Detecting Malicious URLs

資管三 簡巧恩

Content

資料集介紹

使用DataFrame進行訓練

使用矩陣進行訓練

結果比較

資料集介紹

資料集介紹

- **FeatureTypes**

  A text file list of feature indices that correspond to real-valued features.

- **DayX.svm (where X is an integer from 0 to 120)**

  The data for day X in SVM-light format.

  +1: malicious URL, -1: benign URL.

2.4 million URLs (examples) and 3.2 million features

資料集介紹

- Lexical Feature語意資料(62%)
  - bag-of-words representation
- Host-based Feature(38%)
  - WHOIS information
  - 架設位置
  - 連線速度
  - DNS Related Properties
- No Web content related Feature
  - 相對安全
  - 運算較為輕量
- Features are Numeric Numbers

使用DataFrame訓練

擷取欄位

使用FeatureTypes中的64個欄位，並轉為DataFrame

```
# csr_matrix只擷取特定Columns
# flist為FeatureType讀出的內容陣列
# 只選取FeatureType中列出的real-valued features
# 如果選取全部的欄位再轉為DataFrame會導致記憶體不足
def getSingleDay(svm, flist):
    X = svm[:, flist]
    df = pd.DataFrame(X.toarray())
    return df

df = pd.DataFrame()
for svm in X_files:
    df = pd.concat([df, getSingleDay(svm, flist)])
```

