HW4 Report

H24081163 陳詠翰 B14081027 劉恩兆 H24081341 耿皓昀

Introduction

這個作業的目標是要我們精準的透過模型去找到異常值。除了要比較原本指定的 13 個模型在 5 個資料集的表現以外，我們需要設計一個自己的模型盡可能讓自己的模型在不同的資料集皆贏過前面指定的 13 個模型。

以下是 hw4.pdf 的資料集描述。

	# Features	Size	# Inlier	# Outlier	% of Outlier
mnist	101	7603	6903	700	9.2
musk	167	3062	2965	97	3.2
optdigits	65	5216	5066	150	2.9
pendigits	17	6870	6714	156	2.2
satimage-2	37	5803	5732	71	1.2

Methodology

我們觀察到有些資料的變數會出現數值都一樣的情況，例如在 mnist 中有一欄所有的值都等於 0，因此，在把資料放入模型之前，我們移除了那些數值都一樣的變數。除此之外，我們在測試模型時發現有些資料集在經過 standardscaler() 的表現比較好，但有些時候沒有 standardscaler() 的表現比較好。因為沒有特別的規律，因此我們在嘗試不同模型於不同資料集時，會嘗試沒有 scale 和有 scale，並挑出我們嘗試中最好的組合作為結果。

表一、資料及移除變數前後欄數

資料集	# of features (original)	# of features (processed/removed)
mnist	101	79
musk	167	167
optdigits	65	63
pendigits	17	17
satimage	37	37

My model 描述

這次我們使用的是 combo 套件裡的 SimpleDetectorAggregator。Combo 套件裡提供許多集成學習(Ensemble Learning)的方法，SimpleDetectorAggregator 的原理就是挑選幾個異常值偵測器，將這些偵測器預測完的結果取平均，計算這些偵測器的平均表現。

SimpleDetectorAggregator 也有提供加權平均的計算方法，若有些偵測器的表現在該資料集的表現不會太好，也可以將表現弱的偵測器權重調低，訓練時 SimpleDetectorAggregator 就會更注重在權重高的模型上。

在 SimpleDetectorAggregator 中，我們最終挑選出的模型有：

- HBOS(n_bins=30)
- LOF(n_neighbors=300)
- PCA()
- IForest(n_estimators=250)
- OCSVM()

表二、SimpleDetectorAggregator 在每個資料集的模型設定

資料集	scale	權重
mnist		[0.1 , 1 , 0.5 , 0.1 , 1]
musk	✓	[1 , 1 , 1 , 1 , 1]
optdigits		[1 , 1 , 0.01 , 0.1 , 0.1]
pendigits		[1 , 0.8 , 1 , 1 , 1]
satimage	✓	[0.5 , 1 , 1 , 1 , 1]

[註] 權重由左到右分別為 HBOS、LOF、PCA、IForest、OCSVM 的權重

Baseline models 描述

下方五個表格分別代表不同模型在不同資料集的模型設定

表三、不同模型在 mnist 資料集的設定

method	scale	參數
KNN	✓	default
AvgKNN	✓	default
MedKNN	✓	default
PCA	✓	default
MCD		default
HBOS		default
ABOD	✓	default
Feature Bagging	✓	default
OCSVM	✓	kernel='rbf', gamma=0.01
LOF		n_neighbors=250,algorithm='auto', contamination=0.1
Iforest		n_estimators=100,contamination=0.1, max_samples=100
AutoEncoder		hidden_neurons=[2, 32, 64, 64, 32, 2], hidden_activation='relu', epochs=100,contamination=0.1,batch_size=96
DeepSVDD	✓	hidden_neurons=[2, 32, 32], hidden_activation='sigmoid', output_activation='sigmoid', batch_size=10, contamination=0.1

表四、不同模型在 musk 資料集的設定

method	scale	參數
KNN	✓	default
AvgKNN	✓	default
MedKNN	✓	default
PCA	✓	default
MCD		default
HBOS	✓	default
ABOD	✓	default
Feature Bagging	✓	default
OCSVM	✓	kernel='rbf', gamma=0.01
LOF		n_neighbors=100,algorithm='auto', contamination=0.1
Iforest		n_estimators=350,contamination=0.1, max_samples=100
AutoEncoder		hidden_neurons=[2, 32, 64, 64, 32, 2], hidden_activation='relu', epochs=100,contamination=0.1,batch_size=96
DeepSVDD	✓	hidden_neurons=[2, 32, 32], hidden_activation='sigmoid', output_activation='sigmoid', batch_size=100, contamination=0.1

