第三組期末專題報告

高念慈 曾宇謙 施永鴻

Advisor：徐浩雲 助教

目錄 :

- 資料介紹
- EDA
- 模型 :
  - *Logistic 、Decision Tree 、RandomForest*
  - *XGBoost*
  - *Light GBM*
- Submission 調整
- 總結

資料介紹

一份包含缺失值的資料，此資料已分好訓練集和測試集

	f1	f2	f3	f4	f5	f6	f7	f8	f9	f10	...	f110	f111	f112	f113	f114	f115	f116	f117	f118	claim
id																					
0	0.108590	0.004314	-37.566	0.017364	0.289150	-10.251000	135.120	168900.0	3.992400e+14	86.48900	...	-12.2280	1.7482	1.90960	-7.11570	4378.80	1.20960	8.613400e+14	140.100	1.017700	1
1	0.100900	0.299610	11822.000	0.276500	0.459700	-0.837330	1721.900	119810.0	3.874100e+15	9953.60000	...	-56.7580	4.1684	0.34808	4.14200	913.23	1.24640	7.575100e+15	1861.000	0.283590	0
2	0.178030	-0.006980	907.270	0.272140	0.459480	0.173270	2298.000	360650.0	1.224500e+13	15827.00000	...	-5.7688	1.2042	0.26290	8.13120	45119.00	1.17640	3.218100e+14	3838.200	0.406900	1
3	0.152360	0.007259	780.100	0.025179	0.519470	7.491400	112.510	259490.0	7.781400e+13	-36.83700	...	-34.8580	2.0694	0.79631	-16.33600	4952.40	1.17840	4.533000e+12	4889.100	0.514860	1
4	0.116230	0.502900	-109.150	0.297910	0.344900	-0.409320	2538.900	65332.0	1.907200e+15	144.12000	...	-13.6410	1.5298	1.14640	-0.43124	3856.50	1.48300	-8.991300e+12	NaN	0.230490	1
5	0.101530	-0.002612	-1118.700	0.116300	0.318860	-0.478390	2372.800	-1808.0	5.818100e+15	8421.20000	...	-44.0820	3.5812	26.55900	-6.47220	44570.00	1.17760	2.997700e+14	5548.300	-0.033159	0
6	0.003073	0.359530	20913.000	0.003465	0.268060	-1.225100	1301.200	233180.0	1.771400e+15	-48.08700	...	-39.9220	1.4896	1.29230	33.63400	16201.00	1.22710	2.458900e+16	1012.900	0.647970	0
7	0.097340	0.245910	11775.000	0.614860	0.479370	-1.486500	3179.200	123940.0	1.652900e+15	504.40000	...	-22.9950	1.5719	101.95000	1.51560	1117.50	1.24680	-6.971300e+13	1481.700	0.303170	1
8	-0.008948	0.338020	503.810	0.601520	0.261760	0.735380	1614.000	408030.0	7.684400e+14	5.38200	...	-9.5142	2.2176	-1.23750	7.07130	1030.80	1.20400	5.290300e+15	1968.100	0.145930	0
9	0.126230	0.173960	1662.000	0.038081	0.000485	0.059909	296.070	-8035.3	4.526300e+15	-90.30400	...	-1.9684	3.9595	19.16900	-0.24498	838.39	1.19680	2.567000e+17	4914.900	0.519760	0
10	0.121570	0.275670	NaN	0.006199	0.473720	0.582000	2044.100	1117000.0	3.895900e+13	43043.00000	...	-43.1580	3.6750	7.97710	-20.50900	69084.00	1.39610	7.217100e+16	523.190	0.862240	1
11	0.078650	0.387450	1025.900	0.082626	0.267070	-1.476200	2170.100	513560.0	9.716600e+12	0.08325	...	-17.1010	1.5954	121.02000	-9.33300	45667.00	1.24900	5.059300e+15	5422.300	0.570970	0
12	0.081278	0.302320	1341.000	0.516860	0.394170	-1.545600	105.600	1045900.0	-2.677900e+12	8658.60000	...	-1.7416	1.7956	-0.84064	-8.44710	6079.10	1.19260	1.759800e+16	6521.700	0.951500	1
13	0.071683	0.341330	342.300	-0.003108	0.004038	-1.058900	1126.500	126430.0	5.915300e+12	253.78000	...	-2.9630	1.1850	103.17000	30.20300	151650.00	1.20050	5.740900e+15	5203.900	0.688690	1
14	0.072488	0.473410	-22.274	0.002846	0.384460	3.548400	-40.267	739910.0	4.615400e+15	100.20000	...	-10.3290	1.6152	195.36000	1.05310	1806.40	1.22290	6.537300e+15	591.480	0.300380	1
15	0.090136	0.493140	474.410	0.538770	0.369940	3.216200	465.180	190230.0	2.518200e+14	-308.27000	...	-3.2032	2.1177	NaN	-2.63230	88379.00	1.17800	7.683900e+16	557.550	0.529460	1
16	0.092265	0.346540	3001.200	0.128750	0.252040	-0.869180	3813.600	379440.0	1.262800e+13	NaN	...	-44.4310	1.5249	62.55000	-5.82090	552.41	1.27550	2.267700e+16	10901.000	0.608180	1
17	0.120730	0.462630	1598.000	0.006041	0.506400	0.123370	1671.500	22930.0	-2.274400e+13	767.99000	...	-23.3520	3.9528	-1.92940	3.16690	11117.00	1.13690	4.649100e+15	441.280	0.332370	0
18	-0.007044	0.326940	6504.700	0.001551	0.264010	-1.476200	1590.200	415550.0	1.109700e+11	155.32000	...	-32.5430	3.7555	2.22790	9.45410	7990.20	0.99966	-4.167700e+13	861.700	0.245920	1
19	0.032643	0.414260	19891.000	-0.006275	0.250220	-3.508200	1260.100	504980.0	3.043600e+14	-7.52310	...	-1.6408	1.7626	-0.01899	-12.53700	8205.00	1.24120	4.698100e+14	1292.000	0.341880	1
20	0.078008	0.471140	7017.600	0.062191	0.516410	0.179820	438.310	769840.0	NaN	1518.70000	...	-7.9052	1.3450	-2.13500	-5.95510	72442.00	1.14820	6.056800e+15	1524.200	NaN	0
21	0.098638	0.309120	4516.300	0.001562	0.355220	0.342650	1580.100	21719.0	9.080200e+15	56161.00000	...	-18.0790	1.5900	46.39300	5.66130	5984.10	1.36550	3.135200e+16	3306.900	0.710050	1
22	0.036299	-0.012898	4560.900	0.282810	0.340390	-0.517280	282.300	611440.0	7.980700e+15	3057.50000	...	-2.4700	1.7053	70.76100	-8.50040	50133.00	1.14320	-2.067100e+14	2090.100	0.178680	0
23	0.051423	0.267750	5967.000	0.000400	0.384100	-0.729910	38.704	839230.0	6.770400e+14	860.01000	...	-7.8891	1.4308	161.49000	-12.70000	20696.00	1.17060	6.388800e+16	319.210	1.496900	0
24	0.144320	0.298780	2101.000	0.294040	0.436950	-4.987600	2031.000	635340.0	3.744900e+15	22.19500	...	-5.5846	1.9572	3.14770	4.92940	31156.00	1.19810	2.337600e+16	82.334	0.161020	0

25 rows × 119 columns

文件說明(客戶共1451393位):

- 訓練集 : 人數957919人

  id : 客戶代號

  f1 ~ f118 : 變數，共 118 個

  claim : 結果 : 1 提出索賠

  0 不提出索賠

- 標籤 : 提或不提出索賠 1 或 0 (基本事實)

文件說明(客戶共1451393位):

- 測試集 : 人數493474人

  id : 客戶代號

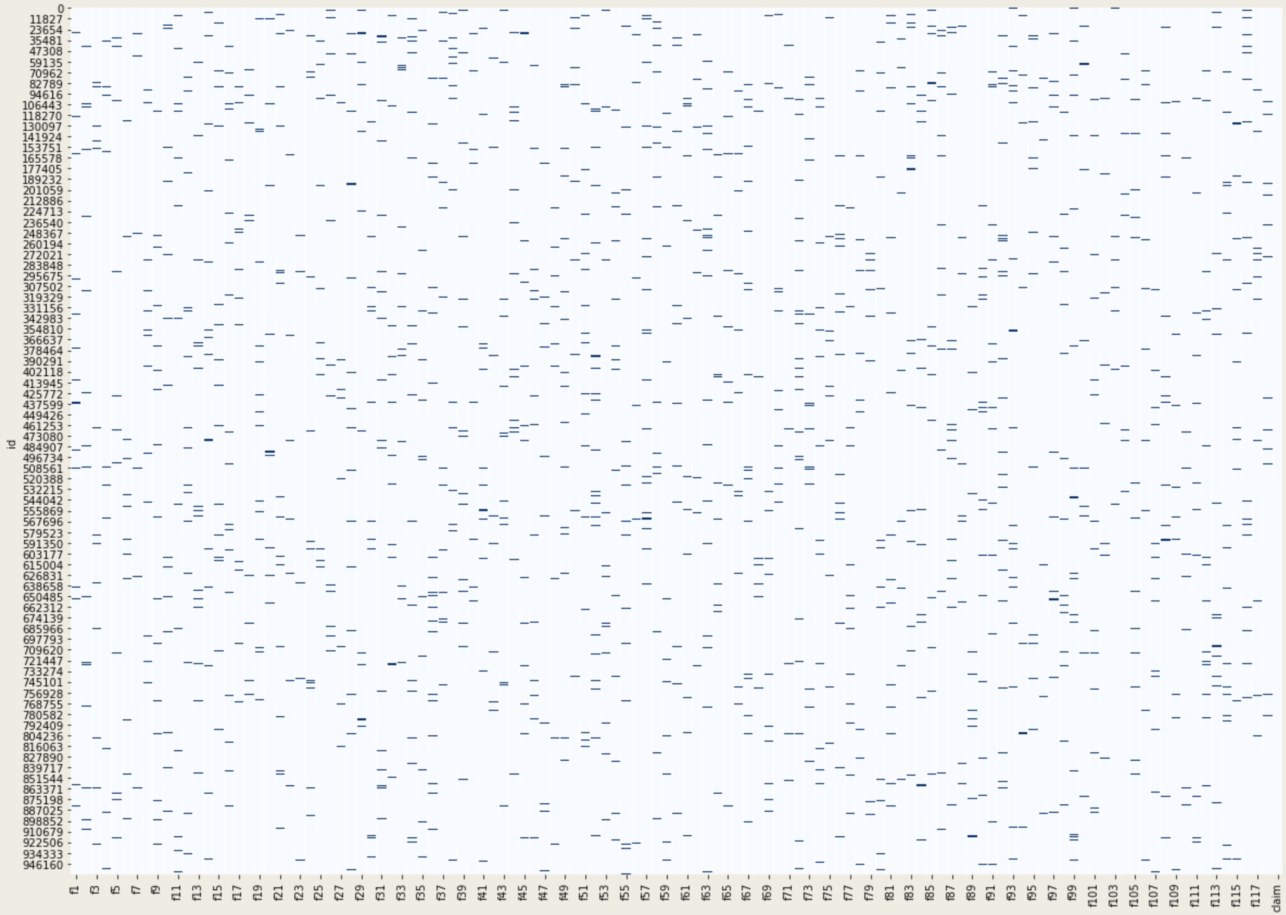
  f1 ~ f118 : 變數，共 118 個

- 標籤 : 為此次目標，將預測每位客戶提出索賠的機率 (介於 0 ~ 1)

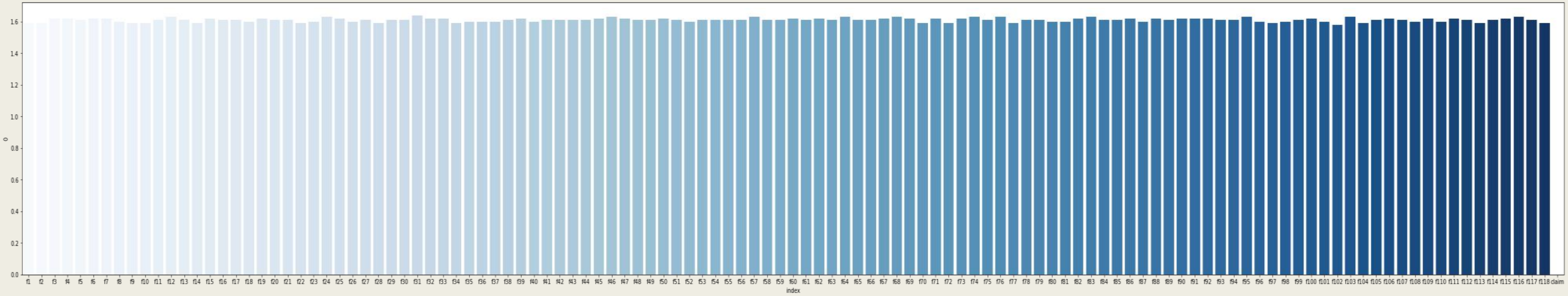
EDA

探索性資料分析

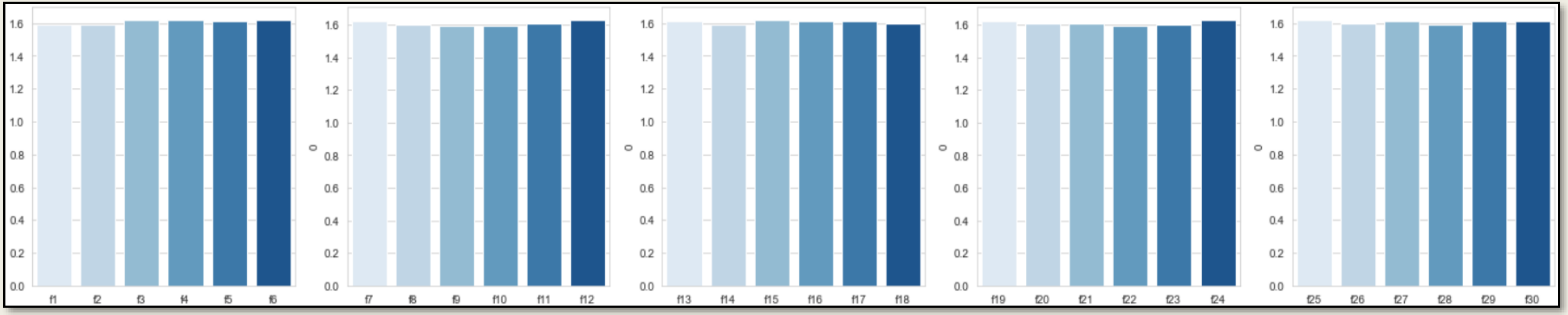
1. 資料是否存在缺失值 (整體：1.6%)



2. 各特徵缺失值比例

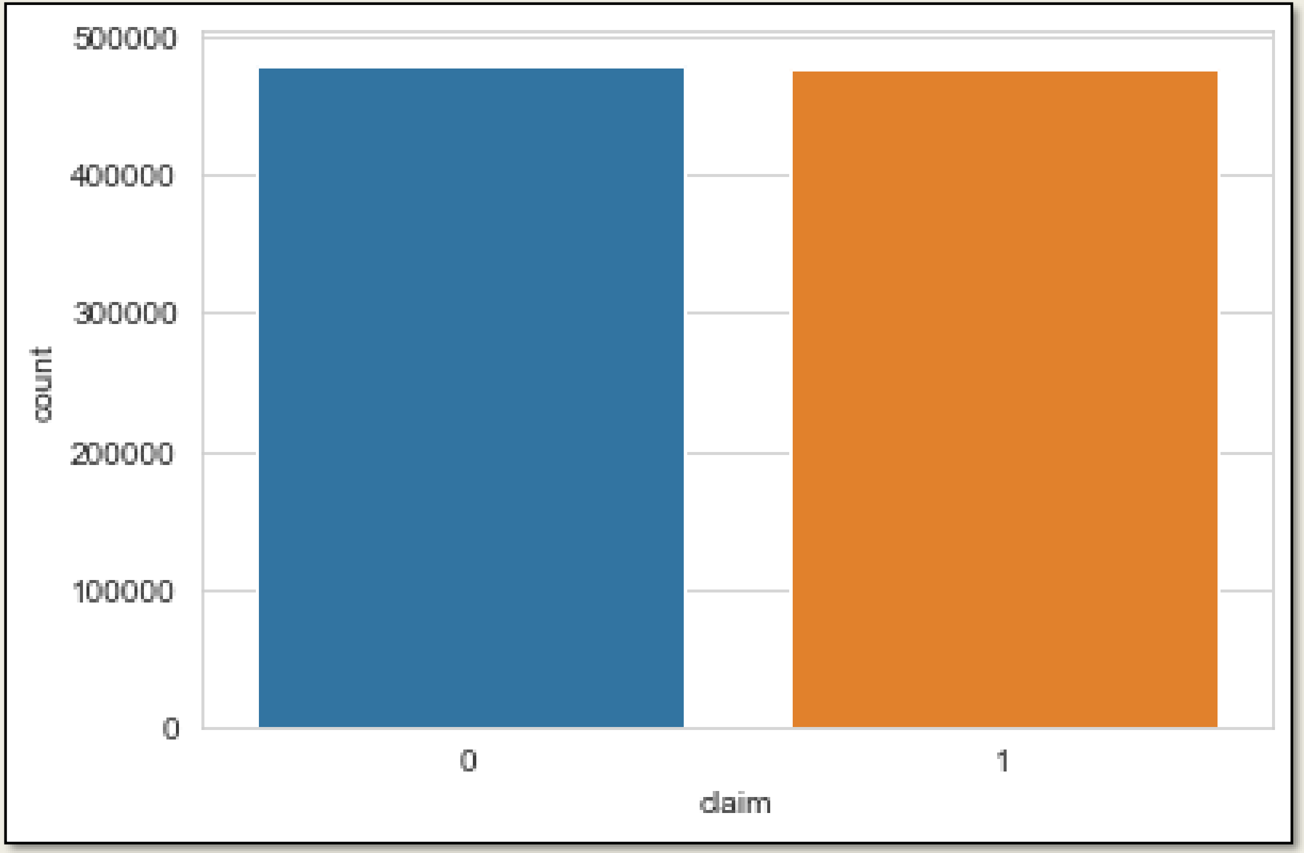


index



f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12
f13
f14
f15
f16
f17
f18
f19
f20
f21
f22
f23
f24
f25
f26
f27
f28
f29
f30

