100	D DAVID	HK TAN	HAPPY LIFE STABLE	1.0	2.0	66.0
9	1	AABIR	1000	V DURIC	M WALKER	JOHN ERIC GILES GALVIN	7.0	8.0	62.0

STAGE 2: FEATURE ENGINEERING

Most of algorithms cannot handle categorical variable, so we will convert the them to numeric values.

For the dataset we have 4 feature is categoric variable :

1. Horse name (Nominal data)
2. Jockey (Nominal data)
3. Trainer (Nominal data)
4. Owner (Nominal data)

This 4 feature is individual and can't compare each other, so we will use the **frequency encoding method** convert them to numeric value.

Step 1 – Convert them to ID (string)

	Pl	horse_name	Track	jockey	trainer	Owner	Last 800m	Last 400m	total second to finish	horse_id	jockey_id	trainer_id	owner_id
0	2	A LA VICTORY	1.2	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0	0001	001	001	001
1	2	A LA VICTORY	1.2	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0	0001	001	001	001
2	2	A LA VICTORY	1.2	SY MOON	J PETERS	CHINA HORSE CLUB STABLE	8.0	7.0	70.0	0001	002	001	001
3	1	A LOT IN HAND	1.1	APP CK NG	HK TAN	HAPPY LIFE STABLE	3.0	3.0	67.0	0002	003	002	002
4	2	A LOT IN HAND	1.2	TH KOH	HK TAN	HAPPY LIFE STABLE	2.0	2.0	74.0	0002	004	002	002

Step 2 – Convert them to numeric by using Frequency Encoding method

	Pl	horse_name	Track	jockey	trainer	Owner	Last 800m	Last 400m	total second to finish	horse_id_freq_encode	jockey_id_freq_encode	trainer_id_freq_encode	owner_id_freq_encode
0	2	A LA VICTORY	1.2	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0	0.000066	0.034776	0.036676	0.004507
1	2	A LA VICTORY	1.2	K A'ISISUHAIRI	J PETERS	CHINA HORSE CLUB STABLE	8.0	9.0	71.0	0.000066	0.034776	0.036676	0.004507
2	2	A LA VICTORY	1.2	SY MOON	J PETERS	CHINA HORSE CLUB STABLE	8.0	7.0	70.0	0.000066	0.001812	0.036676	0.004507
3	1	A LOT IN HAND	1.1	APP CK NG	HK TAN	HAPPY LIFE STABLE	3.0	3.0	67.0	0.000133	0.011666	0.005060	0.001016

Beside that, minimize Track features value in order to scale the value with other features.

```
# Minimize Track features value in order to scale the value with other features
df['Track'] = df['Track'].div(1000)
```

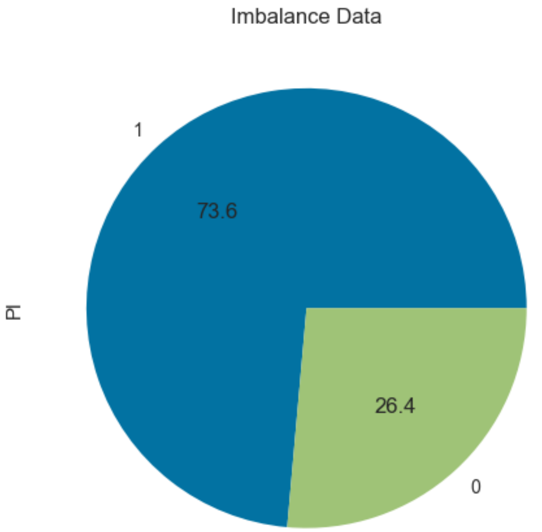
✓ 0.4s

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 45261 entries, 0 to 45471
Data columns (total 13 columns):
 #   Column                  Non-Null Count  Dtype
---  ------                  --------------  -----
 0   Pl                      45261 non-null  int64
 1   horse_name              45261 non-null  object
 2   Track                   45261 non-null  float64
 3   jockey                  45261 non-null  object
 4   trainer                 45261 non-null  object
 5   Owner                   45261 non-null  object
 6   Last 800m               45261 non-null  float64
 7   Last 400m               45261 non-null  float64
 8   total second to finish  45261 non-null  float64
 9   horse_id_freq_encode    45261 non-null  float64
 10  jockey_id_freq_encode   45261 non-null  float64
 11  trainer_id_freq_encode  45261 non-null  float64
 12  owner_id_freq_encode    45261 non-null  float64
dtypes: float64(8), int64(1), object(4)
memory usage: 4.8+ MB
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 1:

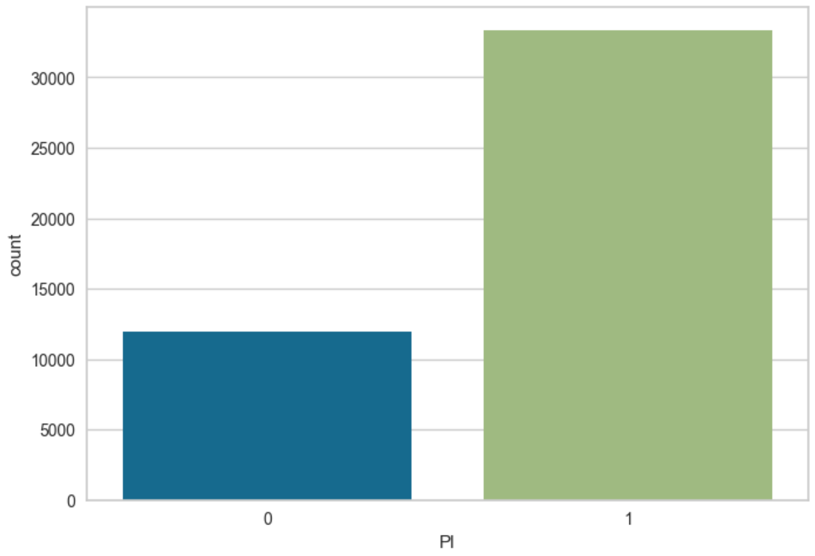
Before choosing which model to build, check the data whether imbalance or balance.




```
# To check target whether have imbalance data
y.value_counts()
```

✓ 0.8s

```
1    33311
0    11950
Name: Pl, dtype: int64
```



STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 2:

Known that data is imbalance. Use Pycaret to compare model selection with accuracy as the target to select model with imbalance data, fixed imbalance data method (SMOTE, RandomOverSampler (ROS) & RandomUnderSampler (RUS))

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
lightgbm	Light Gradient Boosting Machine	0.7797	0.8093	0.9126	0.8115	0.8591	0.3619	0.3754	0.2400
xgboost	Extreme Gradient Boosting	0.7742	0.8062	0.9011	0.8125	0.8545	0.3562	0.3659	0.9900
gbc	Gradient Boosting Classifier	0.7738	0.8026	0.9281	0.7976	0.8579	0.3182	0.3415	4.1100
ada	Ada Boost Classifier	0.7701	0.7889	0.9275	0.7945	0.8559	0.3044	0.3280	1.2100
rf	Random Forest Classifier	0.7635	0.7816	0.9050	0.7999	0.8493	0.3106	0.3237	1.3500
et	Extra Trees Classifier	0.7610	0.7681	0.9091	0.7954	0.8485	0.2950	0.3103	1.0600
lr	Logistic Regression	0.7579	0.7755	0.9193	0.7874	0.8482	0.2672	0.2880	0.2900
lda	Linear Discriminant Analysis	0.7556	0.7743	0.9546	0.7690	0.8518	0.1981	0.2438	0.0800
knn	K Neighbors Classifier	0.7494	0.7312	0.8697	0.8053	0.8363	0.3055	0.3096	0.1500
ridge	Ridge Classifier	0.7473	0.5383	0.9812	0.7515	0.8511	0.1054	0.1743	0.0700
qda	Quadratic Discriminant Analysis	0.7369	0.7586	0.9233	0.7668	0.8378	0.1719	0.1951	0.0600
dummy	Dummy Classifier	0.7360	0.5000	1.0000	0.7360	0.8479	0.0000	0.0000	0.0400
nb	Naive Bayes	0.7154	0.7287	0.7876	0.8188	0.8029	0.2917	0.2924	0.0500
dt	Decision Tree Classifier	0.6956	0.6116	0.7895	0.7954	0.7924	0.2218	0.2218	0.2800
svm	SVM - Linear Kernel	0.6361	0.7006	0.5638	0.9062	0.6952	0.2999	0.3550	1.0500

Imbalance Data
Highest Accuracy: 77.97%

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
xgboost	Extreme Gradient Boosting	0.7754	0.8040	0.8897	0.8203	0.8536	0.3747	0.3805	4.0500
lightgbm	Light Gradient Boosting Machine	0.7716	0.8024	0.8697	0.8285	0.8486	0.3850	0.3869	0.7200
rf	Random Forest Classifier	0.7464	0.7743	0.8189	0.8337	0.8262	0.3577	0.3579	2.9600
et	Extra Trees Classifier	0.7457	0.7652	0.8230	0.8301	0.8265	0.3506	0.3506	1.5500
gbc	Gradient Boosting Classifier	0.7315	0.7815	0.7640	0.8557	0.8073	0.3695	0.3760	9.3300
ada	Ada Boost Classifier	0.7029	0.7613	0.7154	0.8573	0.7800	0.3341	0.3473	2.4600
knn	K Neighbors Classifier	0.6984	0.7271	0.7269	0.8416	0.7801	0.3081	0.3166	0.1400
lr	Logistic Regression	0.6897	0.7694	0.6729	0.8769	0.7615	0.3391	0.3642	0.8900
lda	Linear Discriminant Analysis	0.6847	0.7709	0.6597	0.8821	0.7549	0.3378	0.3668	0.1400
ridge	Ridge Classifier	0.6827	0.7045	0.6584	0.8804	0.7534	0.3338	0.3624	0.0600
dt	Decision Tree Classifier	0.6782	0.6202	0.7428	0.8049	0.7726	0.2252	0.2274	0.4300
qda	Quadratic Discriminant Analysis	0.6702	0.7555	0.6416	0.8774	0.7412	0.3158	0.3463	0.0900
nb	Naive Bayes	0.6174	0.7210	0.5626	0.8721	0.6840	0.2524	0.2935	0.0600
svm	SVM - Linear Kernel	0.4436	0.6098	0.2576	0.9498	0.4052	0.1330	0.2422	2.3800
dummy	Dummy Classifier	0.2640	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.1000