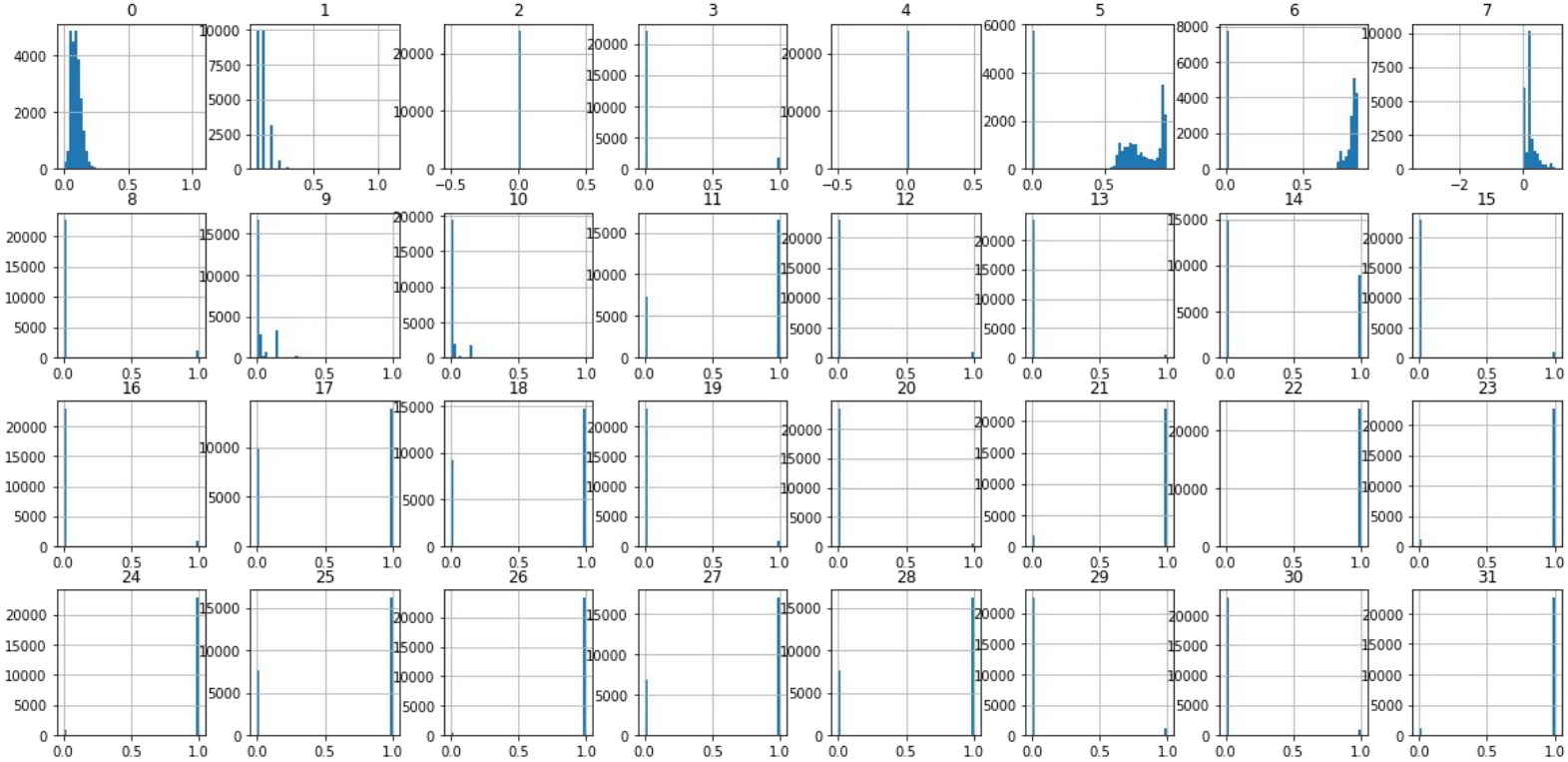
Sampling

使用LabelEncoder將y轉換為0與1（0為非惡意連結，1為惡意連結）

將惡意與非惡意進行1:1採樣

```
# Sampling
le = LabelEncoder()
df['target'] = le.fit_transform(pd.Series(np.concatenate(y_files)))
benign = df.loc[df['target']==0]
malicious = df.loc[df['target']==1]

n_samples = int(df.shape[0]*0.005)
benign = benign.sample(n=n_samples, random_state=1)
malicious = malicious.sample(n=n_samples, random_state=1)
df_sample = pd.concat([benign, malicious])

print(benign.shape, malicious.shape)
df_sample.head()
```