3. 目標類別是否不平衡

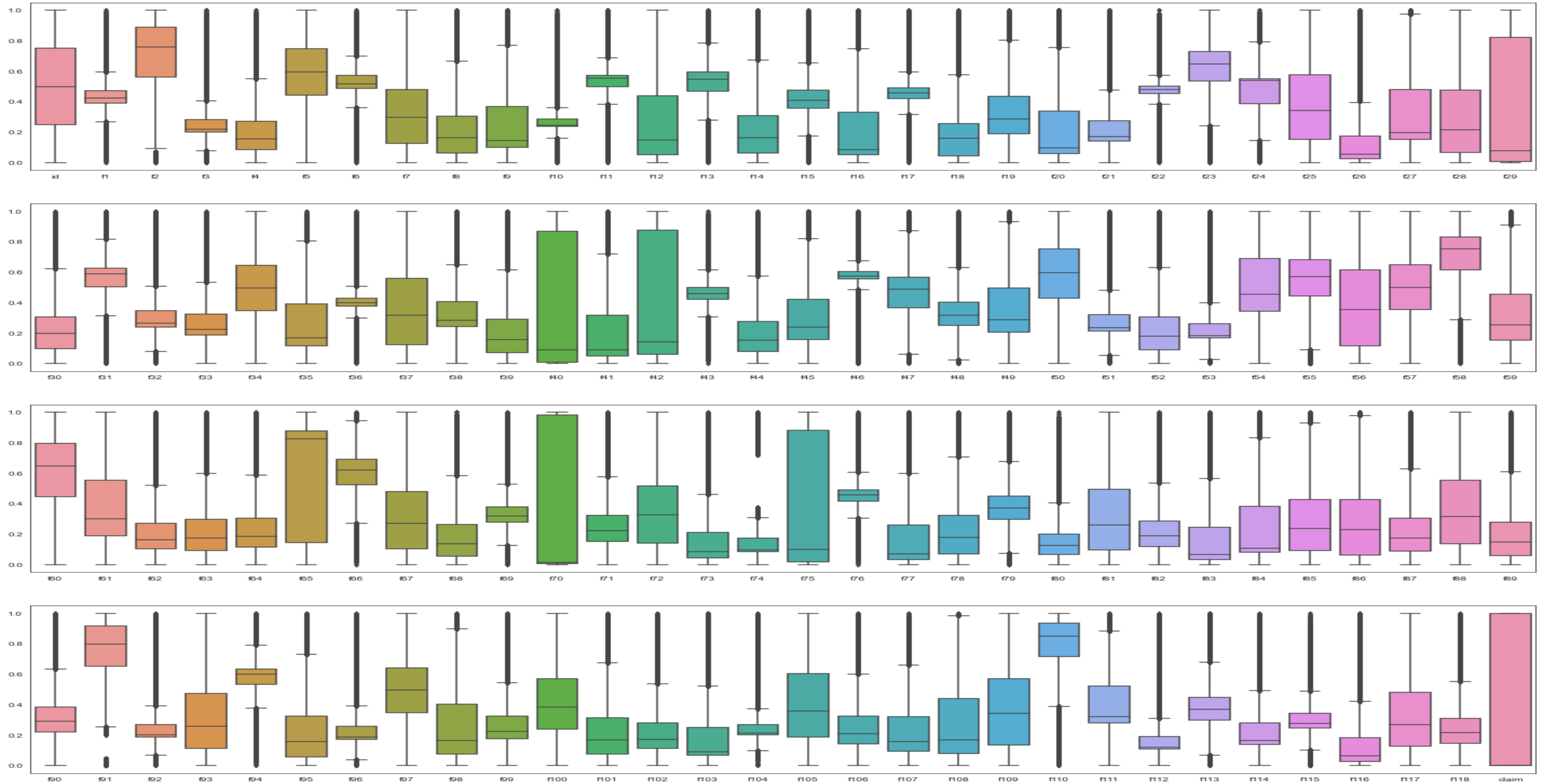


4. 統計性質

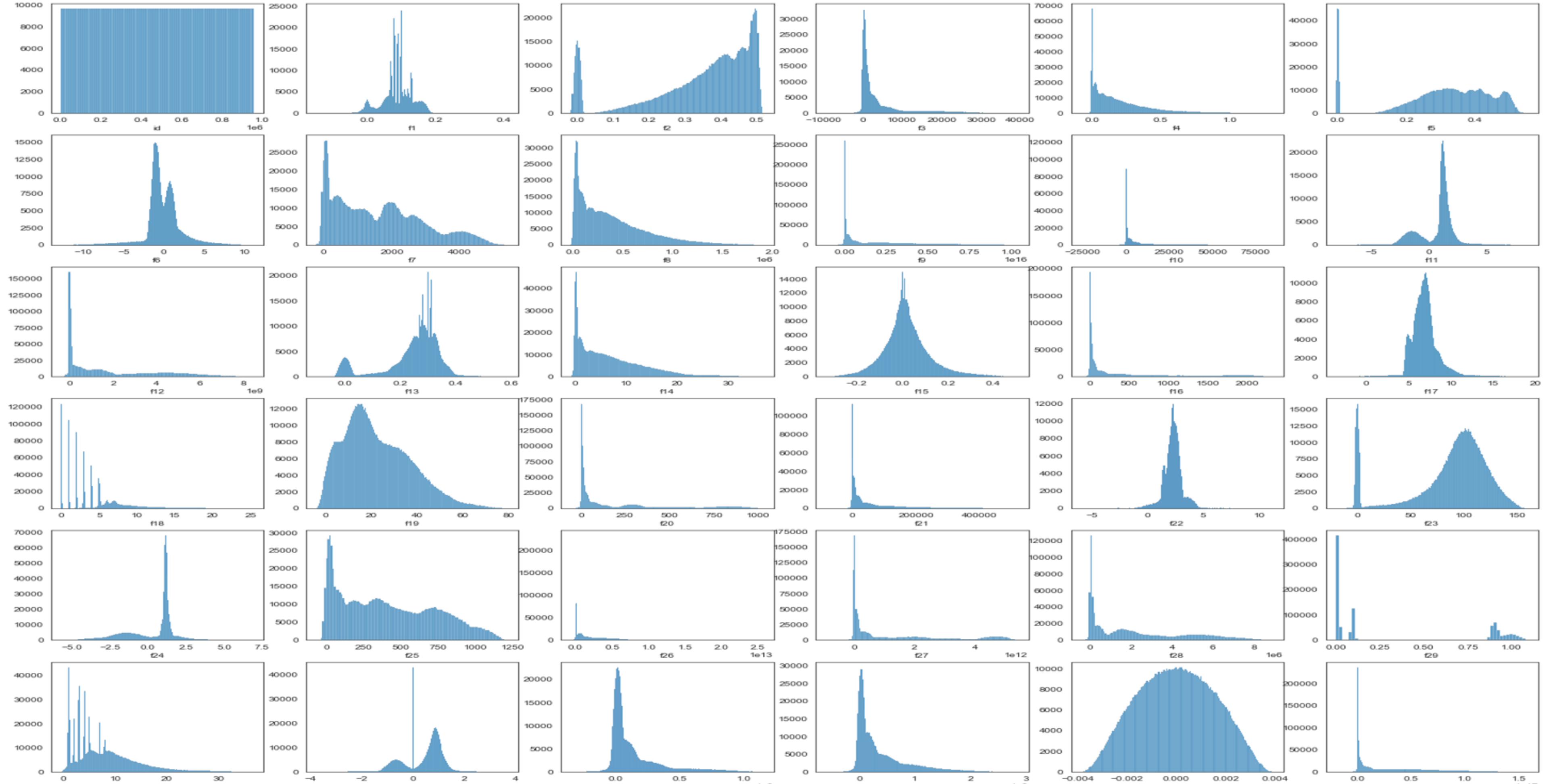
	f1	f2	f3	f4	f5	f6	f7	f8	f9
count	942672.000000	942729.000000	942428.000000	942359.000000	942514.000000	942398.000000	942415.000000	942546.000000	942670.000000
mean	0.090201	0.345964	4068.744207	0.201214	0.304869	-0.071458	1620.843815	377164.164157	1806053749440377.750000
std	0.043564	0.146251	6415.829440	0.212510	0.145343	2.123777	1276.281403	345432.472849	2335204188640509.000000
min	-0.149910	-0.019044	-9421.700000	-0.082122	-0.006990	-12.791000	-224.800000	-29843.000000	-1153300000000000.000000
25%	0.070227	0.283050	418.430000	0.035086	0.240520	-1.120700	481.545000	91209.000000	11531000000000.000000
50%	0.090135	0.389100	1279.500000	0.137000	0.327790	-0.380110	1446.100000	289670.000000	504305000000000.000000
75%	0.116500	0.458450	4444.400000	0.297100	0.412830	0.921940	2495.900000	560560.000000	3103100000000000.000000
max	0.415170	0.518990	39544.000000	1.319900	0.554750	11.202000	5426.600000	1913700.000000	10424000000000000.000000

f111	f112	f113	f114	f115	f116	f117	f118	claim
942420.000000	942509.000000	942686.000000	942481.000000	942360.000000	942330.000000	942512.000000	942707.000000	957919.000000
2.074530	23.885245	1.748777	63152.973540	1.208876	42769052891229504.000000	3959.204669	0.559267	0.498492
0.895793	45.581360	10.088848	92435.016241	0.114959	67324411404429680.000000	3155.991777	0.408426	0.499998
0.277040	-27.691000	-26.589000	-81977.000000	0.905270	-8944400000000000.000000	-415.240000	-0.151240	0.000000
1.487700	-0.628880	-4.473975	2443.200000	1.146800	232110000000000.000000	1306.200000	0.276560	0.000000
1.662100	1.727700	0.885710	19479.000000	1.177200	13275000000000000.000000	3228.000000	0.473440	0.000000
2.522325	18.991000	6.840775	88488.000000	1.242000	52787000000000000.000000	6137.900000	0.746210	1.000000
4.565900	217.840000	47.757000	526050.000000	1.886700	324990000000000000.000000	13151.000000	2.743600	1.000000

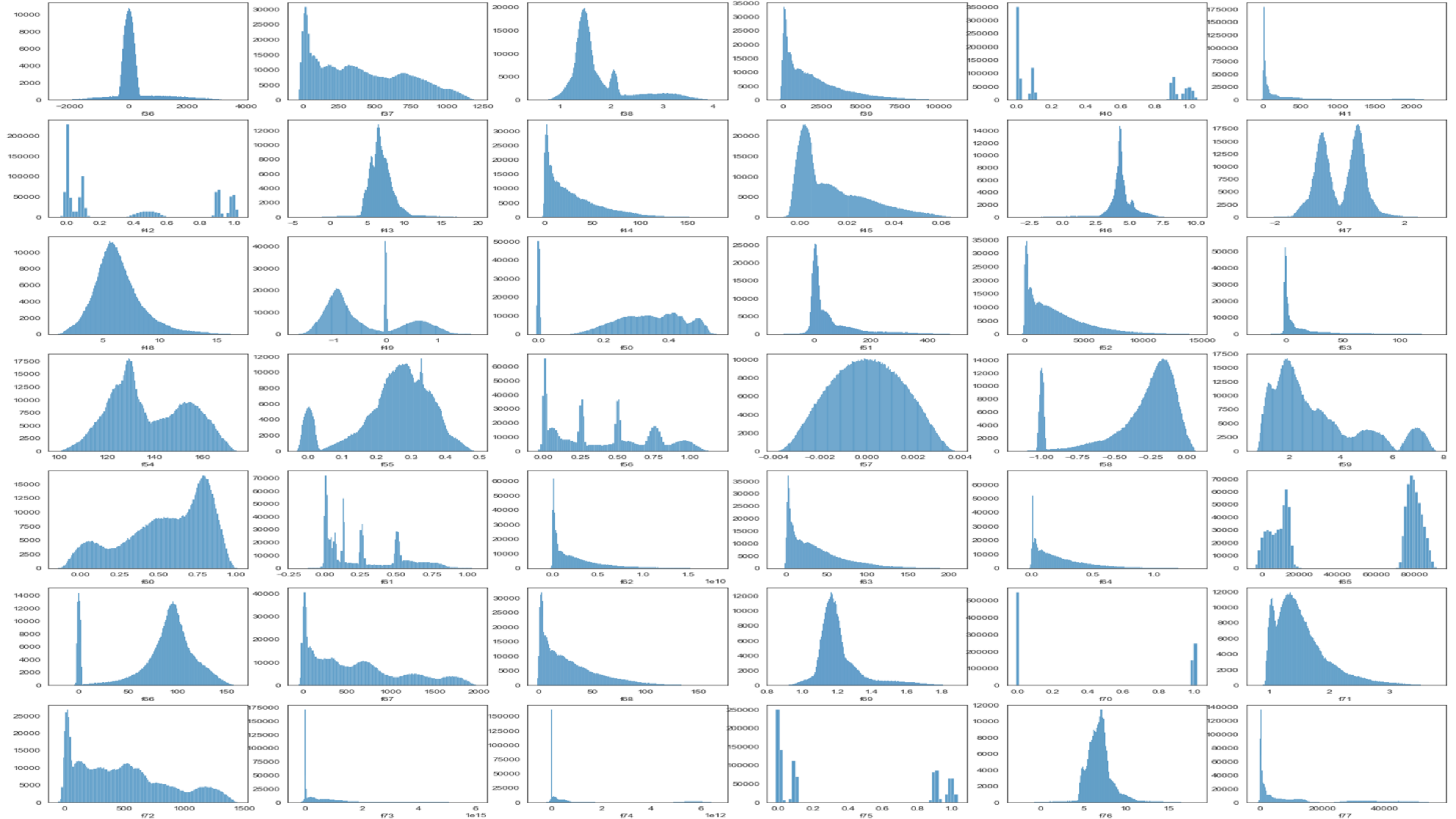
5. Box Plot


id
f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12
f13
f14
f15
f16
f17
f18
f19
f20
f21
f22
f23
f24
f25
f26
f27
f28
f29
f30
f31
f32
f33
f34
f35
f36
f37
f38
f39
f40
f41
f42
f43
f44
f45
f46
f47
f48
f49
f50
f51
f52
f53
f54
f55
f56
f57
f58
f59
f60
f61
f62
f63
f64
f65
f66
f67
f68
f69
f70
f71
f72
f73
f74
f75
f76
f77
f78
f79
f80
f81
f82
f83
f84
f85
f86
f87
f88
f89
f90
f91
f92
f93
f94
f95
f96
f97
f98
f99
f100
f101
f102
f103
f104
f105
f106
f107
f108
f109
f110
f111
f112
f113
f114
f115
f116
f117
f118
claim
0.0
0.2
0.4
0.6
0.8
1.0

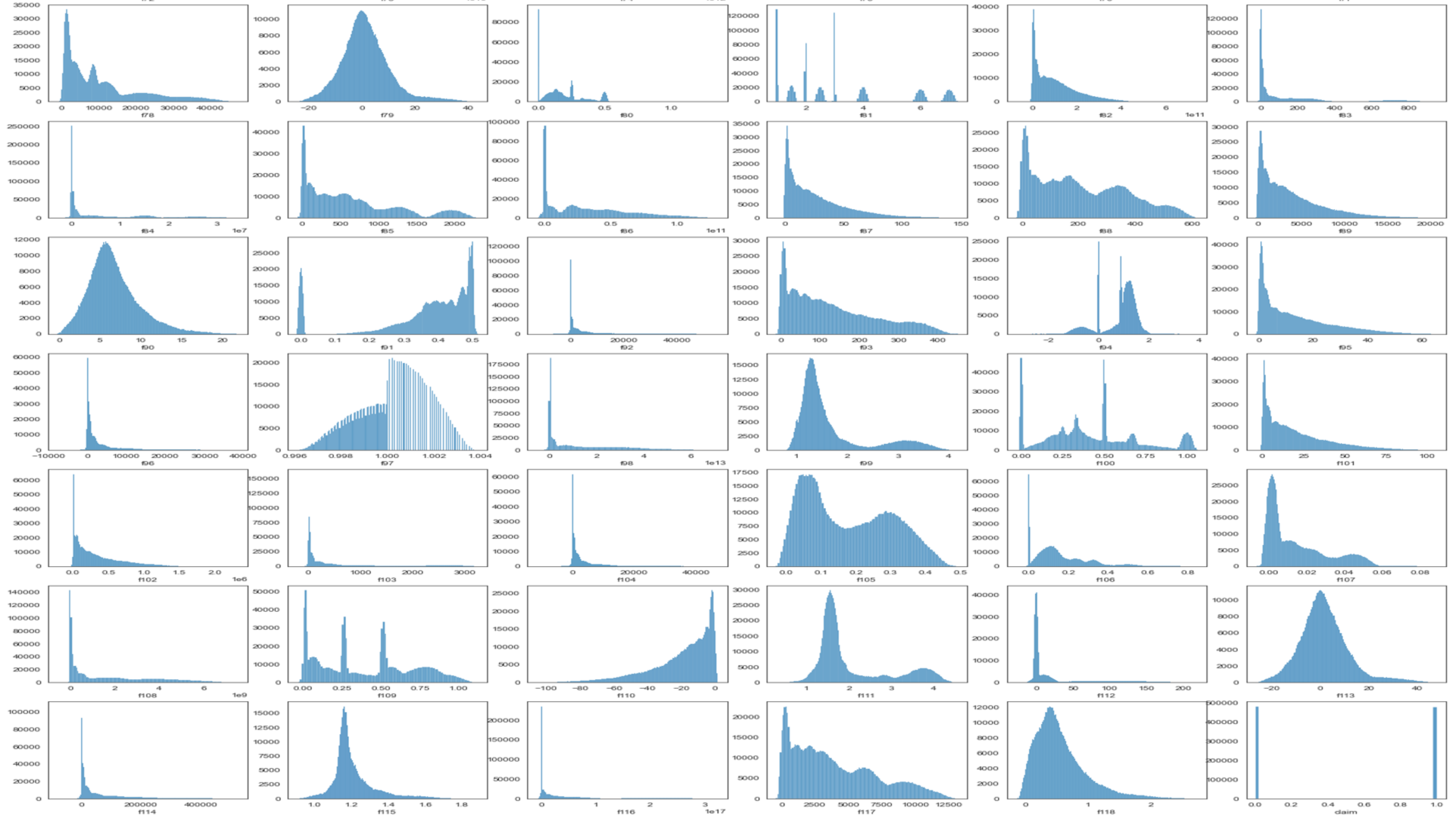
6. 分布圖



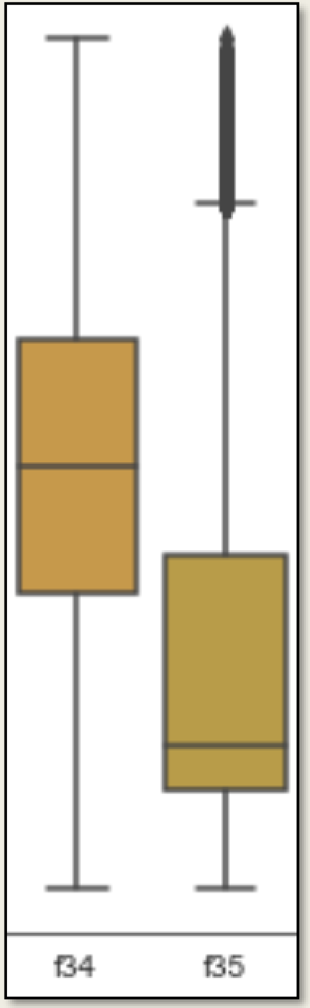
id
f1
f2
f3
f4
f5
f6
f7
f8
f9
f10
f11
f12
f13
f14
f15
f16
f17
f18
f19
f20
f21
f22
f23
f24
f25
f26
f27
f28
f29