Fixed Imbalance Data
SMOTE Method
Highest Accuracy: 77.54%

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 2:

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
et	Extra Trees Classifier	0.7681	0.7770	0.9156	0.7987	0.8532	0.3134	0.3307	2.0400
rf	Random Forest Classifier	0.7615	0.7870	0.8512	0.8292	0.8401	0.3713	0.3718	2.2000
xgboost	Extreme Gradient Boosting	0.7389	0.8054	0.7484	0.8789	0.8084	0.4069	0.4200	2.3900
lightgbm	Light Gradient Boosting Machine	0.7246	0.8095	0.7134	0.8907	0.7923	0.3985	0.4206	0.5300
gbc	Gradient Boosting Classifier	0.7115	0.8017	0.6935	0.8903	0.7797	0.3802	0.4058	10.1600
ada	Ada Boost Classifier	0.7009	0.7879	0.6828	0.8844	0.7706	0.3605	0.3862	3.0600
dt	Decision Tree Classifier	0.6977	0.6188	0.7858	0.7999	0.7928	0.2342	0.2344	0.3800
lr	Logistic Regression	0.6888	0.7748	0.6660	0.8823	0.7590	0.3431	0.3711	0.5800
lda	Linear Discriminant Analysis	0.6852	0.7757	0.6567	0.8862	0.7544	0.3423	0.3732	0.2200
ridge	Ridge Classifier	0.6833	0.7089	0.6547	0.8851	0.7527	0.3390	0.3699	0.0700
knn	K Neighbors Classifier	0.6719	0.7248	0.6726	0.8503	0.7511	0.2887	0.3062	0.2200
qda	Quadratic Discriminant Analysis	0.6675	0.7571	0.6263	0.8892	0.7349	0.3233	0.3606	0.1100
nb	Naive Bayes	0.6148	0.7274	0.5553	0.8758	0.6797	0.2530	0.2967	0.0600
svm	SVM - Linear Kernel	0.5199	0.6535	0.3705	0.9420	0.5318	0.1990	0.2983	2.9000
dummy	Dummy Classifier	0.2640	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0700

	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa	MCC	TT (Sec)
svm	SVM - Linear Kernel	0.7568	0.5891	0.9445	0.7746	0.8511	0.2217	0.2589	0.4300
lightgbm	Light Gradient Boosting Machine	0.7191	0.8079	0.7012	0.8942	0.7860	0.3940	0.4194	0.1900
xgboost	Extreme Gradient Boosting	0.7184	0.8011	0.7016	0.8928	0.7857	0.3918	0.4166	0.5600
gbc	Gradient Boosting Classifier	0.7095	0.8009	0.6881	0.8925	0.7771	0.3795	0.4067	2.4300
rf	Random Forest Classifier	0.6992	0.7813	0.6824	0.8822	0.7695	0.3562	0.3813	0.7500
et	Extra Trees Classifier	0.6977	0.7721	0.6841	0.8782	0.7691	0.3504	0.3740	0.6700
ada	Ada Boost Classifier	0.6965	0.7834	0.6797	0.8807	0.7673	0.3514	0.3765	0.9200
knn	K Neighbors Classifier	0.6893	0.7462	0.6858	0.8640	0.7647	0.3249	0.3442	0.0700
lr	Logistic Regression	0.6886	0.7741	0.6668	0.8811	0.7591	0.3417	0.3692	0.2300
lda	Linear Discriminant Analysis	0.6839	0.7759	0.6576	0.8830	0.7538	0.3376	0.3673	0.0700
ridge	Ridge Classifier	0.6830	0.7061	0.6572	0.8821	0.7532	0.3358	0.3652	0.0400
qda	Quadratic Discriminant Analysis	0.6805	0.7528	0.6564	0.8789	0.7515	0.3296	0.3581	0.0500
dt	Decision Tree Classifier	0.6272	0.6300	0.6241	0.8270	0.7113	0.2133	0.2307	0.1700
nb	Naive Bayes	0.6213	0.7238	0.5680	0.8731	0.6883	0.2575	0.2982	0.0900
dummy	Dummy Classifier	0.2640	0.5000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0500

Fixed Imbalance Data
RandomOverSampler (ROS)
Highest Accuracy: 76.81%

Fixed Imbalance Data
RandomUnderSampler (RUS)
Highest Accuracy: 75.68%

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 2:

Pycaret Model Selection without cross validation

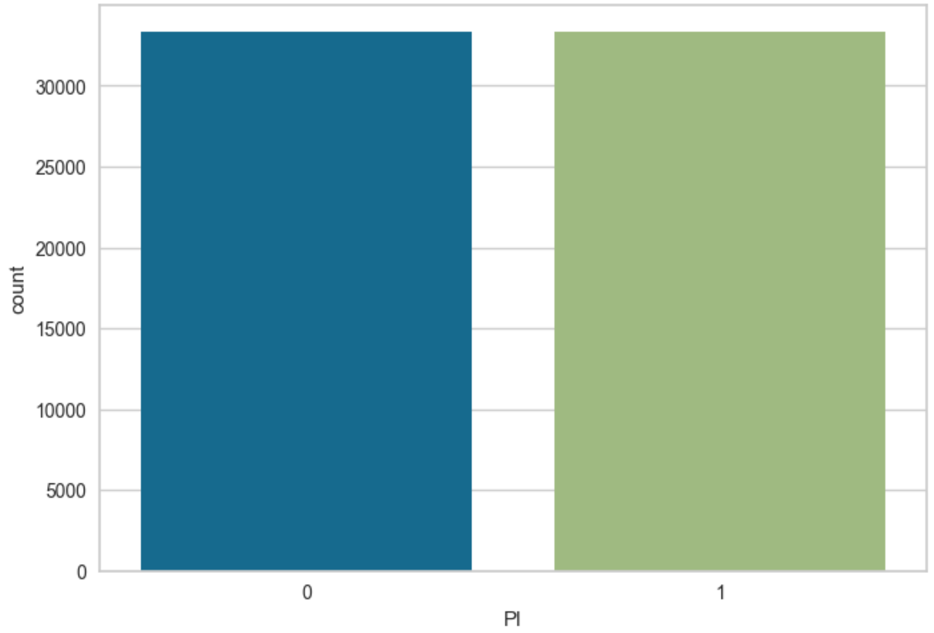
1. Imbalance Data (Highest accuracy:77.97%)
2. Fixed imbalance data SMOTE (Highest accuracy: 77.54%)
3. Fixed imbalance data ROS (Highest accuracy: 76.81%)
4. Fixed imbalance data RUS (Highest accuracy: 75.68%)

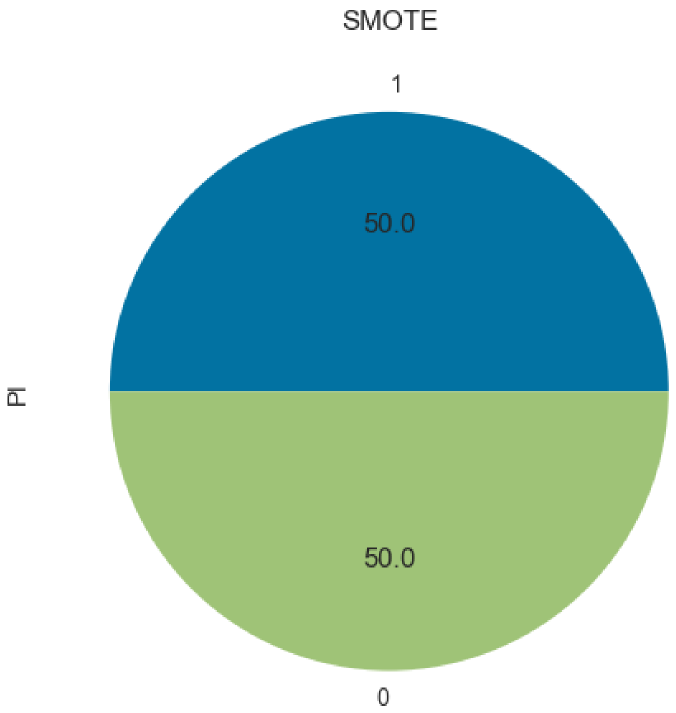
SMOTE method will use to fix the imbalance data.

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 3:

Fixed imbalanced data by using SMOTE method






```python
# Using SMOTE method to fix imbalance data
from imblearn.over_sampling import SMOTE

rus = SMOTE(random_state=111)

X_res , y_res = rus.fit_resample(X , y)

y_res.value_counts()
```