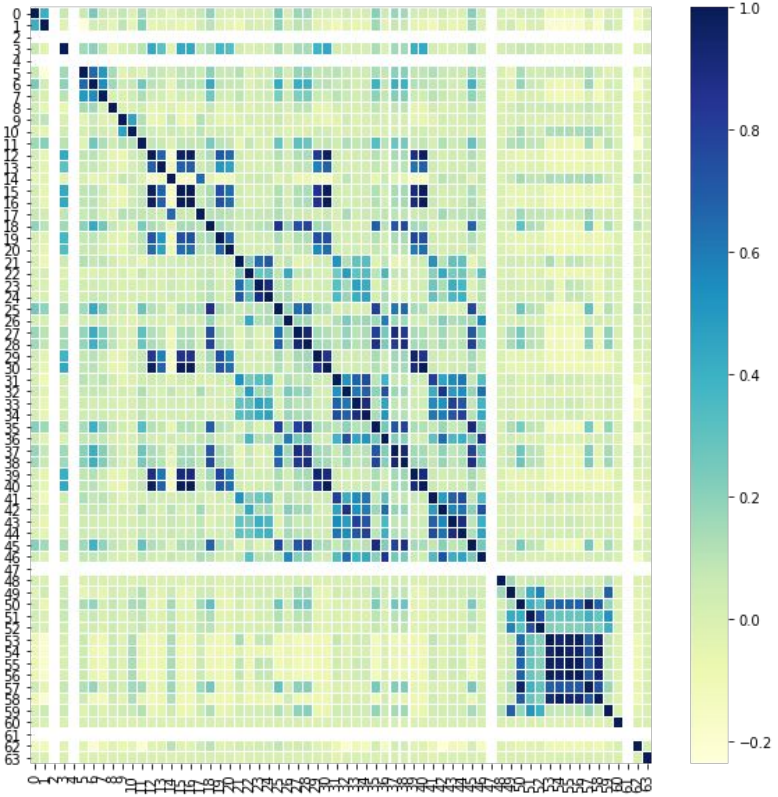
EDA - 各欄位值分佈


0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31

EDA - Correlation Heatmap

值全部為0的欄位

共線性非常大的欄位



Filter Features

NA values

沒有缺失值

MinMaxScaler

多為語意資料, 推測欄位表示為「是否有特定字串」

大部分的欄位值只有0或1

Convert to Discrete

以標準化結果, 將門檻設為0.5, 把所有Feature轉為0或1的格式

處理所帶資訊不夠的欄位

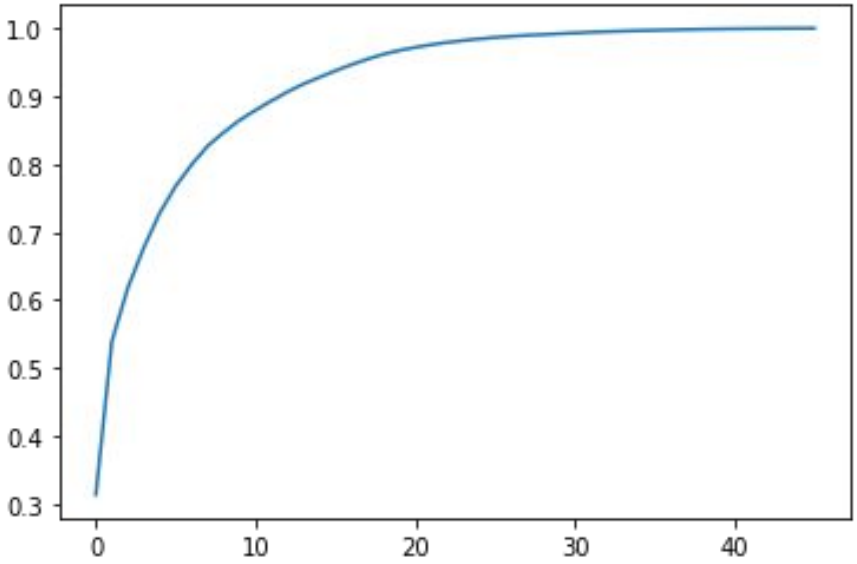
過濾欄位總和非0的欄位

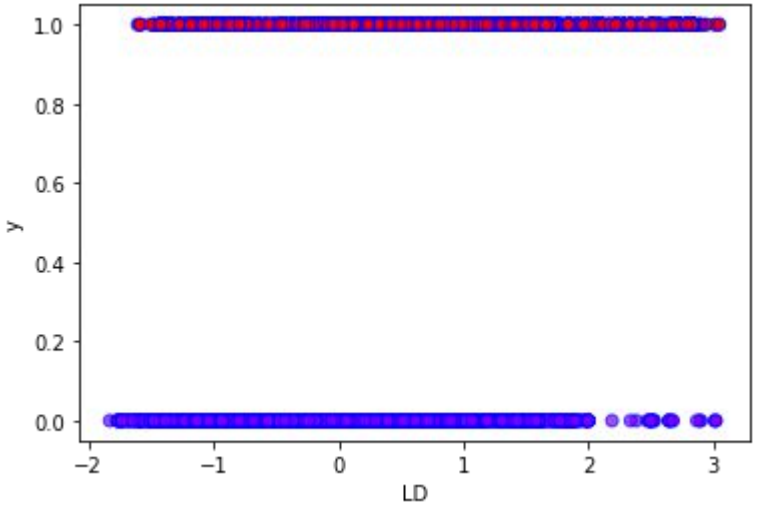
```
not_zeros_discrete = [i for i, x in enumerate(df_discrete.sum()) if x!=0.0]
df_discrete = df_discrete[:][not_zeros_discrete]
```

以Information Gain過濾欄位(Threshold=0.0001)

```
from sklearn.feature_selection import mutual_info_classif
drop_cols = []
for i, e in zip(df_discrete.columns,
                    mutual_info_classif(df_discrete, y, discrete_features=True)):
  if e < 0.0001:
    print(i)
    drop_cols.append(i)
```