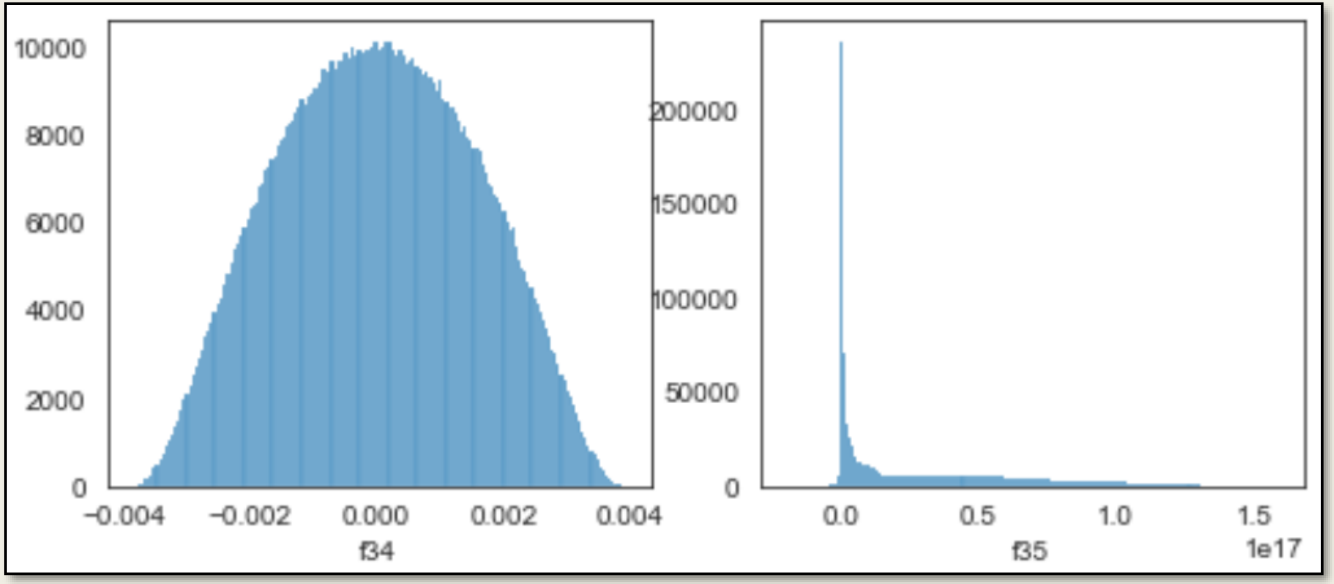


f36
f37
f38
f39
f40
f41
f42
f43
f44
f45
f46
f47
f48
f49
f50
f51
f52
f53
f54
f55
f56
f57
f58
f59
f60
f61
f62
1e10
f63
f64
f65
f66
f67
f68
f69
f70
f71
f72
f73
1e15
f74
1e12
f75
f76
f77

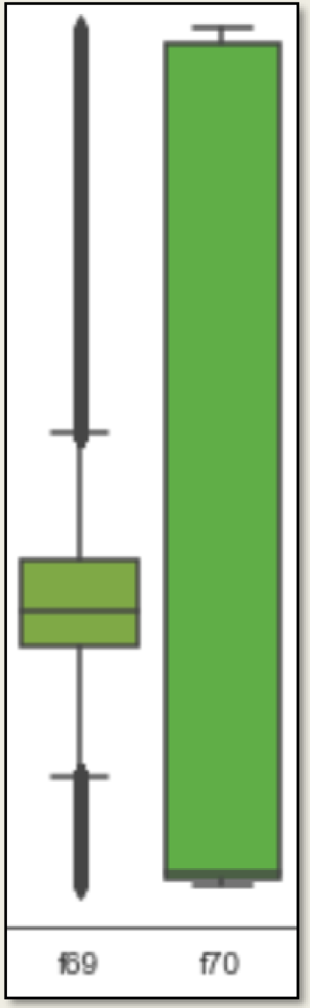

f78
f79
f80
f81
f82
f83
f84
f85
f86
f87
f88
f89
f90
f91
f92
f93
f94
f95
f96
f97
f98
f99
f100
f101
f102
f103
f104
f105
f106
f107
f108
f109
f110
f111
f112
f113
f114
f115
f116
f117
f118
claim

f34 V.S. f35

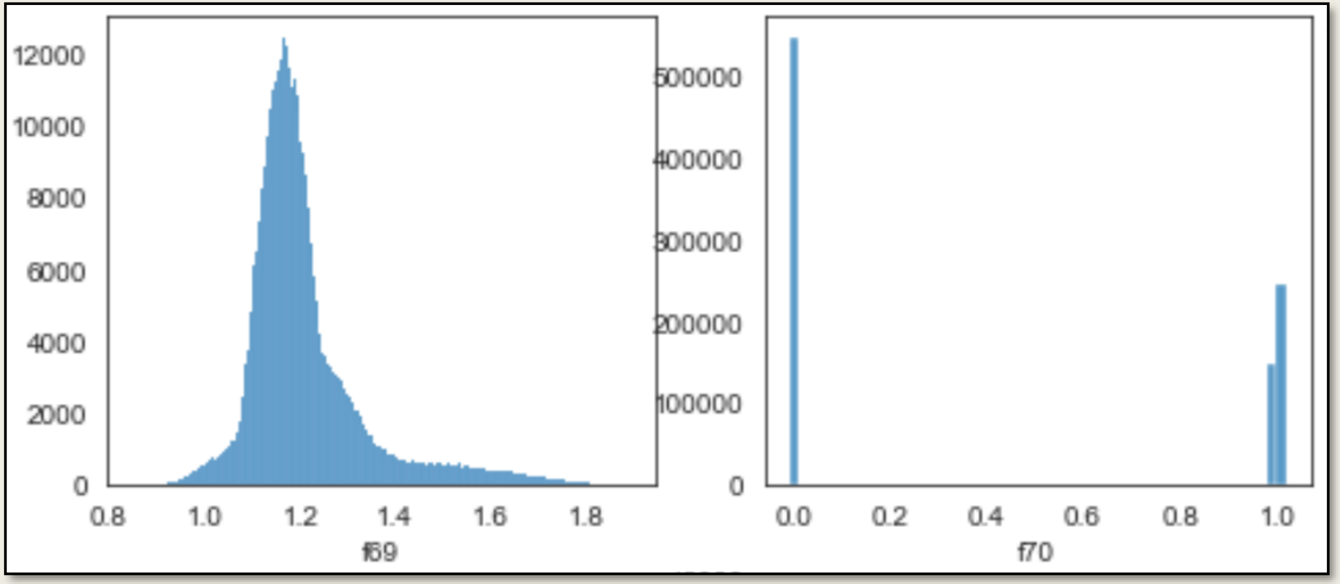

f34
f35


10000
8000
6000
4000
2000
0
−0.004
−0.002
0.000
0.002
0.004
f34
200000
150000
100000
50000
0
0.0
0.5
1.0
1.5
1e17
f35

f69 V.S. f70



f69
f70



12000
10000
8000
6000
4000
2000
0
0.8
1.0
1.2
1.4
1.6
1.8
f69
500000
400000
300000
200000
100000
0
0.0
0.2
0.4
0.6
0.8
1.0
f70

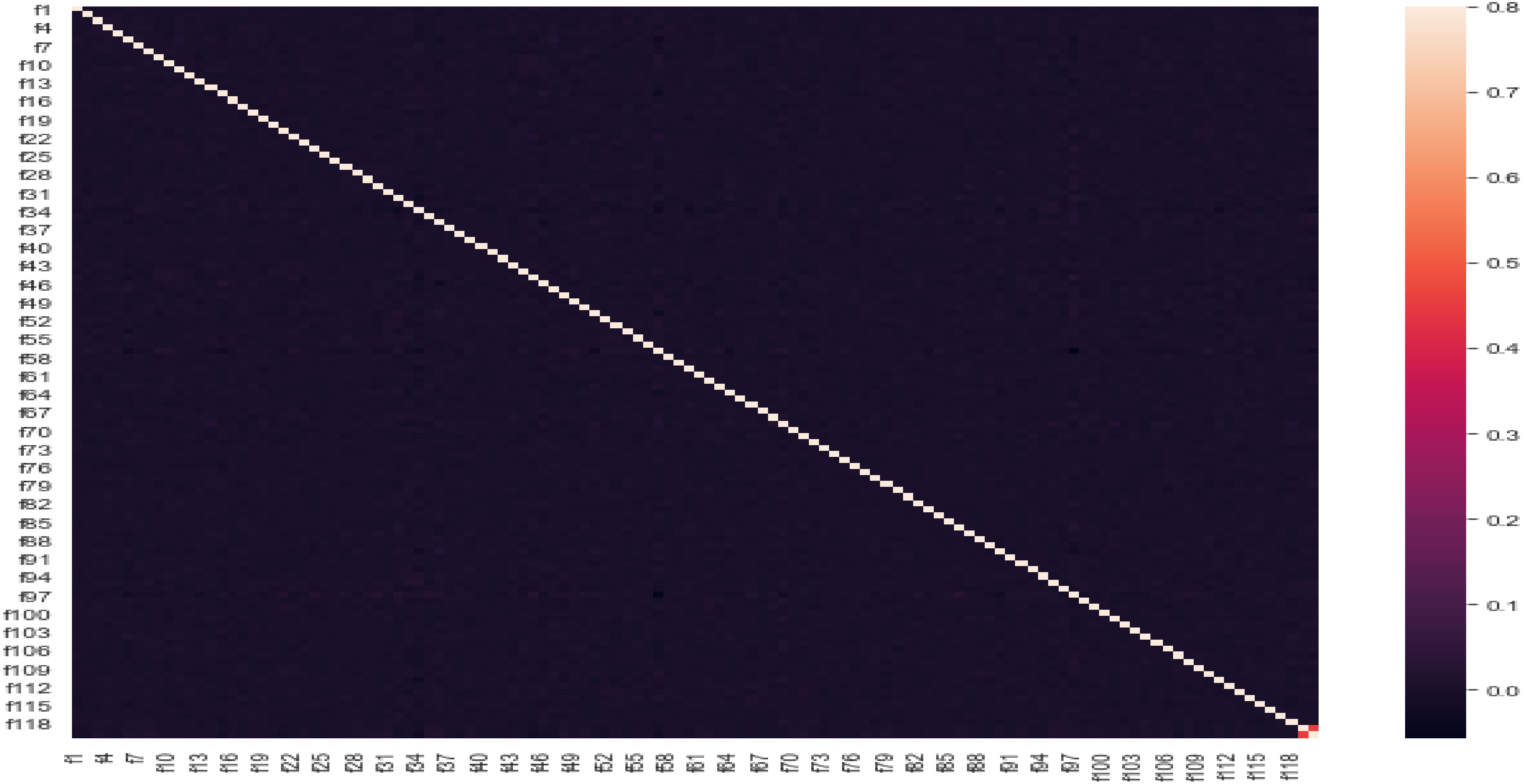
7. 考慮新特徵

```
id
0         1
1         0
2         5
3         2
4         8
         ..
957914    0
957915    4
957916    0
957917    1
957918    4
Length: 957919, dtype: int64
```

```
df.isna().sum(axis=1).head(30)
```

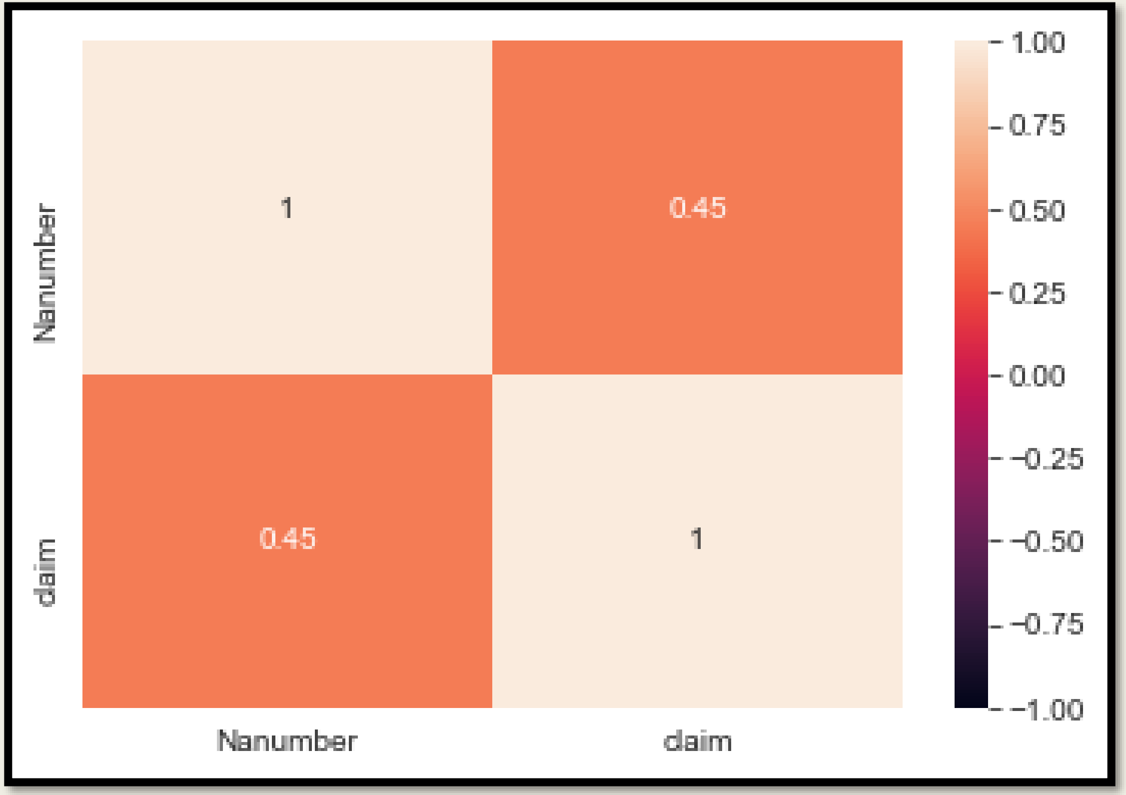
```
id
0     1
1     0
2     5
3     2
4     8
5     1
6     3
7     1
8     0
9     0
10    8
11    0
12    3
13    2
14    0
15    2
16    5
17    0
18    4
19    6
20    4
21    1
22    0
23    0
24    0
25    0
26    0
27    3
28    4
29    0
dtype: int64
```

8. 相關性



f1
f4
f7
f10
f13
f16
f19
f22
f25
f28
f31
f34
f37
f40
f43
f46
f49
f52
f55
f58
f61
f64
f67
f70
f73
f76
f79
f82
f85
f88
f91
f94
f97
f100
f103
f106
f109
f112
f115
f118
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0.0

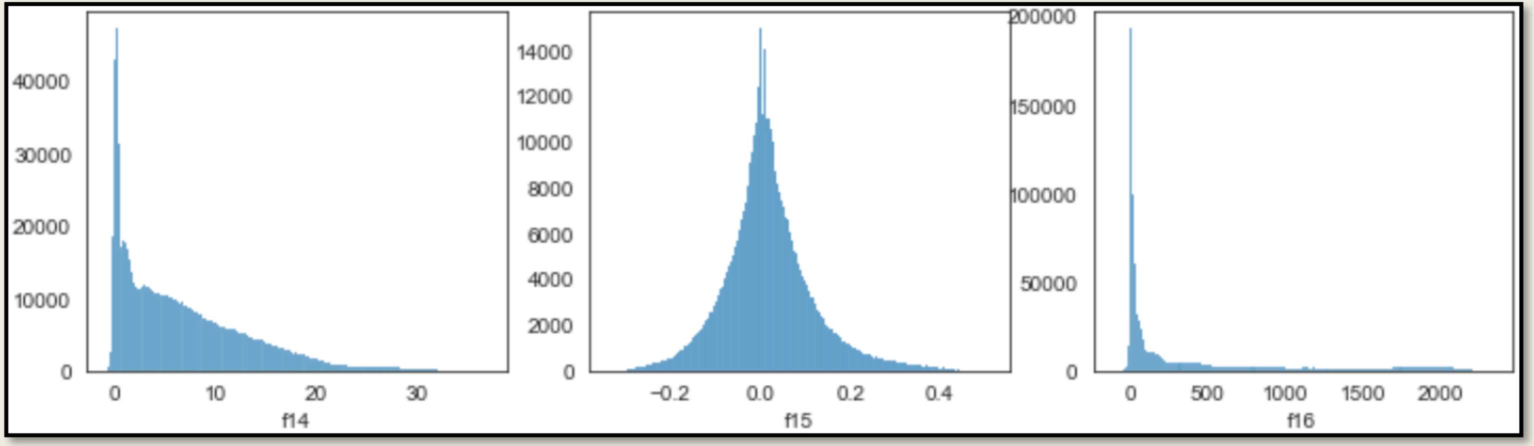
- Claim V.S. 缺失值數量



缺失值處理

這裡給出了5種可能

- 不做處理 / 補 0
- 補中位數
- 手動補值 → 看分布圖決定
- 不做處理 + 缺失值數量
- 補值 + 缺失值數量



ROC & AUC

ROC曲線
(receiver operating characteristic curve)

- ROC一般使用二元分類模型，也就是結果只會有:(有/無)、(陰性/陽性)等等。
- 以高血壓為例，收縮壓超過140或舒張壓超過90我們就判定為高血壓患者。
- 此收縮壓140和舒張壓90則稱為閾值。

混淆矩陣(Confusion Matrix)