0.3s

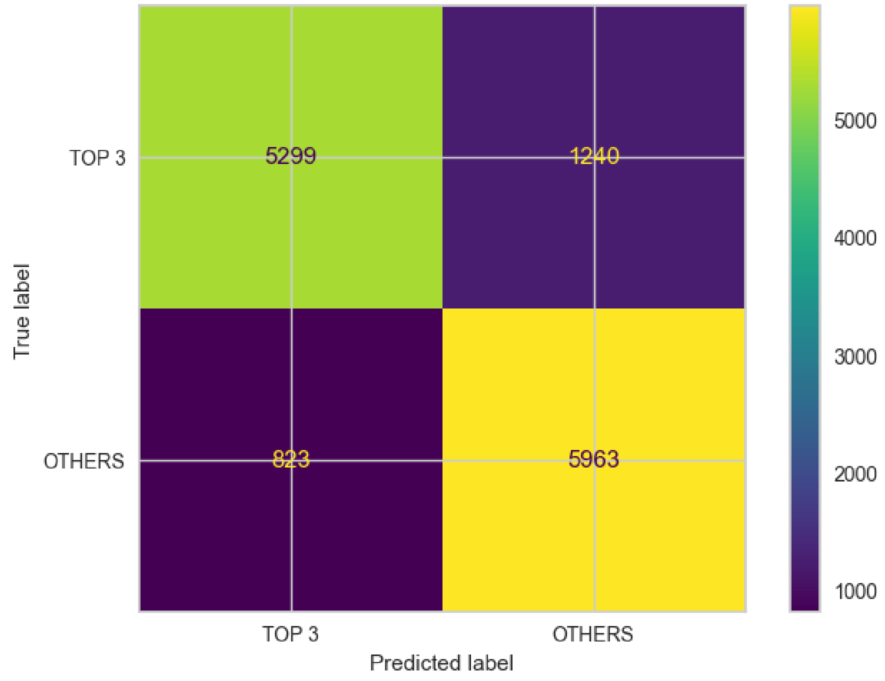
```
1    33311
0    33311
Name: PI, dtype: int64
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 4:

Hypothesis: Fixed imbalance data will give better accuracy for imbalance data.
To compare by selecting one model (xgboost) with imbalance data & fixed imbalance data (SMOTE).

Fixed Imbalance Data
SMOTE Method
Confusion Matrix Plot



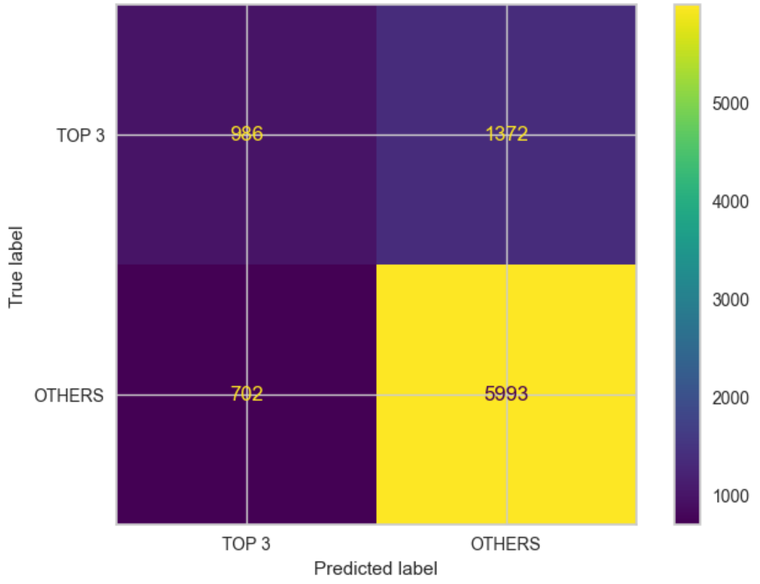

```
[[5299 1240]
 [ 823 5963]]
              precision    recall  f1-score   support

           0       0.87      0.81      0.84      6539
           1       0.83      0.88      0.85      6786

    accuracy                           0.85     13325
   macro avg       0.85      0.84      0.84     13325
weighted avg       0.85      0.85      0.84     13325

XGBoost Model:  0.8451782363977486
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 4:

Hypothesis: Fixed imbalance data will give better accuracy for imbalance data.
To compare by selecting one model (xgboost) with imbalance data & fixed imbalance data (SMOTE).