Dimension Reduction - PCA, LDA, T-SNE





選用模型

SVM

Decision Tree

Logistic Regression

Neural Network

```
svm = svm.SVC()
svm.fit(X_train, y_train)

dt = tree.DecisionTreeClassifier(max_depth = 5)
dt.fit(X_train, y_train)

lr = LogisticRegression(random_state=0).fit(X_train, y_train)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 20)                100
_________________________________________________________________
dense_1 (Dense)              (None, 50)                1050
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 50)                2550
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_5 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_6 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 21
=================================================================
Total params: 12,391
Trainable params: 12,391
Non-trainable params: 0
_________________________________________________________________
```

結果 - 準確率

PCA

Model	SVM	Decision Tree	Logistic Regression	Neural Network
Cross Entropy	9.903	10.58	11.91	8.78
Accuracy	0.7133	0.6937	0.6552	0.7458
F1 Score	0.6826	0.6797	0.6593	0.7425

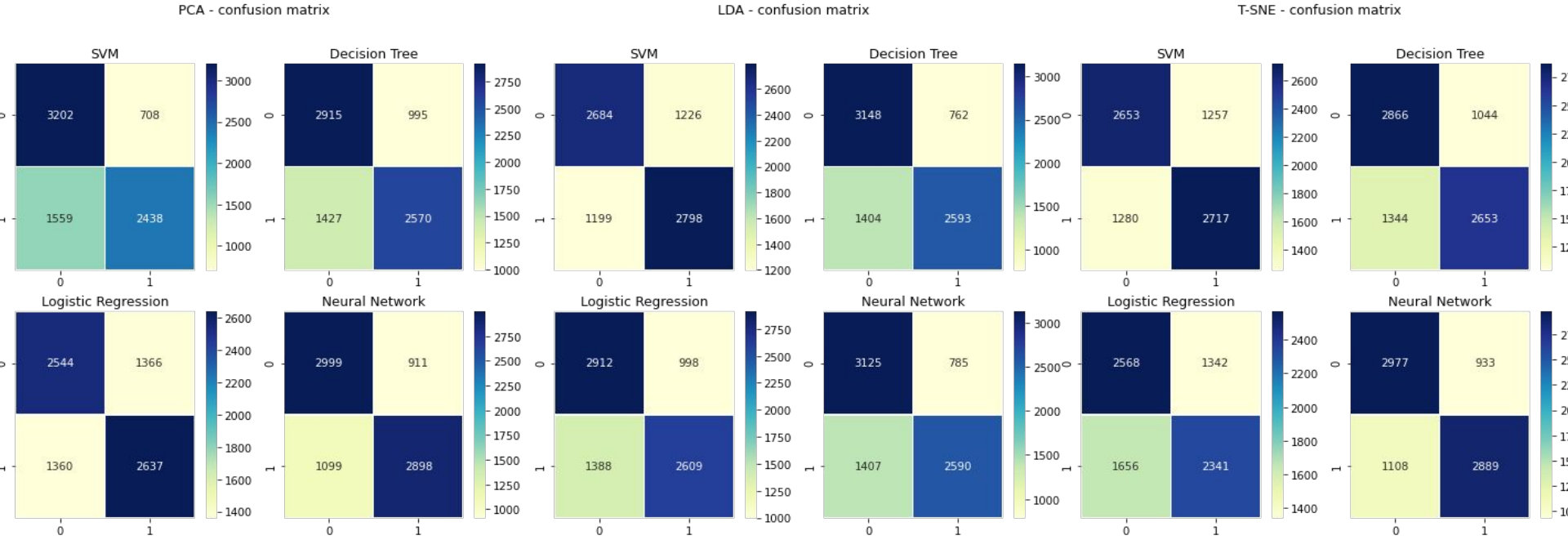
LDA

Model	SVM	Decision Tree	Logistic Regression	Neural Network
Cross Entropy	10.59	9.461	10.42	9.575
Accuracy	0.6933	0.7261	0.6982	0.7228
F1 Score	0.6977	0.7054	0.6862	0.7027