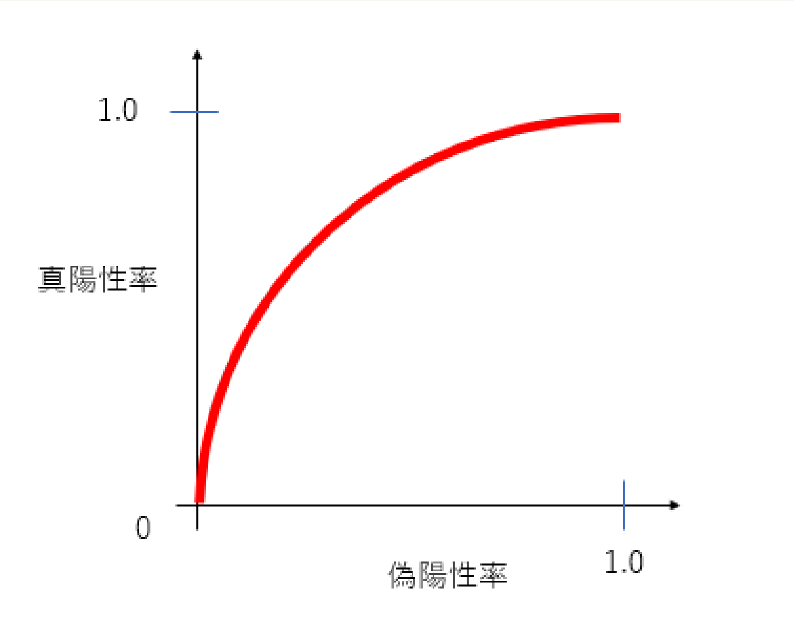
		真實值		總數
		Ture	False	
決策	接受	真陽性(TP)	假陽性(FP)	TP+FP
	拒絕	假陰性(FN)	真陰性(TN)	FN+TN
總數		TP+FN	FP+TN	

TPR= $\frac{TP}{TP+FN}$ (判斷正確)

FPR= $\frac{FP}{FP+TN}$ (判斷錯誤)

ACC= $\frac{TP+TN}{TP+FN+FP+TN}$ (準確度)

ROC曲線



Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

真實狀況　　真實狀況

$$\text{TPR}=\frac{TP}{TP+FN}$$

$$\text{FPR}=\frac{FP}{FP+TN}$$

$$\text{ACC}=\frac{TP+TN}{TP+FP+TN+FN}$$

Inst#	Class	Score	Inst#	Class	Score
1	**p**	.9	11	**p**	.4
2	**p**	.8	12	**n**	.39
3	**n**	.7	13	**p**	.38
4	**p**	.6	14	**n**	.37
5	**p**	.55	15	**n**	.36
6	**p**	.54	16	**n**	.35
7	**n**	.53	17	**p**	.34
8	**n**	.52	18	**n**	.33
9	**p**	.51	19	**p**	.30
10	**n**	.505	20	**n**	.1

閾值以0.9算

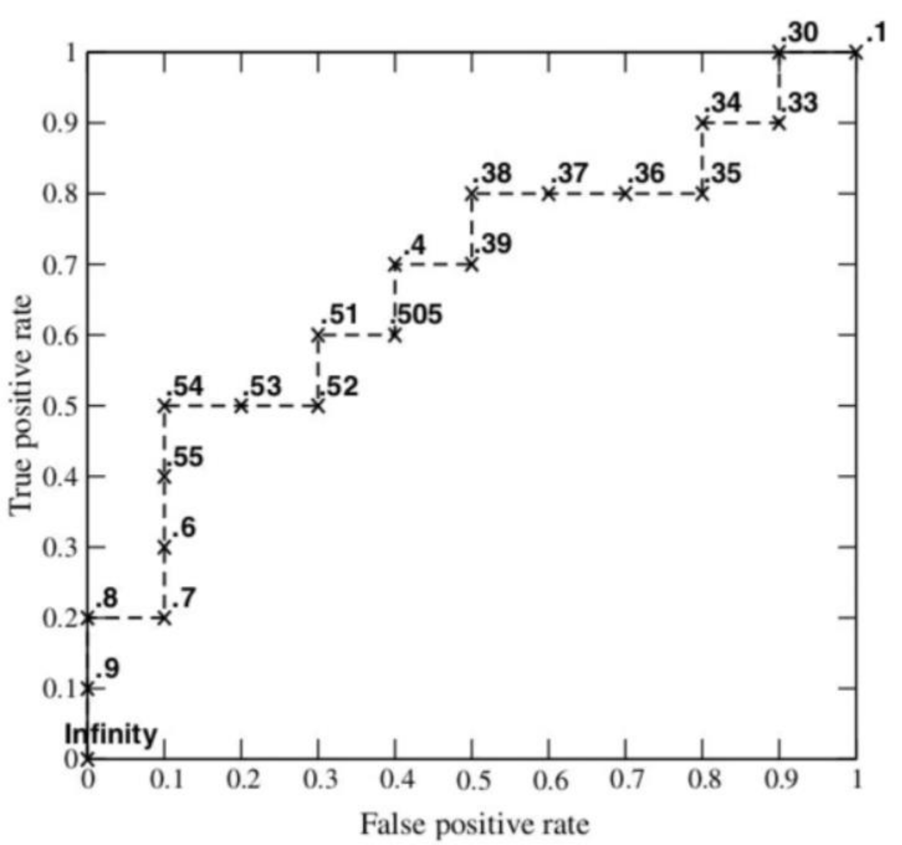
$$\text{TPR}=\frac{TP}{TP+FN}=\frac{1}{1+9}=\frac{1}{10}$$

$$\text{FPR}=\frac{FP}{FP+TN}=\frac{0}{0+10}=0$$

閾值以0.1算

$$\text{TPR}=\frac{TP}{TP+FN}=\frac{10}{10+0}=1$$

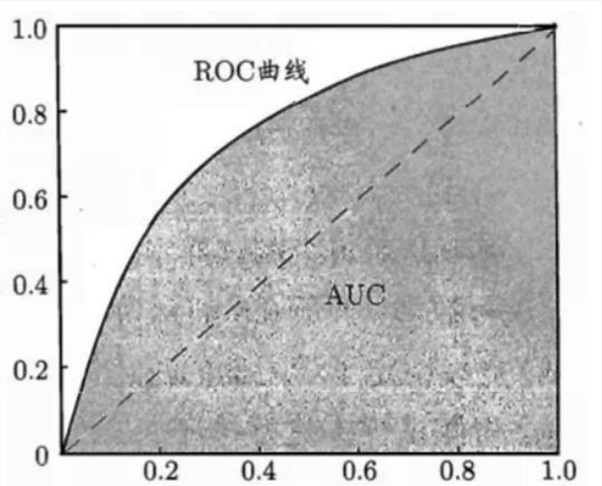
$$\text{FPR}=\frac{FP}{FP+TN}=\frac{10}{10+0}=1$$



True positive rate
False positive rate
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
1
Infinity
.9
.8
.7
.6
.55
.54
.53
.52
.51
.505
.4
.39
.38
.37
.36
.35
.34
.33
.30
.1

AUC
(Area under the Curve of ROC)

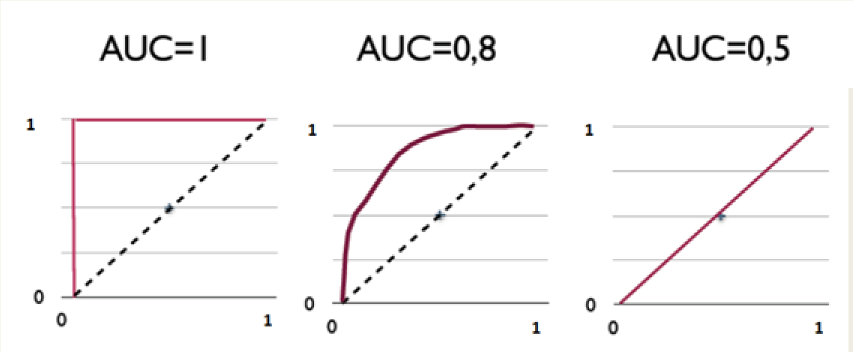
- ROC曲線下的面積稱為AUC。
- 可更有效的分辨兩條有相交的ROC曲線哪條分類的更好。
- AUC 遠離 0.5 好的模型
- AUC 靠近 0.5 差的模型



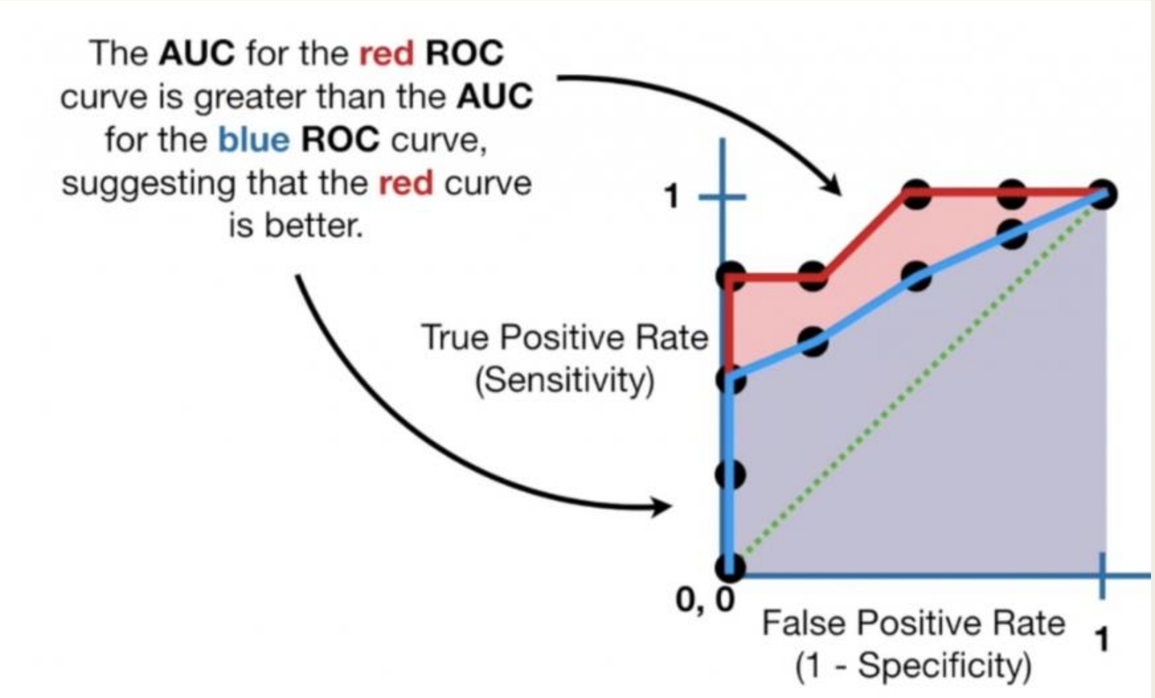
如何計算AUC

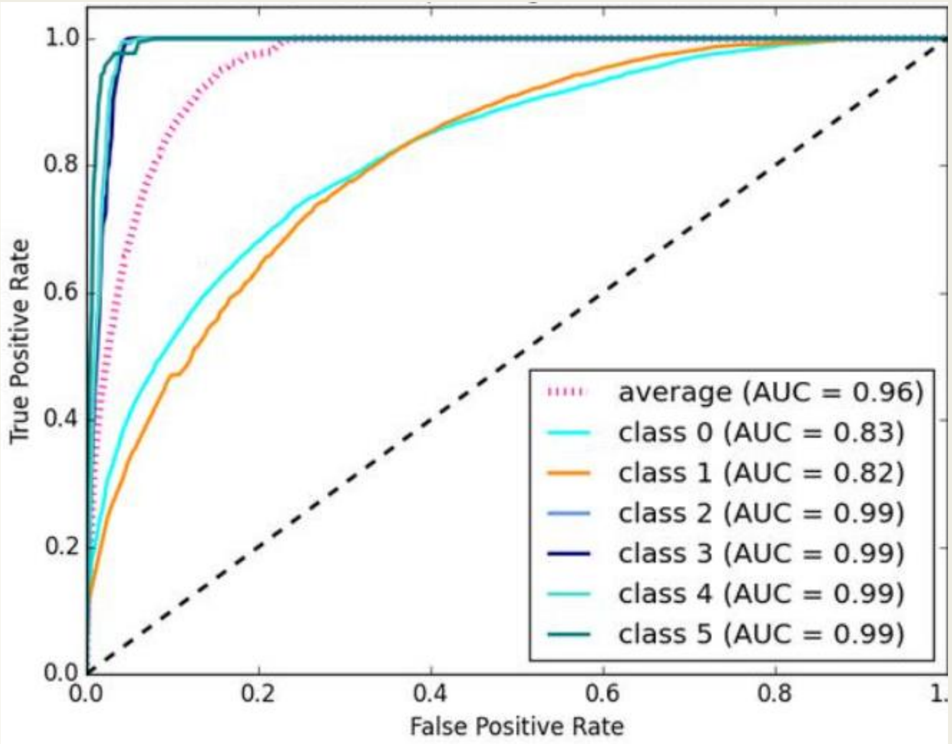
- AUC面積介於 0~1 之間

- 簡單地將每個相鄰的點以直線連接，計算連線下方的總面積。因為每一線段下方都是一個梯形，所以叫梯形法。



AUC如何分辨兩條ROC的好壞




True Positive Rate
False Positive Rate
1.0
0.8
0.6
0.4
0.2
0.0
0.0
0.2
0.4
0.6
0.8
1.
average (AUC = 0.96)
class 0 (AUC = 0.83)
class 1 (AUC = 0.82)
class 2 (AUC = 0.99)
class 3 (AUC = 0.99)
class 4 (AUC = 0.99)
class 5 (AUC = 0.99)

5種模型

Logistic & CART & Random Forest & XGBoost & Light GBM

常態 & 標準化處理

- Logistic、 Decision Tree、 Random Forest 都沒有常態假設
- β_i 大小直接影響預測機率 → 標準化 →各特徵貢獻相同
- $Logistic(f(x)) = \frac{1}{1+e^{-f(x)}}$, $f(x) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \ldots$
- Decision Tree : 每個特徵被單獨處理 (Entropy & Gini impurity)
- Random Forest : 由多棵決策樹組成

Logistic

預測二元類別目標變數的發生機率

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.1s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.1s finished
```

```
Seed-42 | Fold-0 | OOF Score: 0.7976887705220119
```

```
random_state=42
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.2s finished
```

```
Seed-42 | Fold-1 | OOF Score: 0.7997294098730139
```

```
score = roc_auc_score(val_y, y_pred)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.2s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.2s finished
```

```
Seed-42 | Fold-2 | OOF Score: 0.7984193136313238
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    2.0s finished
```

```
Seed-42 | Fold-3 | OOF Score: 0.7998322419186739
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.0s remaining:    0.0s
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.0s finished
2it [01:49, 54.86s/it]
```

```
Seed-42 | Fold-4 | OOF Score: 0.7997507316245783
```

```
Seed: 42 | Aggregate OOF Score: 0.7990840935139204
```

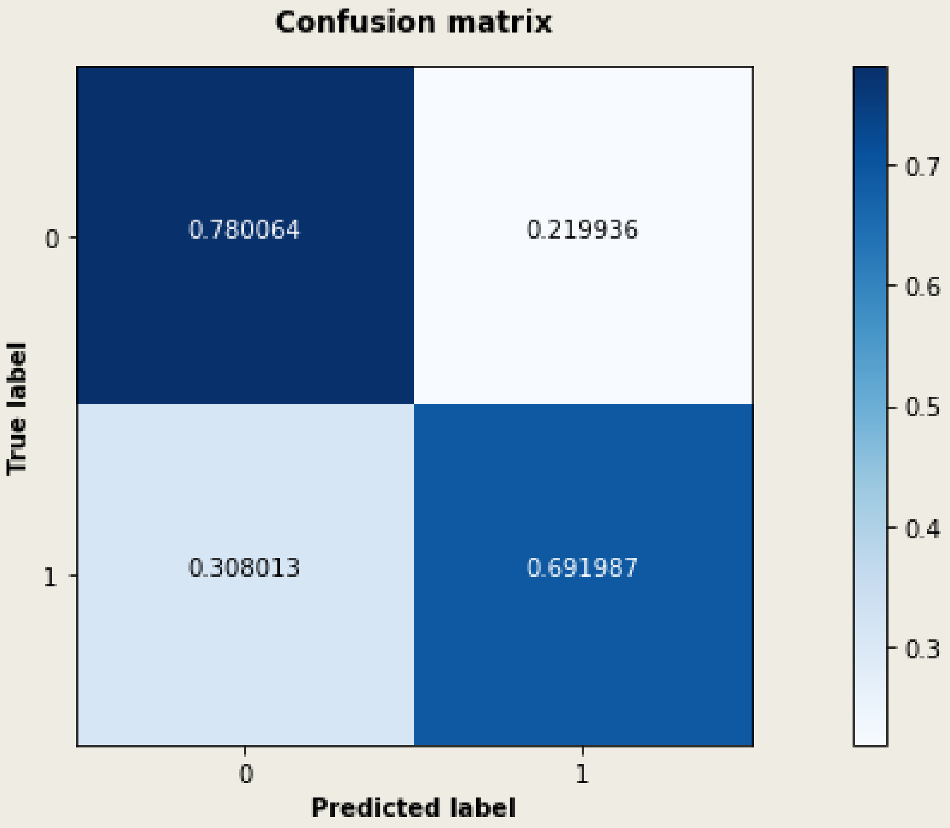
Logistic

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:

 Seed-42 | Fold-0 | OOF Score: 0.7976887705220119

[Parallel(n_jobs=1)]: Using backend SequentialBackend
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:

 Seed-42 | Fold-1 | OOF Score: 0.7997294098730139

[Parallel(n_jobs=1)]: Using backend SequentialBackend
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:

 Seed-42 | Fold-2 | OOF Score: 0.7984193136313238

[Parallel(n_jobs=1)]: Using backend SequentialBackend
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:

 Seed-42 | Fold-3 | OOF Score: 0.7998322419186739

[Parallel(n_jobs=1)]: Using backend SequentialBackend
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:
2it [01:49, 54.86s/it]

 Seed-42 | Fold-4 | OOF Score: 0.7997507316245783

 Seed: 42 | Aggregate OOF Score: 0.7990840935139204
```

Rf5

```
0it [00:00, ?it/s]

 Seed-24 | Fold-0 | OOF Score: 0.7904225431168876

 Seed-24 | Fold-1 | OOF Score: 0.789410790479903

 Seed-24 | Fold-2 | OOF Score: 0.7935818125406779

 Seed-24 | Fold-3 | OOF Score: 0.790900926455021

1it [40:01, 2401.80s/it]

 Seed-24 | Fold-4 | OOF Score: 0.789661078818029

 Seed: 24 | Aggregate OOF Score: 0.7907954302821038


 Seed-42 | Fold-0 | OOF Score: 0.7915421563957606

 Seed-42 | Fold-1 | OOF Score: 0.791165291138846

 Seed-42 | Fold-2 | OOF Score: 0.7913487836915104

 Seed-42 | Fold-3 | OOF Score: 0.7925757170893544

2it [1:15:33, 2266.88s/it]

 Seed-42 | Fold-4 | OOF Score: 0.7920922576317434

 Seed: 42 | Aggregate OOF Score: 0.791744841189443

Aggregate OOF Score: 0.7912701357357734
```