Imbalance Data
Confusion Matrix Plot




```
[[2955 3584]
 [ 556 6230]]
              precision    recall  f1-score   support

           0       0.84      0.45      0.59      6539
           1       0.63      0.92      0.75      6786

    accuracy                           0.69     13325
   macro avg       0.74      0.68      0.67     13325
weighted avg       0.74      0.69      0.67     13325

XGBoost Model with imbalance data:  0.6893058161350845
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 4:

Hypothesis: Fixed imbalance data will give better accuracy for imbalance data.

Hypothesis Correct!

Imbalance Data: 68.9%

Fixed Imbalance Data:84.5%

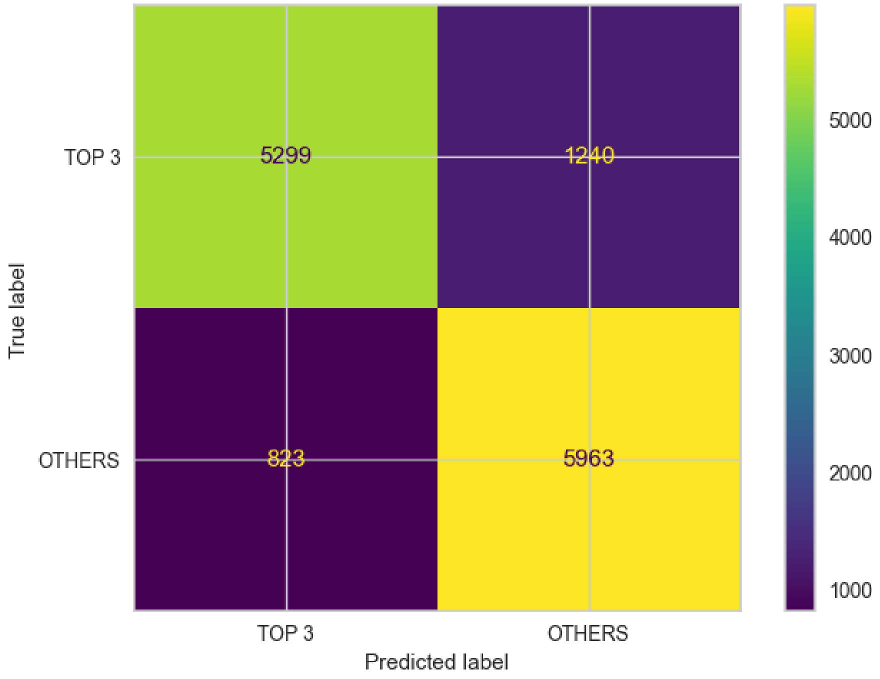
SELECT the top 5 modeling from pycaret under SMOTE method.
1. Extreme Gradient Boosting (xgboost)
2. Light Gradient Boosting (lightgbm)
3. Gradient Boosting Classifier (gbc)
4. Random Forest Classifier (rf)
5. Extra Trees Classifier (et)

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Top 5 modeling from pycaret under SMOTE method

1. Extreme Gradient Boosting (xgboost)



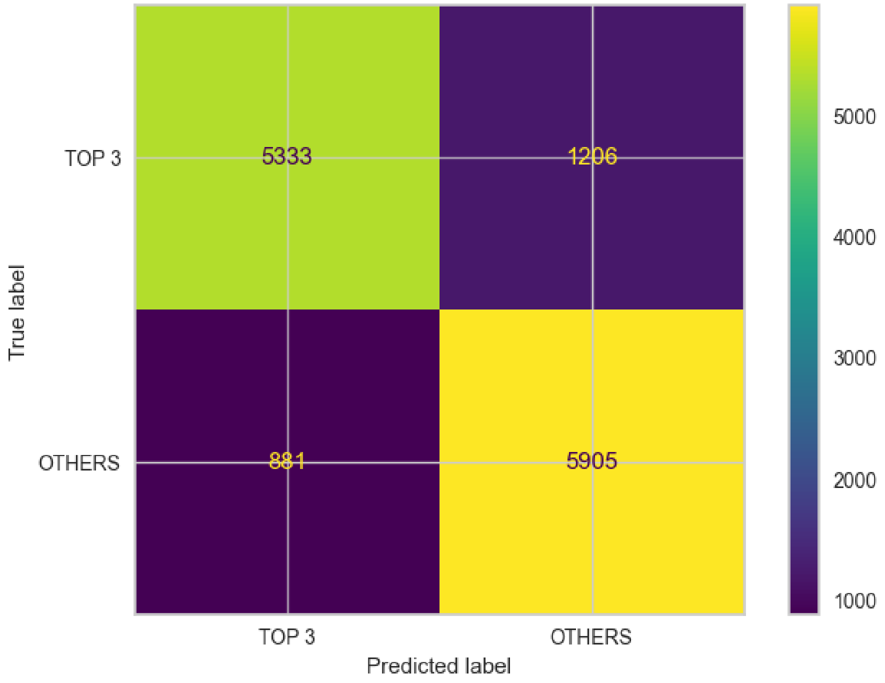

```
[[5299 1240]
 [ 823 5963]]
              precision    recall  f1-score   support

           0       0.87      0.81      0.84      6539
           1       0.83      0.88      0.85      6786

    accuracy                           0.85     13325
   macro avg       0.85      0.84      0.84     13325
weighted avg       0.85      0.85      0.84     13325

XGBoost Model:  0.8451782363977486
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Top 5 modeling from pycaret under SMOTE method