T-SNE

Model	SVM	Decision Tree	Logistic Regression	Neural Network
Cross Entropy	11.08	10.43	13.1	8.915
Accuracy	0.6791	0.698	0.6208	0.7419
F1 Score	0.6817	0.6896	0.6096	0.739

結果 - Confusion Matrix

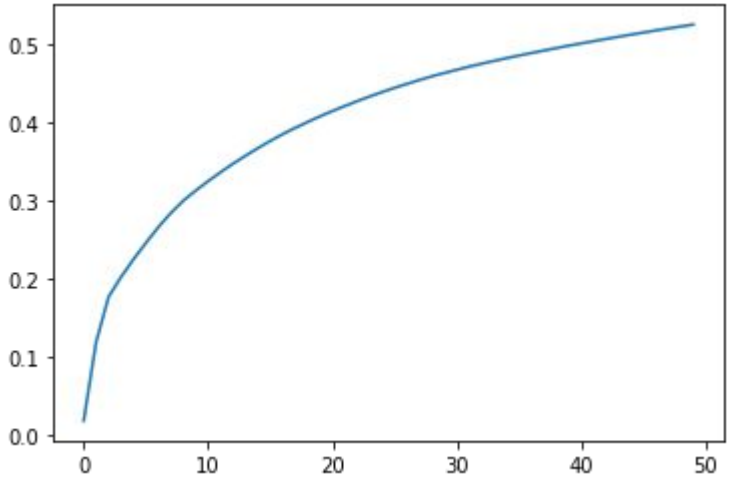


使用csr_matrix進行訓練

Filter Feature: Dimension Reduction

TruncatedSVD: 對矩陣進行降維的函式

```
from sklearn.decomposition import TruncatedSVD
# from sklearn.decomposition import PCA
svd = TruncatedSVD(n_components=50, n_iter=7, random_state=42)
x_svd = svd.fit_transform(x_csr_sampled)
```



選用模型

SVM

Decision Tree

Logistic Regression

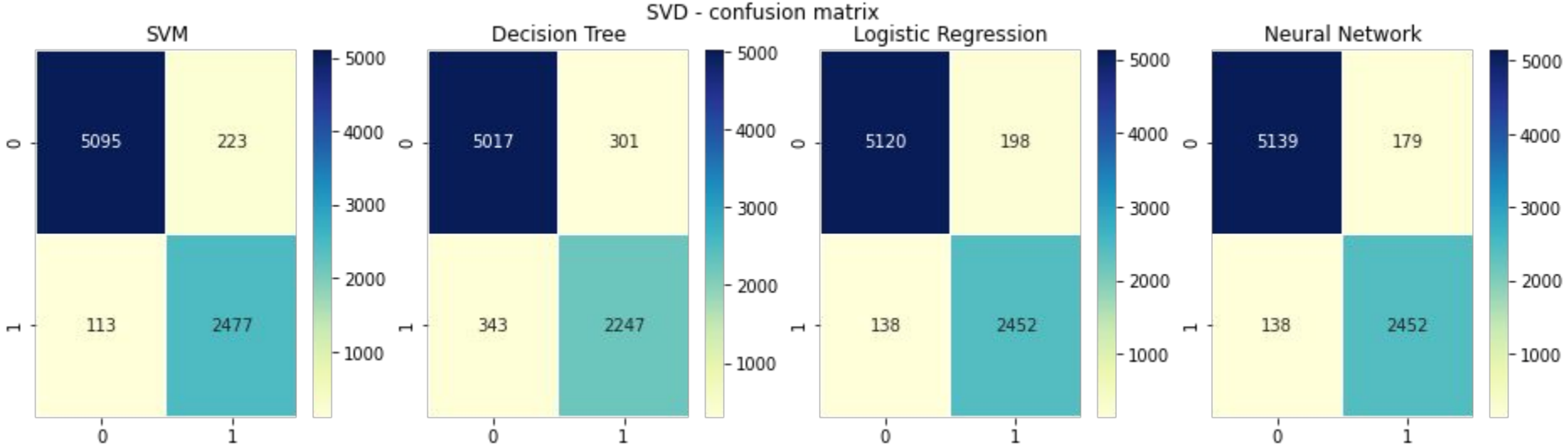
Neural Network

```
svm = svm.SVC()
svm.fit(X_train, y_train)

dt = tree.DecisionTreeClassifier(max_depth = 5)
dt.fit(X_train, y_train)

lr = LogisticRegression(random_state=0).fit(X_train, y_train)
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense (Dense)                (None, 20)                100
_________________________________________________________________
dense_1 (Dense)              (None, 50)                1050
_________________________________________________________________
dense_2 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_3 (Dense)              (None, 50)                2550
_________________________________________________________________
dropout (Dropout)            (None, 50)                0
_________________________________________________________________
dense_4 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_5 (Dense)              (None, 50)                2550
_________________________________________________________________
dense_6 (Dense)              (None, 20)                1020
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 21
=================================================================
Total params: 12,391
Trainable params: 12,391
Non-trainable params: 0
_________________________________________________________________
```