Logistic



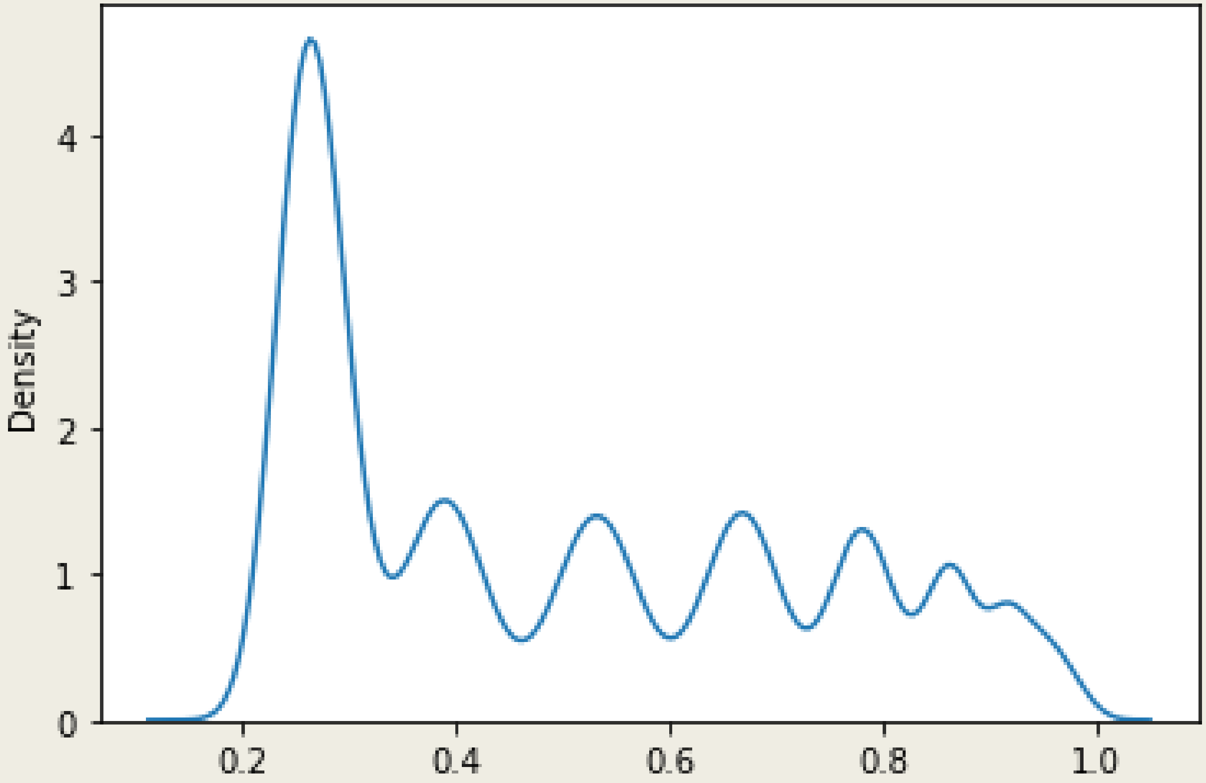

```
print((cnf_matrix[0,0] + cnf_matrix[1,1]) / sum(cnf_matrix).sum())
```

```
0.7378786724138471
```

Accuracy(準確度) : (TP+TN) / 總資料

Confusion matrix	
TN（真陰性）	FP（假陽性）
FN（假陰性）	TP（真陽性）

KDE Plot

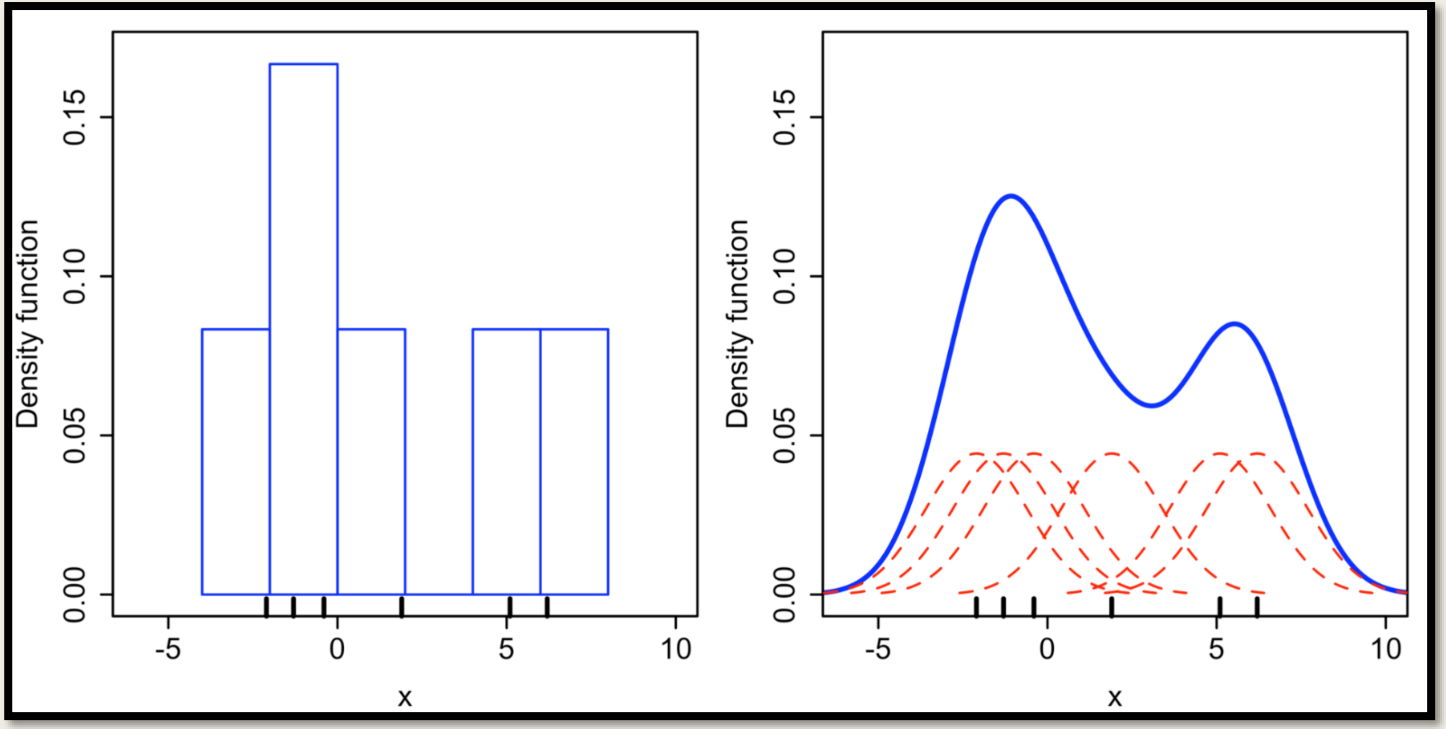


id	logistic
957919	0.395188
957920	0.207112
957921	0.375288
957922	0.263373
957923	0.285499
...	...
1451388	0.581088
1451389	0.216424
1451390	0.844771
1451391	0.223254
1451392	0.908159

Logistic

核密度估計(Kernel Density Estimation)

估計未知的機率密度函數(核函數 , ex:高斯)



標準化 V.S. 不標準化

```
# 直接跑不用 kfold

Xtrain = Xtrain.fillna(0)                          # 補 0
Xtest = Xtest.fillna(0)                            # Logistic 不能有 NaN

sc=StandardScaler()

sc.fit(Xtrain)

x_train_nor=sc.transform(Xtrain)
x_test_nor=sc.transform(Xtest)

data_train, data_test, target_train, target_test = train_test_split(x_train_nor,
                                                                    Ytrain,
                                                                    test_size=0.2,
                                                                    random_state=42)

log_model = LogisticRegression(max_iter=10000,verbose=True)

log_model.fit(data_train,target_train)

# 機率分類判斷
predictions = log_model.predict_proba(data_test)  # 分別給出各類預測機率

score = roc_auc_score(target_test, predictions[:,-1])          # (957919,)
print(score)
```

```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
0.7976887705220119
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    1.0s finished
```

```
# 不標準化
# 直接跑不用 kfold

Xtrain = Xtrain.fillna(0)                                  # 補 0
Xtest = Xtest.fillna(0)                                    # Logistic 不能有 NaN

data_train, data_test, target_train, target_test = train_test_split(Xtrain,
                                                                    Ytrain,
                                                                    test_size=0.2,
                                                                    random_state=42)

log_model = LogisticRegression(max_iter=10000,verbose=True,penalty='l2')

log_model.fit(data_train,target_train)                     # 放訓練!!

# 印出係數
# print(log_model.coef_ ,'\n')

# 印出截距
# print(log_model.intercept_ ,'\n' )

# 機率分類判斷
predictions = log_model.predict_proba(data_test)  # 分別給出各類預測機率

score = roc_auc_score(target_test, predictions[:,-1])                # (957919,)
print(score)
```

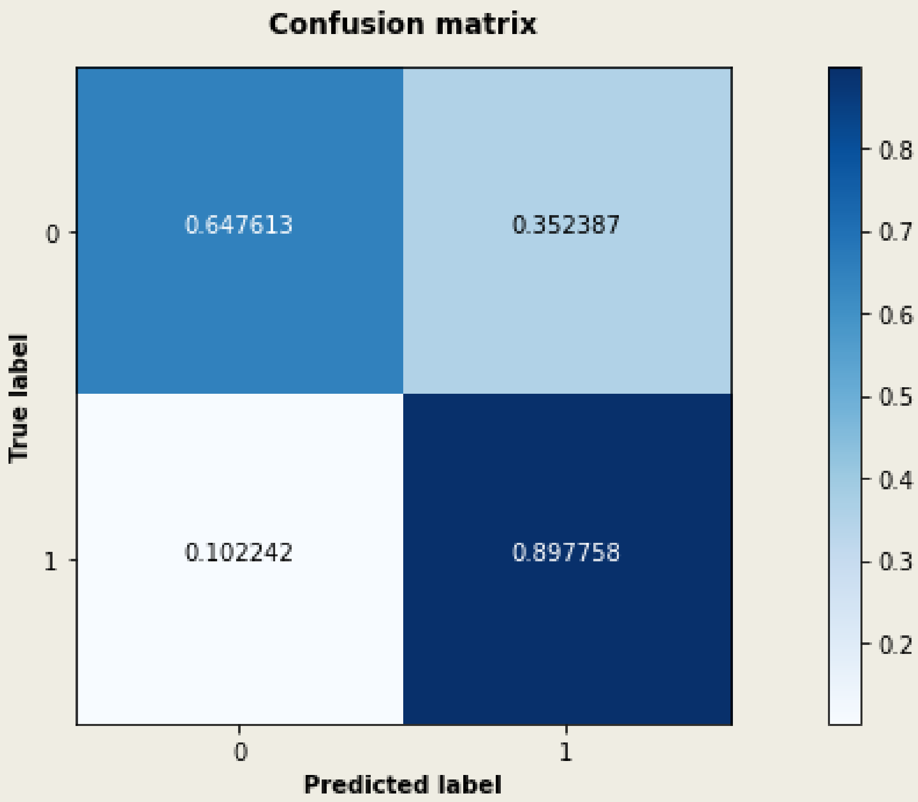
```
[Parallel(n_jobs=1)]: Using backend SequentialBackend with 1 concurrent workers.
[Parallel(n_jobs=1)]: Done   1 out of   1 | elapsed:    5.2s finished
0.507425029718713
```

Decision Tree

尋找特徵進行決策，試著讓同一個類別混亂程度越小越好

Cross validation : 5 – fold & Max depth = 5

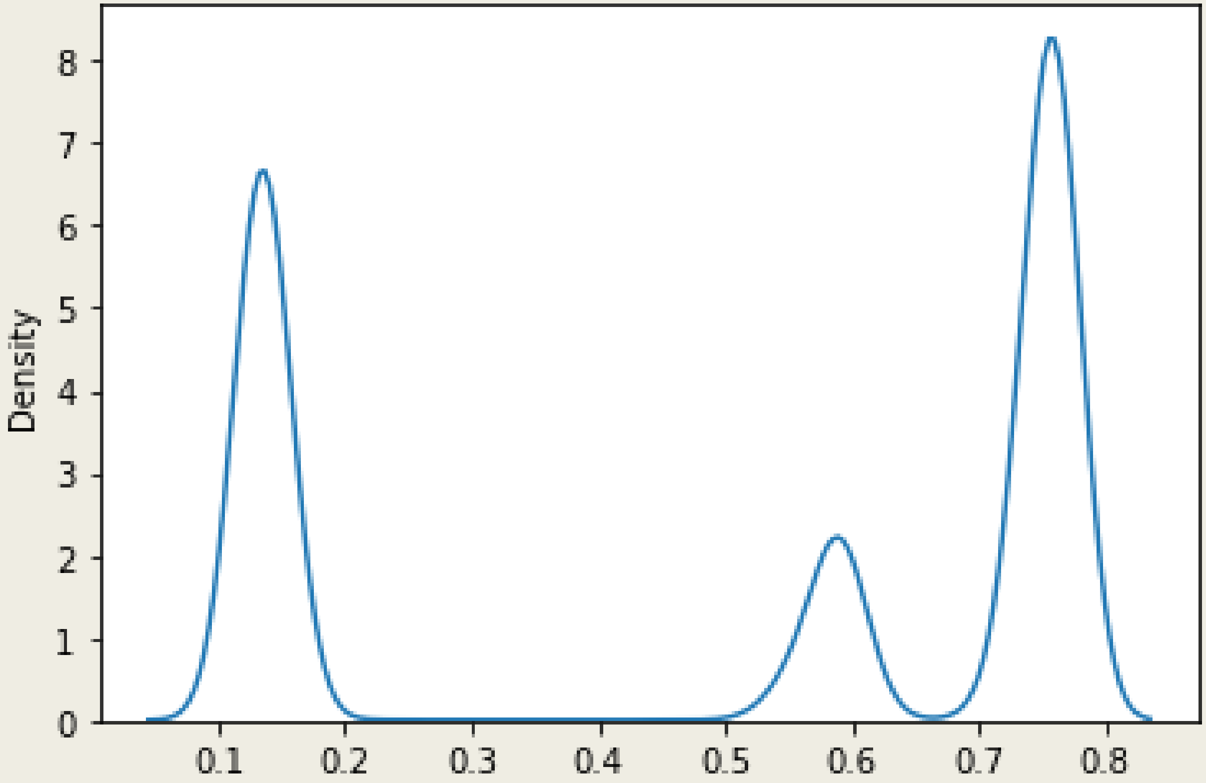
```
Seed-42 | Fold-0 | OOF Score: 0.7986493842043213

Seed-42 | Fold-1 | OOF Score: 0.800868873022804

Seed-42 | Fold-2 | OOF Score: 0.7992917527832667

Seed-42 | Fold-3 | OOF Score: 0.8009613011039297

2it [17:22, 521.46s/it]

Seed-42 | Fold-4 | OOF Score: 0.8003615492041063

Seed: 42  Aggregate OOF Score: 0.8000265720636855
```



Accuracy(準確度)

```
print((cnf_matrix[0,0] + cnf_matrix[1,1]) / sum(cnf_matrix).sum())

0.7726857222236054
```