2. Light Gradient Boosting (lightgbm)



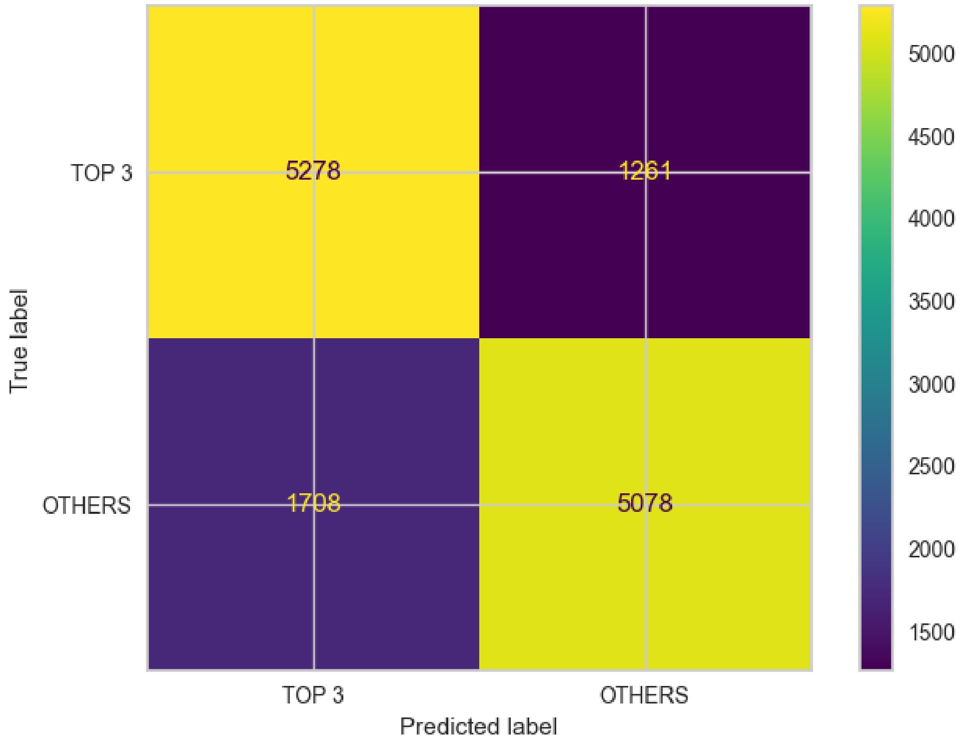

```
[[5333 1206]
 [ 881 5905]]
              precision    recall  f1-score   support

           0       0.86      0.82      0.84      6539
           1       0.83      0.87      0.85      6786

    accuracy                           0.84     13325
   macro avg       0.84      0.84      0.84     13325
weighted avg       0.84      0.84      0.84     13325

Light Gradient Boosting(lightgbm):  0.8433771106941839
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Top 5 modeling from pycaret under SMOTE method

3. Gradient Boosting Classifier (gbc)



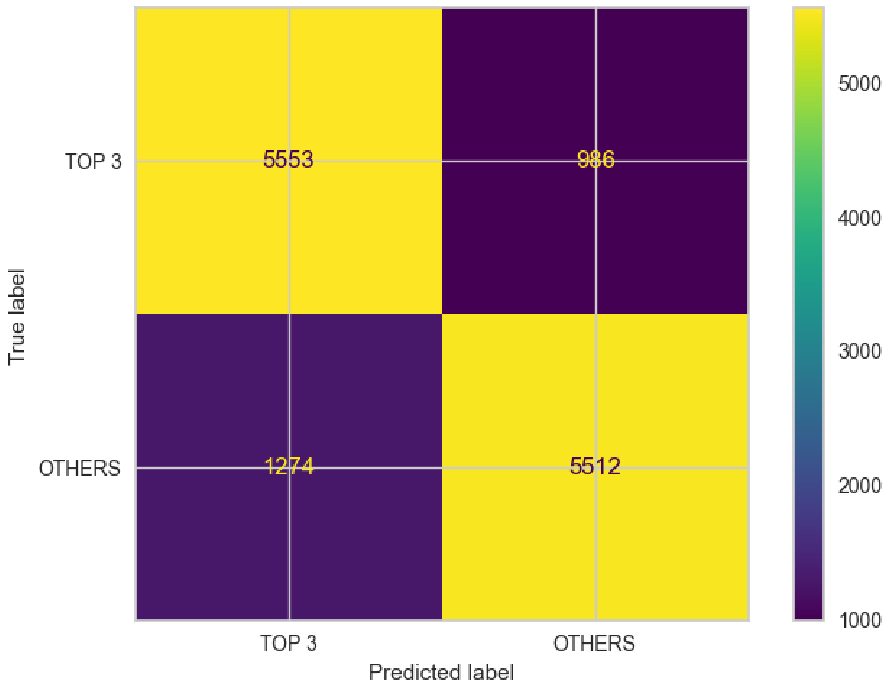

```
[[5278 1261]
 [1708 5078]]
              precision    recall  f1-score   support

           0       0.76      0.81      0.78      6539
           1       0.80      0.75      0.77      6786

    accuracy                           0.78     13325
   macro avg       0.78      0.78      0.78     13325
weighted avg       0.78      0.78      0.78     13325

Gradient Boosting Classification (gbc):  0.7771857410881801
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Top 5 modeling from pycaret under SMOTE method