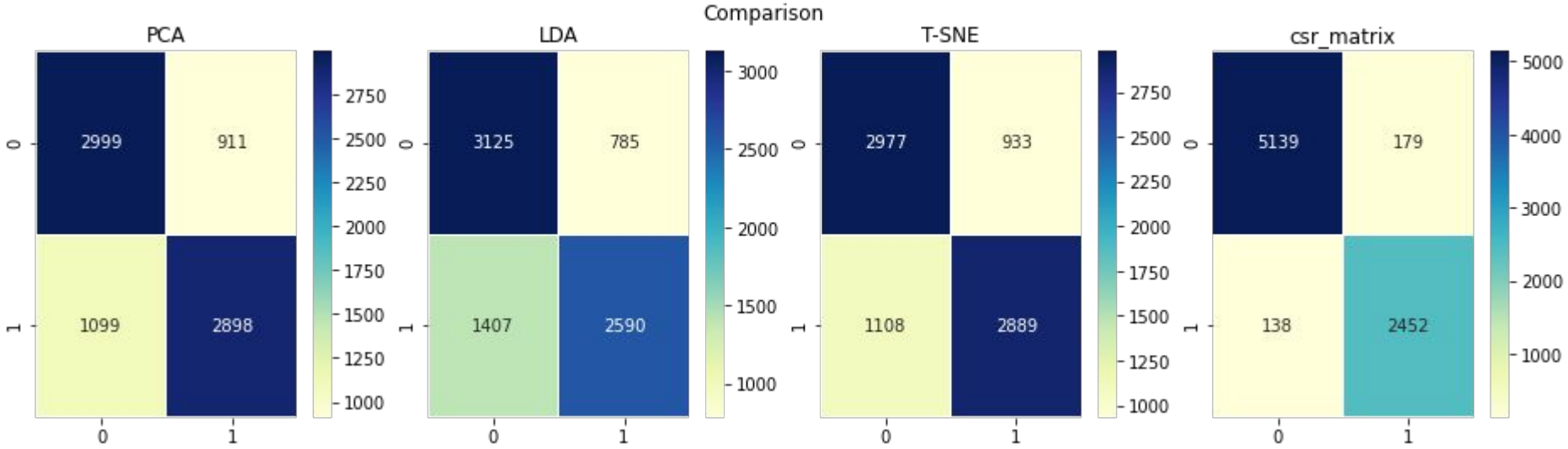
訓練結果

Model	SVM	Decision Tree	Logistic Regression	Neural Network
Cross Entropy	1.468	2.813	1.468	1.385
Accuracy	0.9575	0.9186	0.9575	0.9599
F1 Score	0.9365	0.8747	0.9359	0.9393



結果比較

Model	PCA Neural Network	LDA Neural Network	T-SNE Neural Network	SVD Neural Network
Cross Entropy	8.78	9.575	8.915	1.385
Accuracy	0.7458	0.7228	0.7419	0.9599
F1 Score	0.7425	0.7027	0.739	0.9393



謝謝大家！