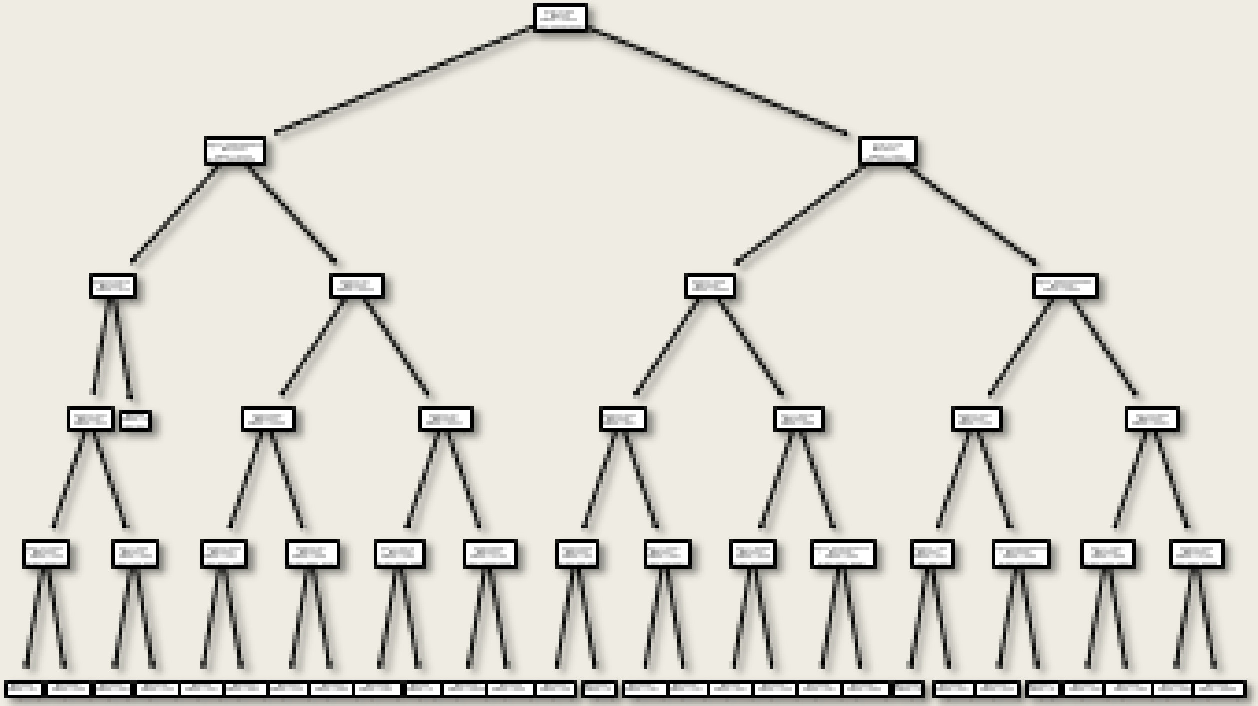
KDE Plot



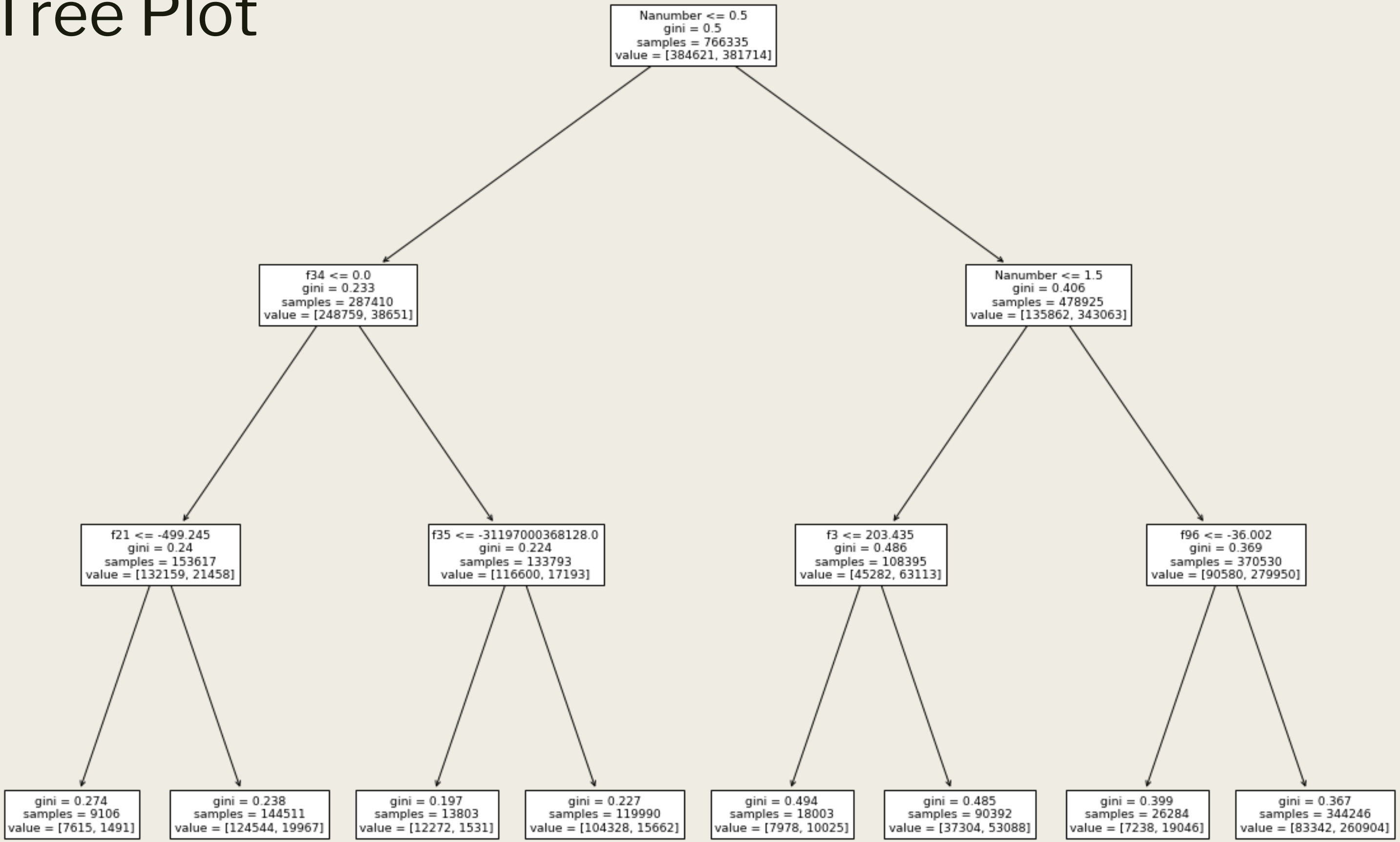
decesion_tree
0.583207
0.129017
0.595605
0.144138
0.151109
...
0.762203
0.129017
0.757146
0.123806
0.752278

Decision Tree

Tree Plot

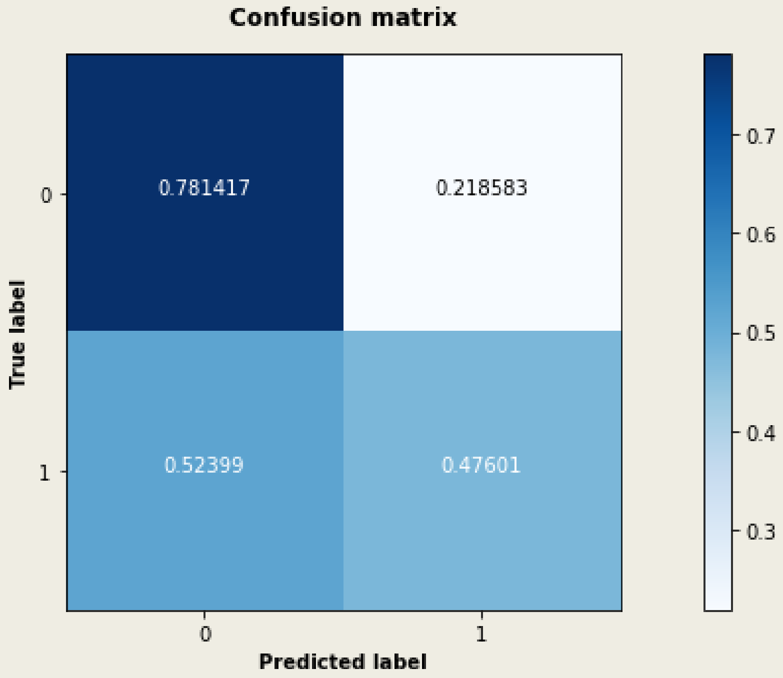


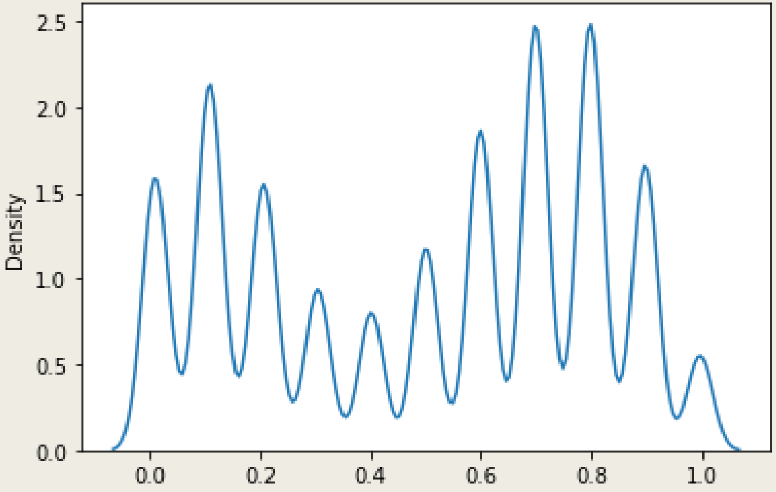
Tree Plot


Nanumber <= 0.5
gini = 0.5
samples = 766335
value = [384621, 381714]
f34 <= 0.0
gini = 0.233
samples = 287410
value = [248759, 38651]
Nanumber <= 1.5
gini = 0.406
samples = 478925
value = [135862, 343063]
f21 <= -499.245
gini = 0.24
samples = 153617
value = [132159, 21458]
f35 <= -31197000368128.0
gini = 0.224
samples = 133793
value = [116600, 17193]
f3 <= 203.435
gini = 0.486
samples = 108395
value = [45282, 63113]
f96 <= -36.002
gini = 0.369
samples = 370530
value = [90580, 279950]
gini = 0.274
samples = 9106
value = [7615, 1491]
gini = 0.238
samples = 144511
value = [124544, 19967]
gini = 0.197
samples = 13803
value = [12272, 1531]
gini = 0.227
samples = 119990
value = [104328, 15662]
gini = 0.494
samples = 18003
value = [7978, 10025]
gini = 0.485
samples = 90392
value = [37304, 53088]
gini = 0.399
samples = 26284
value = [7238, 19046]
gini = 0.367
samples = 344246
value = [83342, 260904]

Cross validation : 5 – fold & Max depth = 60

```
Seed-42 | Fold-0 | OOF Score: 0.6438749861950258

Seed-42 | Fold-1 | OOF Score: 0.6431312914987519

Seed-42 | Fold-2 | OOF Score: 0.6421279613404753

Seed-42 | Fold-3 | OOF Score: 0.6439408267716362

2it [1:29:11, 2675.92s/it]

Seed-42 | Fold-4 | OOF Score: 0.646647222553194

Seed: 42 | Aggregate OOF Score: 0.6439444576718165
```






```
print((cnf_matrix[0,0] + cnf_matrix[1,1]) / sum(cnf_matrix).sum())
```

```
0.6287137423953053
```

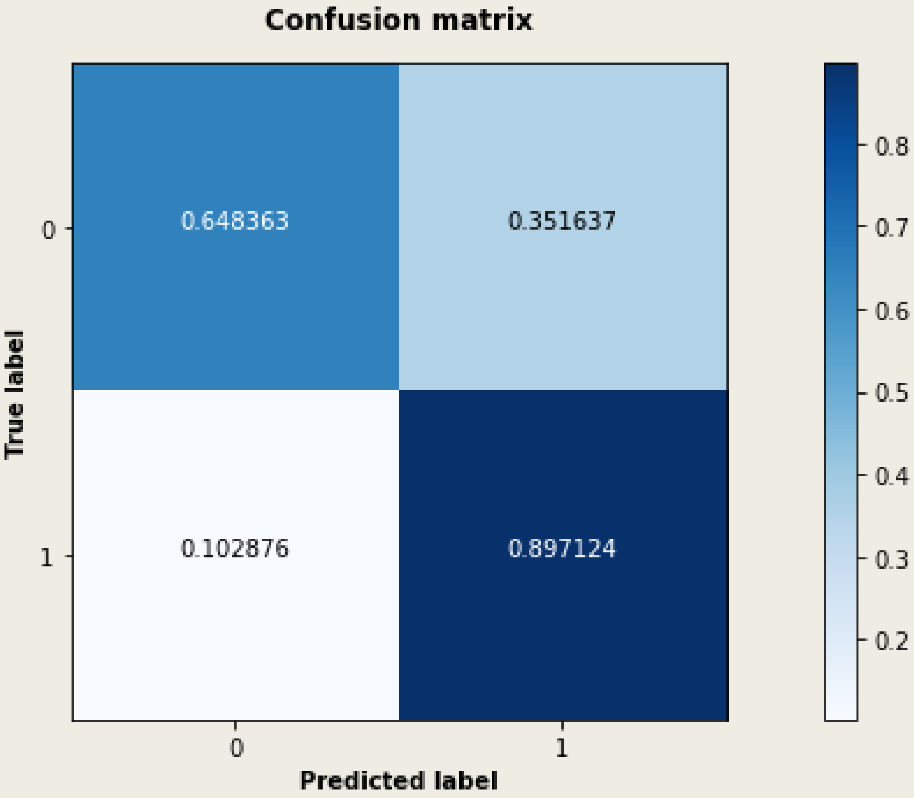
Decision Tree

Random Forest

Bagging + 隨機特徵採樣的多個決策樹

Cross validation : 5 – fold & Max depth = 60

```
Seed-42 | Fold-0 | OOF Score: 0.7986236765874141

Seed-42 | Fold-1 | OOF Score: 0.8006792334909092

Seed-42 | Fold-2 | OOF Score: 0.7982417104993047

Seed-42 | Fold-3 | OOF Score: 0.7999114689226849

2it [9:13:28, 16604.47s/it]

Seed-42 | Fold-4 | OOF Score: 0.8003058428876731

Seed: 42 | Aggregate OOF Score: 0.7995523864775972
```

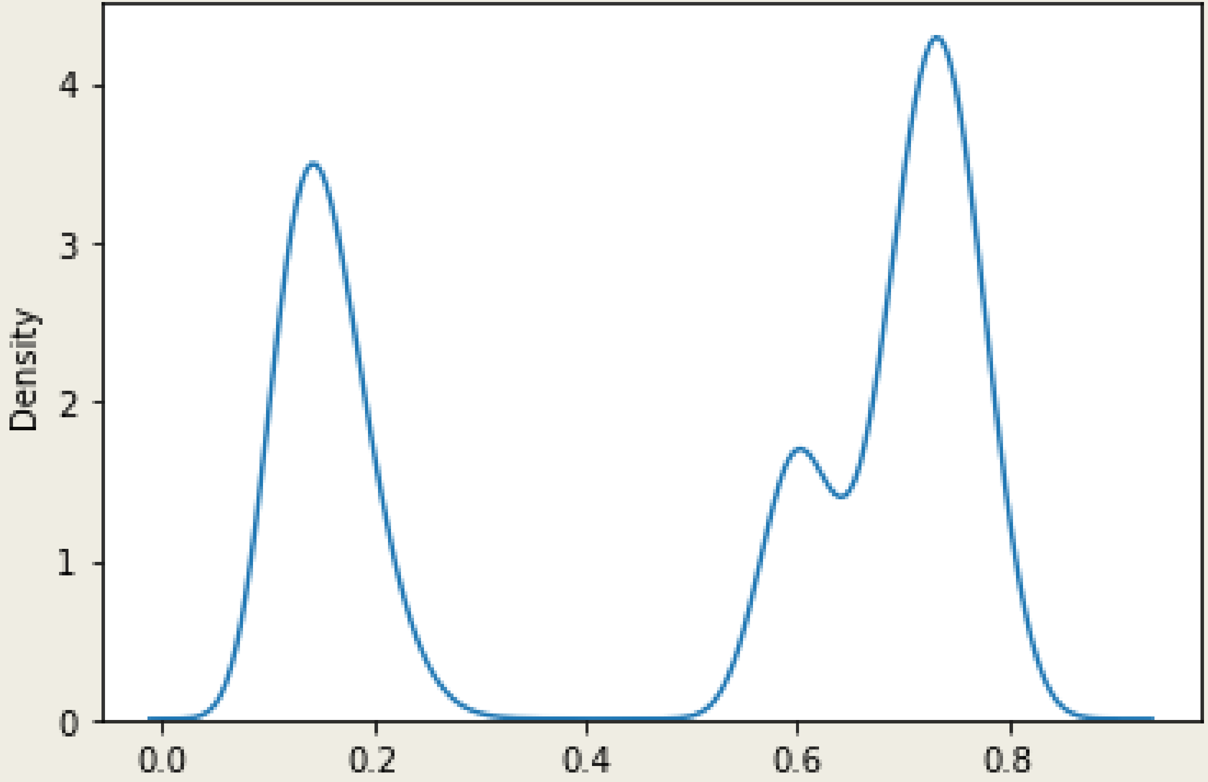


Accuracy(準確度)

```
print((cnf_matrix[0,0] + cnf_matrix[1,1]) / sum(cnf_matrix).sum())
```

```
0.772743139343641
```

KDE Plot

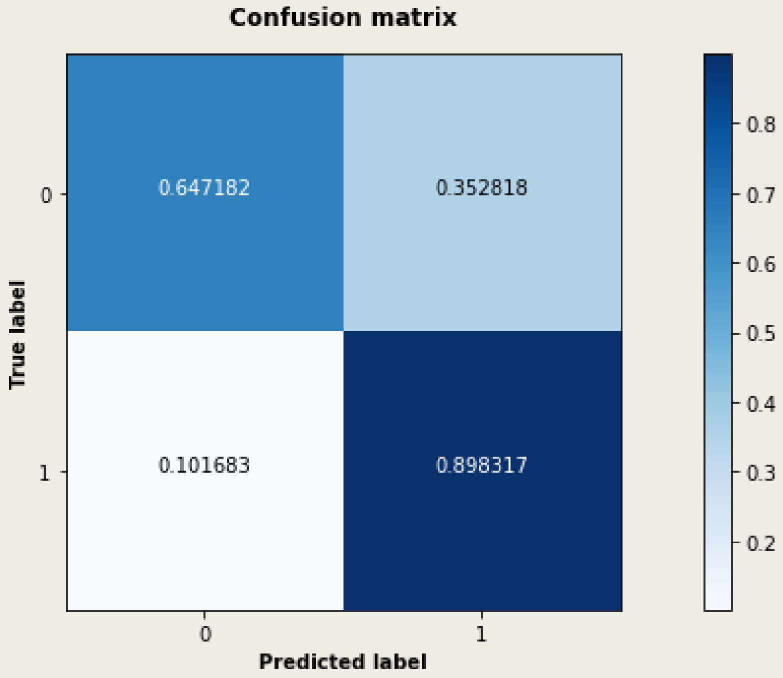


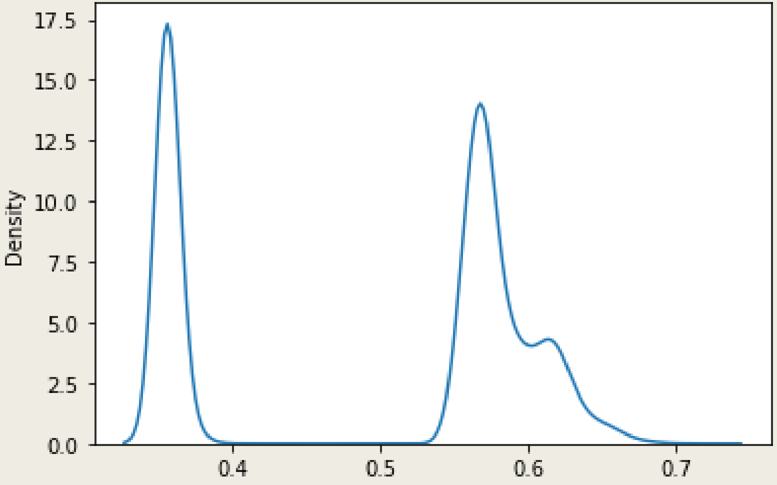
random_forest
0.653503
0.107132
0.621970
0.129830
0.170366
...
0.742159
0.151462
0.766042
0.204669
0.690662

Random Forest

Cross validation : 5 – fold & Max depth = 5

```
Seed-42 | Fold-0 | OOF Score: 0.7915421563957606

Seed-42 | Fold-1 | OOF Score: 0.791165291138846

Seed-42 | Fold-2 | OOF Score: 0.7913487836915104

Seed-42 | Fold-3 | OOF Score: 0.7925757170893544

2it [1:15:33, 2266.88s/it]

Seed-42 | Fold-4 | OOF Score: 0.7920922576317434

Seed: 42 | Aggregate OOF Score: 0.791744841189443
```






```
print((cnf_matrix[0,0] + cnf_matrix[1,1]) / sum(cnf_matrix).sum())
```

```
0.7727498509248139
```

Random Forest

以XGBOOST處理缺失特徵值

處理方式/評分	Private score	Public score
不做處理	0.80023	0.80043
補中位數	0.52158	0.52063
補平均	0.51676	0.51813
觀察各特徵分佈，人工標識	0.51809	0.51865