4. Random Forest Classifier (rf)



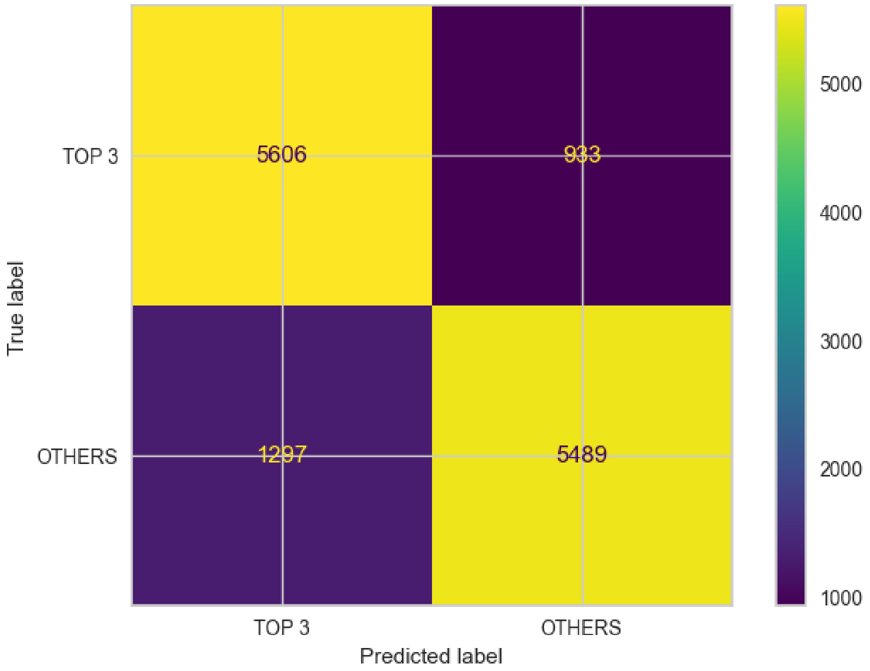

```
[[5553  986]
 [1274 5512]]
              precision    recall  f1-score   support

           0       0.81      0.85      0.83      6539
           1       0.85      0.81      0.83      6786

    accuracy                           0.83     13325
   macro avg       0.83      0.83      0.83     13325
weighted avg       0.83      0.83      0.83     13325

Random Forest Classification (rf):  0.8303939962476548
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Top 5 modeling from pycaret under SMOTE method

5. Extra Trees Classifier (et)




```
[[5606  933]
 [1297 5489]]
              precision    recall  f1-score   support

           0       0.81      0.86      0.83      6539
           1       0.85      0.81      0.83      6786

    accuracy                           0.83     13325
   macro avg       0.83      0.83      0.83     13325
weighted avg       0.83      0.83      0.83     13325

Extra Trees Classification (rf):  0.8326454033771107
```

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Overall models accuracy using fixed imbalance data SMOTE method

1. Extreme Gradient Boosting (xgboost) – 84.5%
2. Light Gradient Boosting (lightgbm) – 84.3%
3. Gradient Boosting Classifier (gbc) – 77.7%
4. Random Forest Classifier (rf) – 83.0%
5. Extra Trees Classifier (et) – 83.3%

EXTREME GRADIENT BOOSTING (XGBOOST) selected for hyper tuning model.

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 5:

Compare both emsemble model (Hard Voting & Stacking Classifier) & GridSearch CV by using highest accuracy on model selection (XGboost) to see whether can improve the accuracy.

1. HardVoting Classifier – 84.9%
2. **Stacking Classifier – 85.9%**
3. GridSearchCV(XGBoost)- 84.7%

```
Fitting 5 folds for each of 81 candidates, totalling 405 fits

{'eval_metric': 'error',
 'gamma': 0.5,
 'learning_rate': 0.3,
 'max_depth': 6,
 'scale_pos_weight': 1}
```

GridSearchCV parameters

```
from sklearn.ensemble import StackingClassifier

estimatorsSC= []
lightgbm_model = LGBMClassifier(random_state=111)
estimatorsSC.append(('lightgbm',lightgbm_model))

et_model = ExtraTreesClassifier(n_estimators=100, random_state=111)
estimatorsSC.append(('et', et_model))

rf_model = RandomForestClassifier(criterion='entropy', random_state=111)
estimatorsSC.append(('rf', rf_model))

sc = StackingClassifier(estimators= estimatorsSC, final_estimator=xgb.XGBClassifier(random_state=111))
```

Stacking Classifier parameters

```
# create the sub models
estimatorsHV = []

LIGHTBOOST_model = LGBMClassifier(random_state=111)
estimatorsHV.append(('Lightbooster', LIGHTBOOST_model))

XGBOOST_model = xgb.XGBClassifier(random_state=111)
estimatorsHV.append(('XGBOOST', XGBOOST_model))

RANDOMFOREST_model = RandomForestClassifier(criterion='entropy', random_state=111)
estimatorsHV.append(('Random forest', RANDOMFOREST_model))
```

HardVoting Classifier parameters

STAGE 3 & 4: MODEL BUILDING & PRODUCTION

STEP 6:

To check whether combine all 4 features into 1 ID or separate 4 IDs will give better accuracy.

Method:
Compare by using XGBoost with fixed imbalance data SMOTE method with accuracy.

Single ID: 77.4%
Multiple ID: 84.5%

Final model to use for horse prediction:
Stacking Classifier with XGBoost with multiple ID (85.9%)

CHALLENGES/TAKEAWAY

Challenges

1. Although the data easy to obtain, the scraping part with selenium more complicated that took lots of time to scrap it
2. Data limitation – no horse profile that able to scrap it to know the horse size & its heritage
3. In feature 400m & 800m that has higher correlation with the final placement, we can’t place the bet as the bet only open before the race. In the end we need to rely on the final placement of the race.

Takeaway

1. Selenium is suitable to scrap for large pools of datasets.
2. Pycaret is good to use for comparing models
3. Choose suitable features in datasets to do machine learning

GitHub link:
https://github.com/evansim85/Horse-Racing-Machine-Learning
https://github.com/Ansonex/Horse-_Racing_Machine_Learning