為什麼處理後預測結果變差了？

NA存在是合理的：

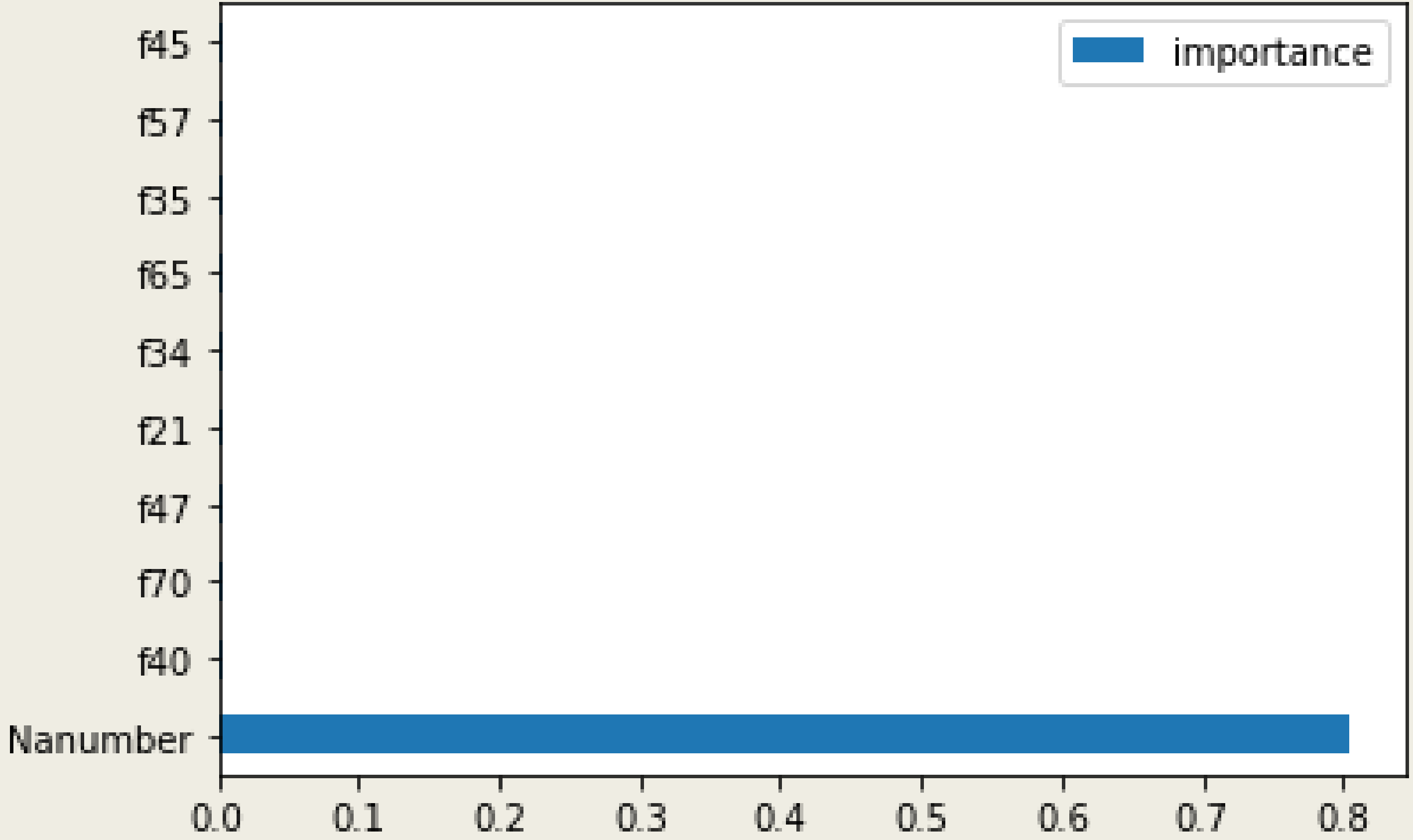
逆選擇

道德危機

提取na個數作為新的特徵

id	Nannumber
0	1
1	0
2	5
……	……
957917	1
957918	4

Private score	Public score
0.81155	0.81176


importance
f45
f57
f35
f65
f34
f21
f47
f70
f40
Nanumber
0.0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8

提取na個數作為新的特徵，並對NA值進行填補(平均)

Private score	Public score
0.64020	0.63977

參數調整

- 選擇一個相對較高的學習率，尋找符合學習率的樹個數
- 調整樹的特定參數
- 調整正則化參數，減少複雜度，防止過擬合

參數調整(XGboost)

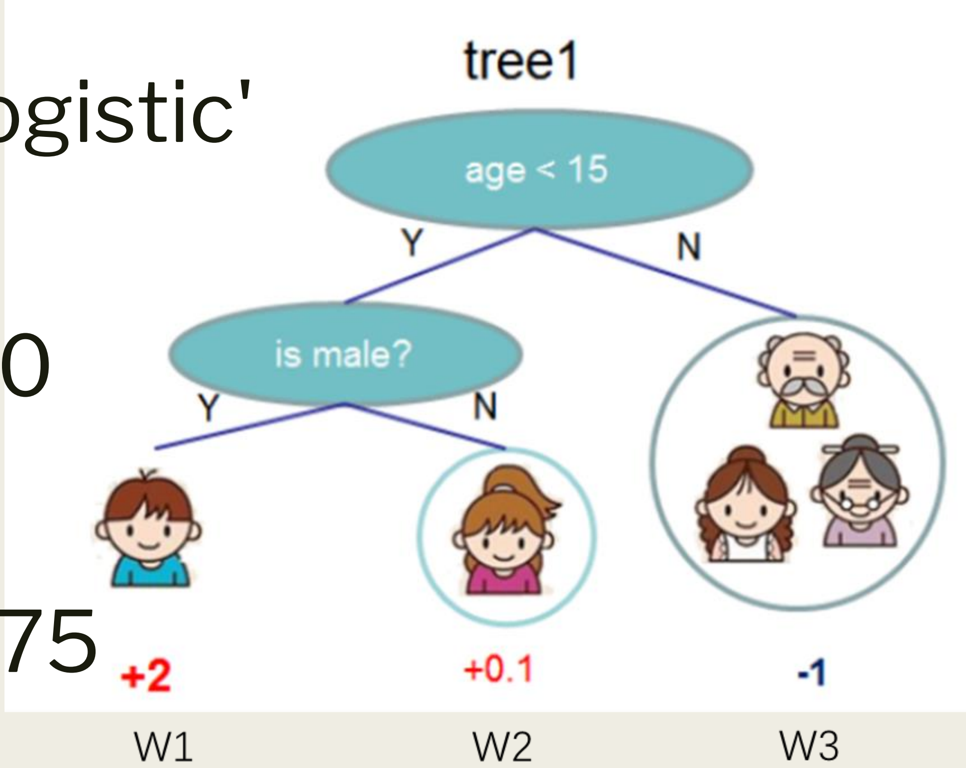
'objective': 'binary:logistic'

'learing_rate': 0.1

'n_estimators': 3000

'max_depth’: 5

'min_child_weight': 75



參數調整(XGBOOST)

'gamma': 0.1,

'subsample': 0.55

'colsample_bytree': 0.7

'reg_alpha': 10

'verbosity': 0

'random_state': 42

參數調整(Lightgbm)

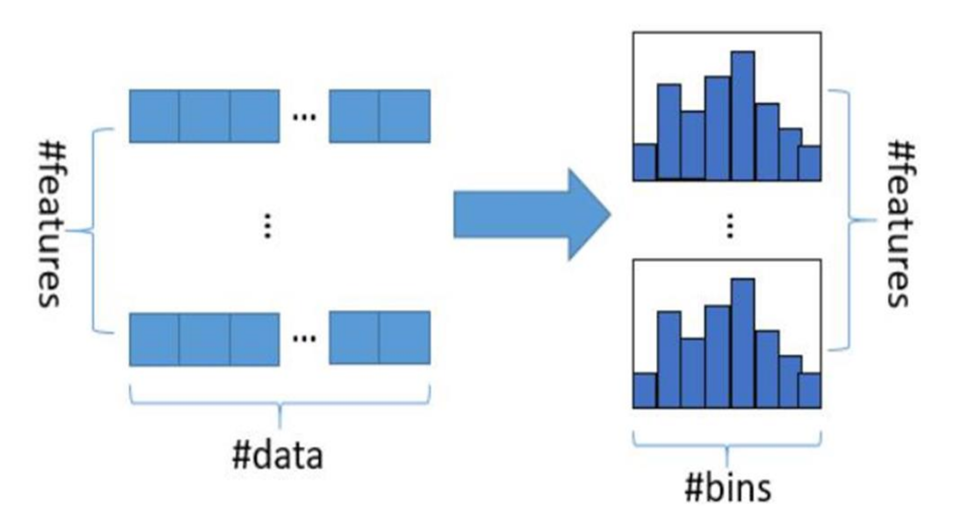
'objective': 'binary'

'learing_rate': 0.095

'n_estimators': 10000

'max_depth’: 4

'max_bin': 200



參數調整(Lightgbm)

'colsample_bytree': 0.5

'subsample': 0.5

'num_leave': 10

'reg_alpha': 25

'reg_lambda': 17

'random_state': 42

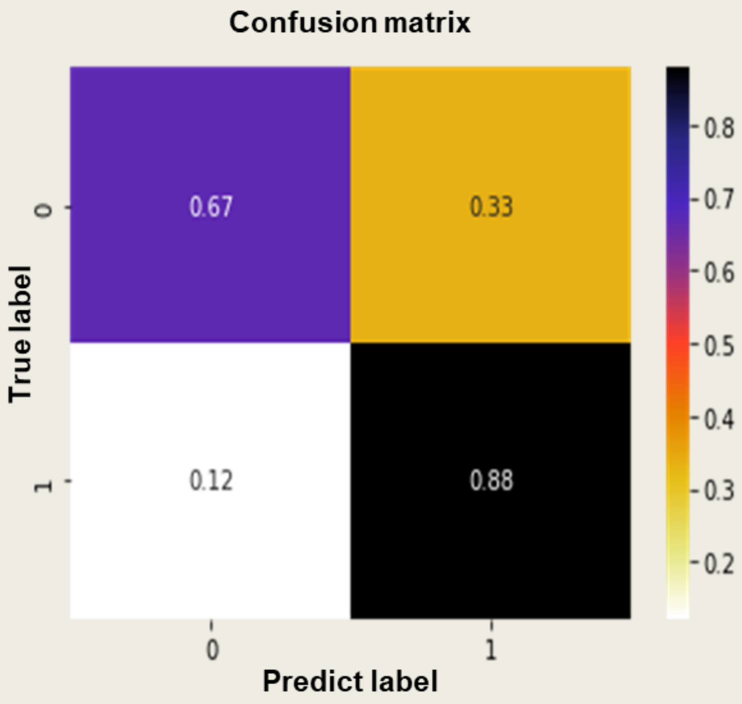
調參對比

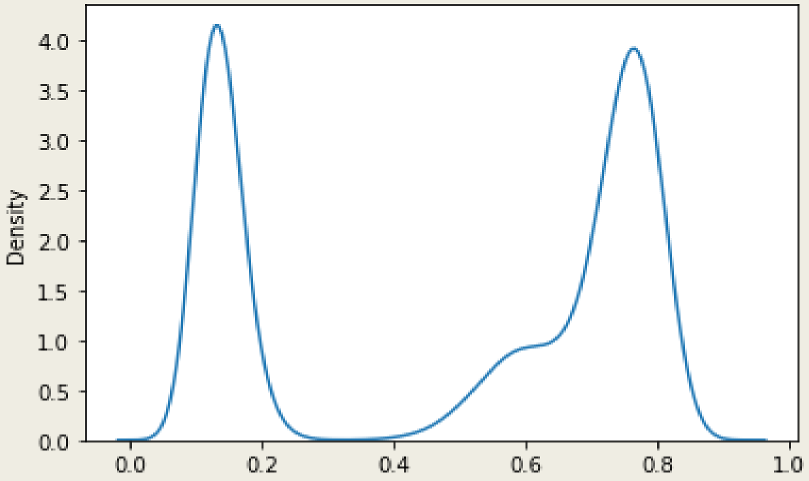
XGboost	Private score	Public score
調參前	0.81155	0.81176
調參後	0.81463	0.81579

Lightgbm	Private score	Public score
調參前	0.80169	0.80169
調參後	0.81687	0.81687

XGBOOST

K-FOLD	auc_roc
Fold 0	0.8111713807059621
Fold 1	0.8114563223182802
Fold 2	0.8106771933948776
Fold 3	0.8104075896896903
Fold 4	0.8103948440969936
Overall	0.8108214660411608



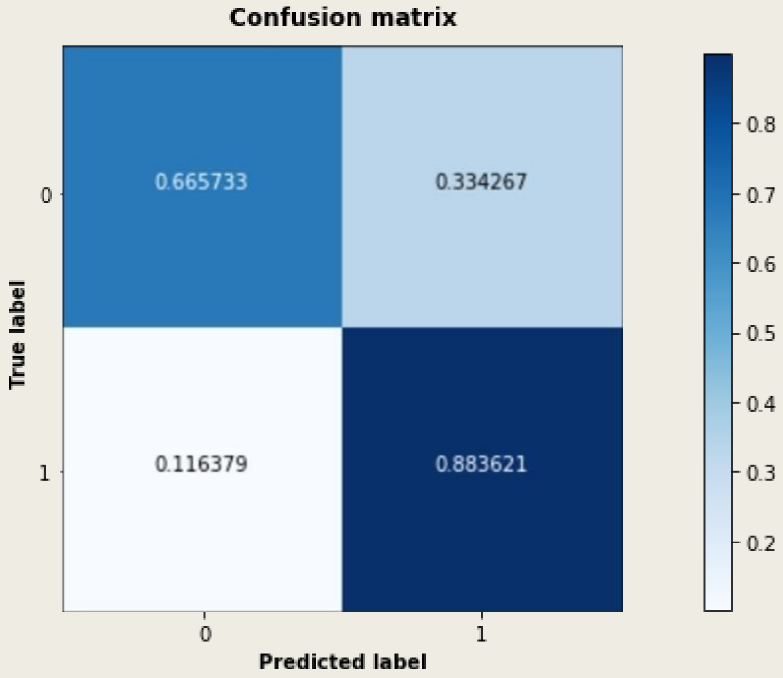


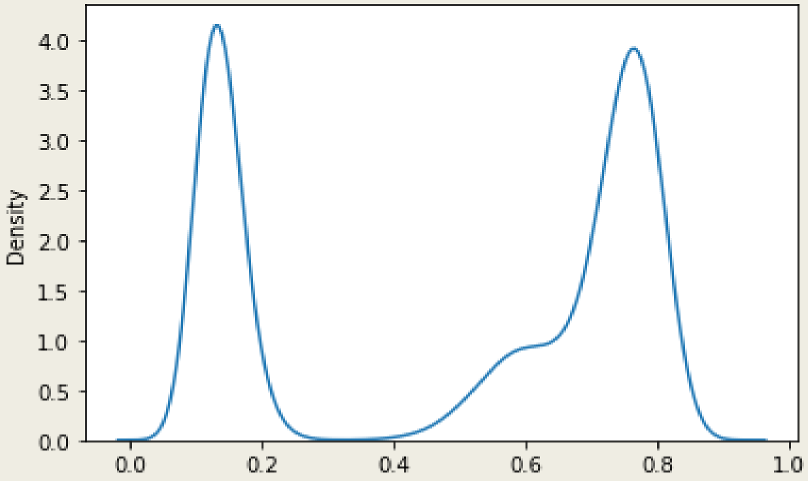
Accuracy of XGBoost: 0.7719

XGBoost

Lightgbm

K-FOLD	auc_roc
Fold 0	0.8144006843544659
Fold 1	0.8162195957334226
Fold 2	0.8148835653228536
Fold 3	0.8155307406938865
Fold 4	0.816542131451717
Overall	0.8155153435112691





Accuracy of LGBM : 0.7746

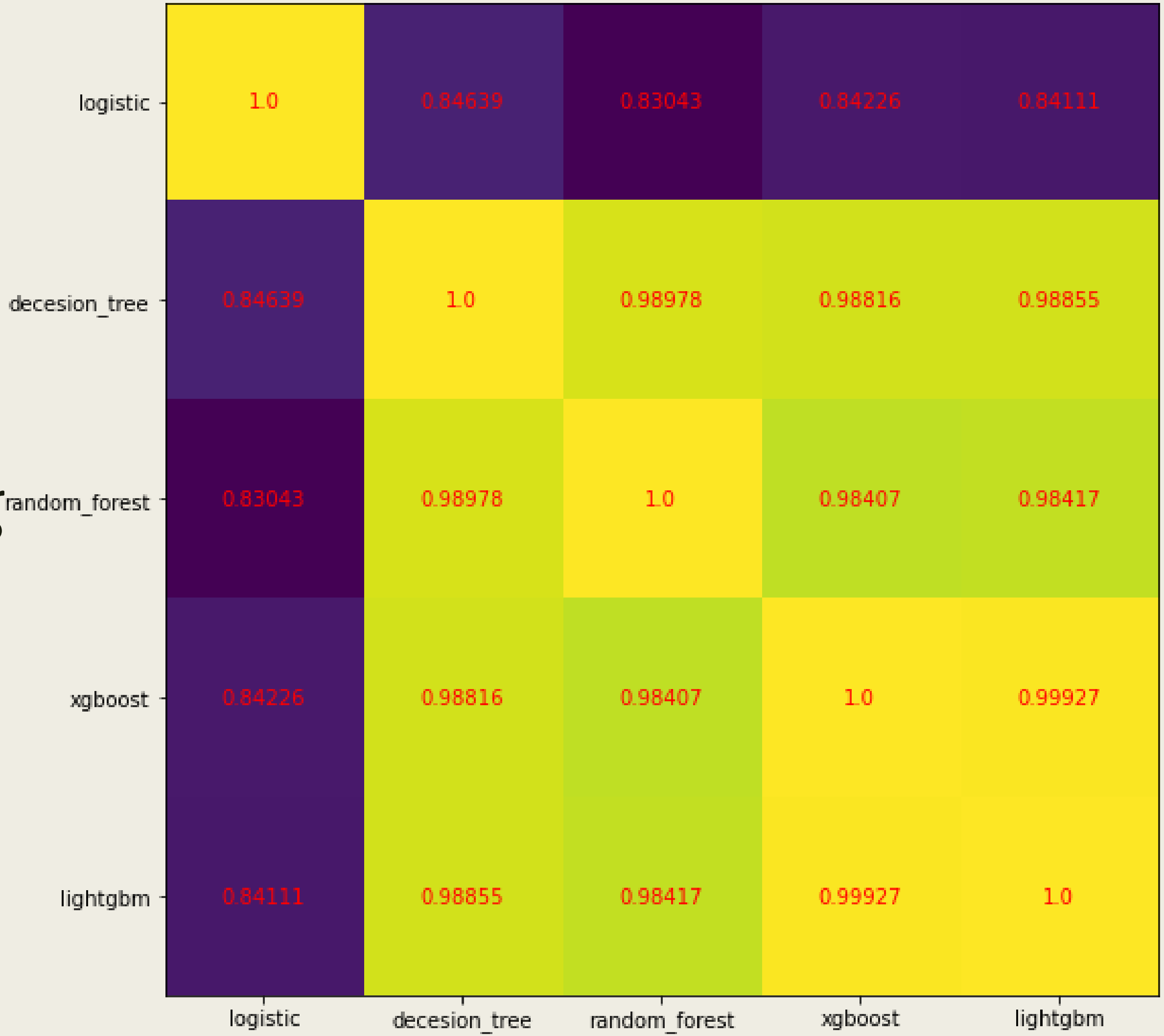
LGBM

Voting

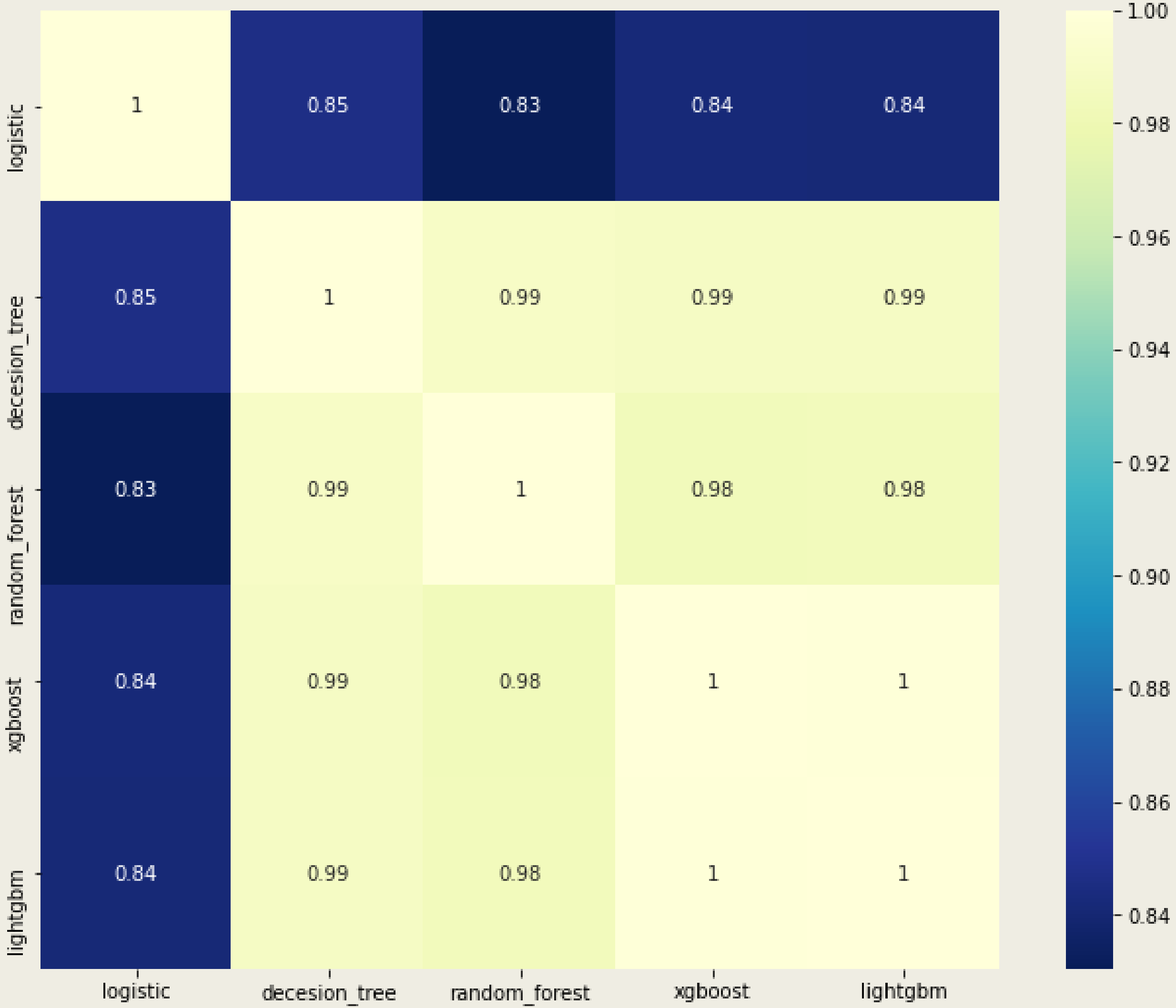
獲取每個模型的優勢，希望能有更好的結果
預測分權加總（XGBoost、Lightgbm有更多的權重）

$$\frac{2}{5}\text{XGB} + \frac{2}{5}\text{LGBM} + \frac{1}{15}\text{logistics} + \frac{1}{15}\text{cart} + \frac{1}{15}\text{Random Forest}$$

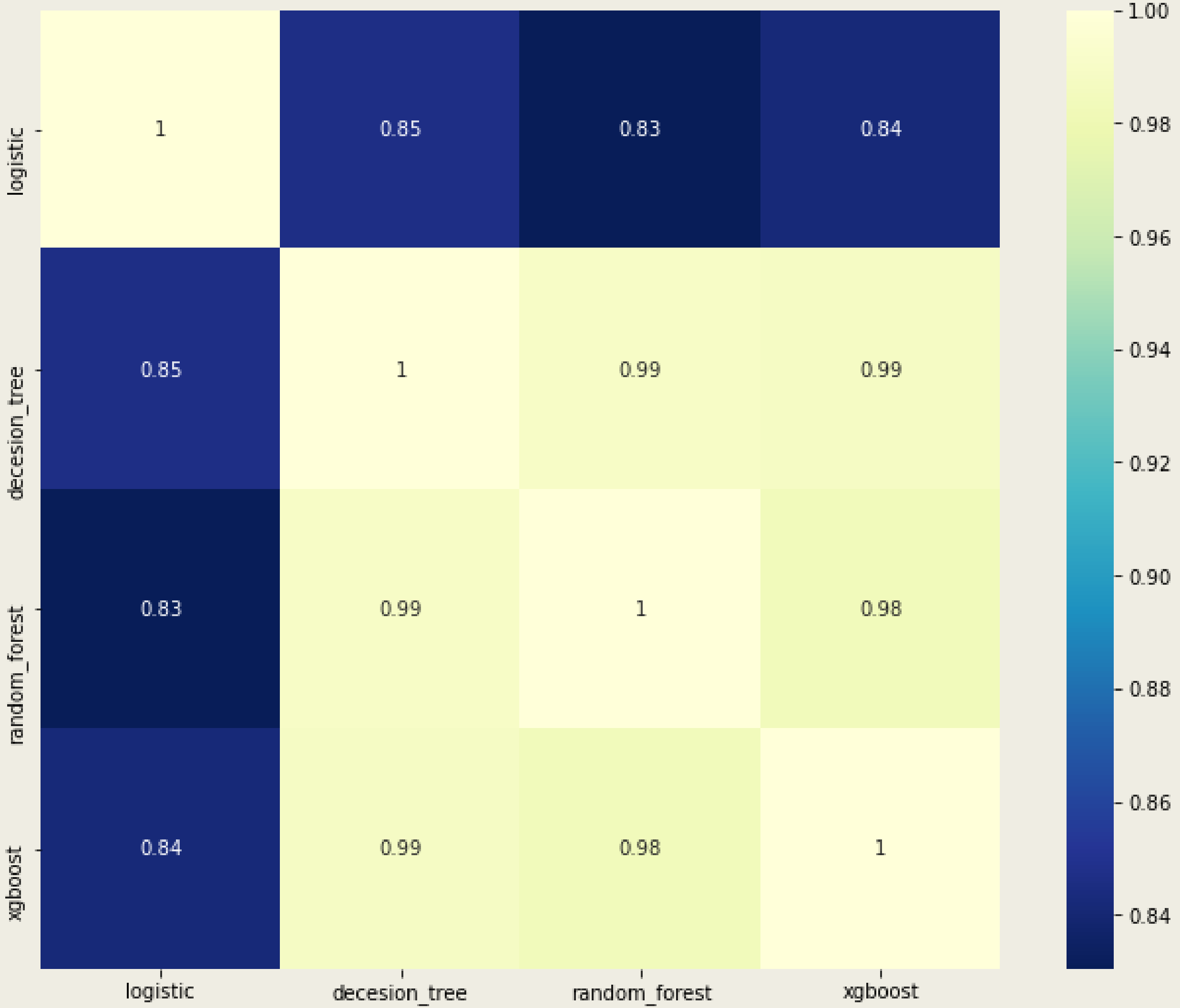
Power averaging



Lightgbm與XGBoost最高相關



XGBoost與random_forest最高相關



Correlation jumping

Lightgbm-XGBoost-Random_forest-Cart-logistic

Threshold power

Cov(Lightgbm, Random_forest) = 0.98851

Cov(Lightgbm, Random_forest)^n = 0.9

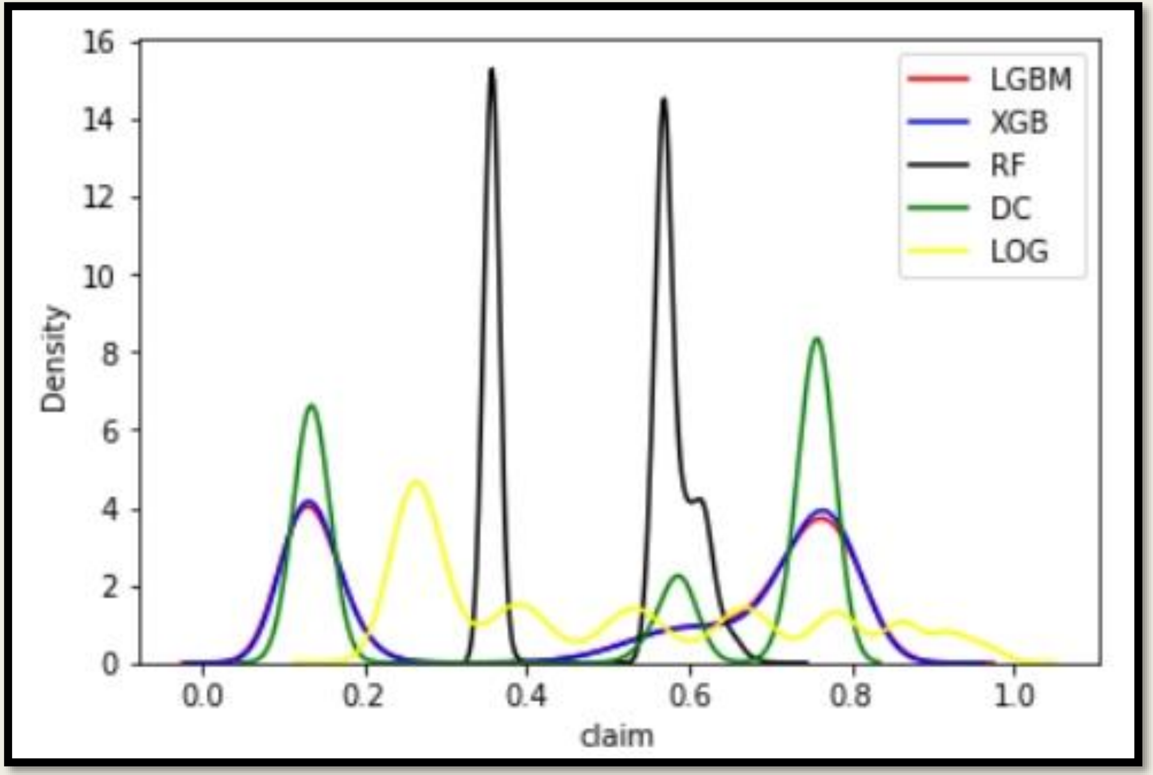
n =Power =9

Ensembling

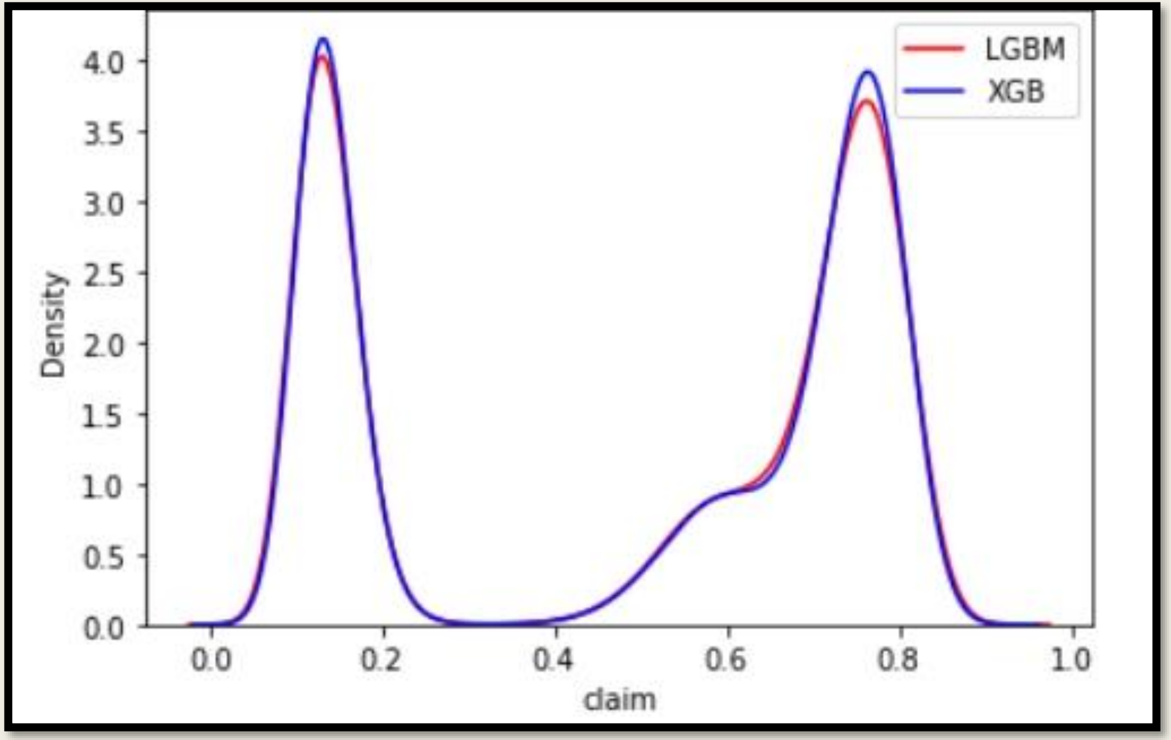
((9**xgb+9**lgb)/2+ (9**rf+9**cart))/2

模型成果

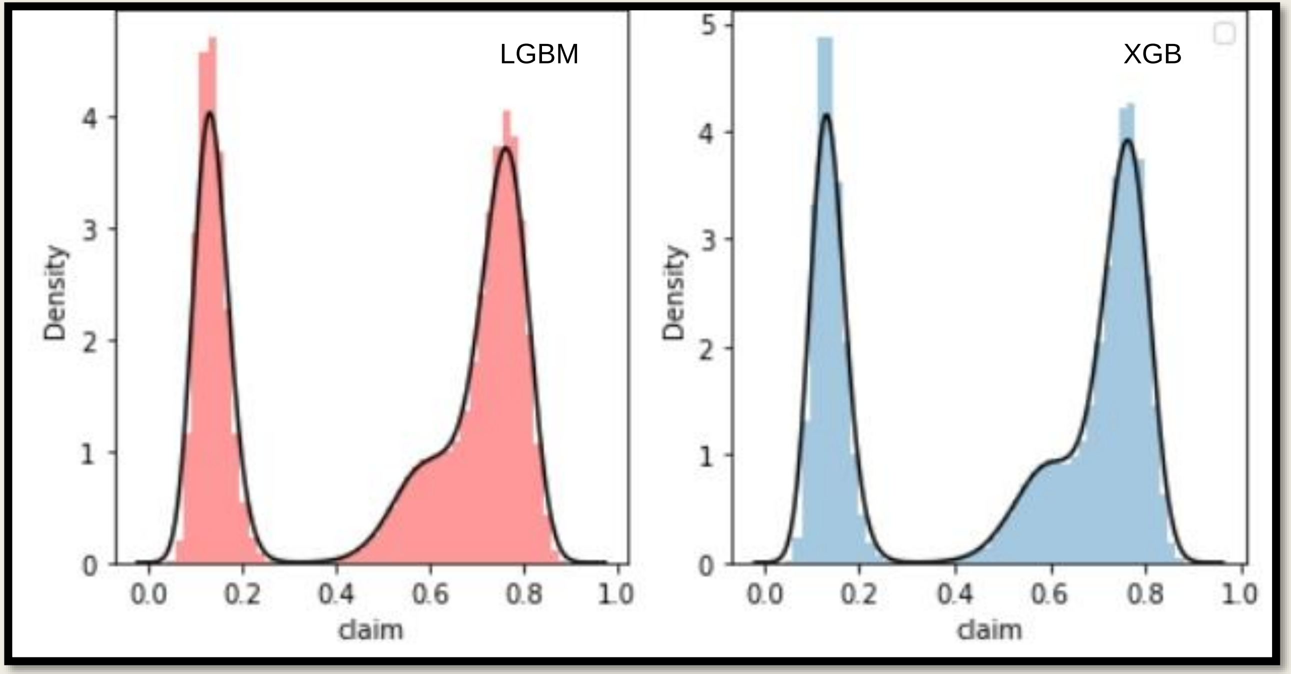
KDE



LGBM
XGB
RF
DC
LOG
16
14
12
10
8
6
4
2
0
Density
0.0
0.2
0.4
0.6
0.8
1.0
claim



LGBM
XGB
Density
claim
4.0
3.5
3.0
2.5
2.0
1.5
1.0
0.5
0.0
0.0
0.2
0.4
0.6
0.8
1.0



LGBM
XGB
Density
Density
claim
claim
0
1
2
3
4
5
0.0
0.2
0.4
0.6
0.8
1.0

	Logistic regression（羅吉斯迴歸）	random forest（隨機森林）	Decision Tree（決策樹）	xgboost	LightGBM	kaggle第一名
模型分數(ROC/AUC)	0.799	0.800	0.800	0.810	0.815	0.817
模型時間	40sec	4h	40min	4.07h	1.87h	1.53h
準確度(ACC)	0.736	0.772	0.772	0.772	0.774	NA
Kaggle得分（AUC+準確度）	0.502	0.80322	0.80239	0.81462	0.81687	0.81875

XGB vs LGBM

XGB	LGBM
預排序的決策樹演算法	直方圖演算法
Level wise	Leaf wise
不支持類別特徵	支持類別特徵
全樣本去做學習	單邊梯度樣本演算法

Voting vs Power averaging

	Voting	POW
選取模型	五個	四個
係數決定	自由選取	根據Threshold power
Kaggle得分	0.81621	0.81465