表五、不同模型在 optdigits 資料集的設定

method	scale	參數
KNN		default
AvgKNN		default
MedKNN		default
PCA	✓	default
MCD	✓	default
HBOS		default
ABOD	✓	default
Feature Bagging		default
OCSVM	✓	default
LOF		n_neighbors=100,algorithm='auto', contamination=0.1
Iforest		n_estimators=250,contamination=0.1, max_samples=100
AutoEncoder		hidden_neurons=[2, 32, 64, 64, 32, 2], hidden_activation='relu', epochs=100,contamination=0.1,batch_size=96
DeepSVDD		hidden_neurons=[2, 32, 32], hidden_activation='sigmoid', output_activation='sigmoid', batch_size=100, contamination=0.1

表六、不同模型在 pendigits 資料集的設定

method	scale	參數
KNN	✓	default
AvgKNN	✓	default
MedKNN	✓	default
PCA	✓	default
MCD		default
HBOS		default
ABOD	✓	default
Feature Bagging	✓	default
OCSVM		kernel='rbf', gamma=0.01
LOF		n_neighbors=500,algorithm='auto', contamination=0.1
Iforest		n_estimators=500,contamination=0.1, max_samples=100
AutoEncoder		hidden_neurons=[2, 32, 64, 64, 32, 2], hidden_activation='relu', epochs=100,contamination=0.1,batch_size=96
DeepSVDD	✓	hidden_neurons=[2, 32, 32], hidden_activation='sigmoid', output_activation='sigmoid', batch_size=100, contamination=0.1

表七、不同模型在 satimage 資料集的設定

method	scale	參數
KNN	✓	default
AvgKNN	✓	default
MedKNN	✓	default
PCA	✓	default
MCD	✓	default
HBOS	✓	default
ABOD	✓	default
Feature Bagging	✓	default
OCSVM	✓	default
LOF		n_neighbors=500,algorithm='auto', contamination=0.1
Iforest		n_estimators=150,contamination=0.1, max_samples=100
AutoEncoder		hidden_neurons=[2, 32, 64, 64, 32, 2], hidden_activation='relu', epochs=100,contamination=0.1,batch_size=96
DeepSVDD	✓	hidden_neurons=[2, 32, 32], hidden_activation='sigmoid', output_activation='sigmoid', batch_size=100, contamination=0.1

Experimental Analysis

下方兩張表格分別呈現了不同模型在不同資料集的表現

表八、不同模型在不同資料集的 AUC 表現

	AUC				
data	**mnist**	**musk**	**optdigits**	**pendigits**	**satimage**
KNN	0.8416	0.8172	0.4294	0.7699	0.9437
AvgKNN	0.8257	0.4718	0.4005	0.7582	0.9274
MedKNN	0.8263	0.6154	0.4400	0.7593	0.9199
PCA	0.8506	1	0.5154	0.9383	0.9899
MCD	0.8612	0.9999	0.3688	0.8408	0.9958
HBOS	0.6917	1	0.8744	0.9202	0.9886
ABOD	0.7796	0.1193	0.5004	0.6997	0.8245
Feature Bagging	0.7058	0.6238	0.5827	0.5248	0.4622
OCSVM	0.8502	1	0.5044	0.9337	0.9981
LOF	0.8383	1	0.9222	0.8968	0.9988
Iforest	0.8070	0.9999	0.7121	0.9487	0.9955
AutoEncoder	0.8612	1	0.5023	0.9391	0.9910
DeepSVDD	0.5894	0.4862	0.5424	0.7101	0.5303
Mymodel	0.8483	1	0.9506	0.9693	0.9973

表九、不同模型在不同資料集的 Recall@N 表現

	Recall @N				
data	**mnist**	**musk**	**optdigits**	**pendigits**	**satimage**
KNN	0.4207	0.3402	0.0446	0.1315	0.3324
AvgKNN	0.4108	0.1612	0.0134	0.1273	0.2973
MedKNN	0.4148	0.2105	0.0344	0.122	0.3299
PCA	0.3785	1	0	0.368	0.8871
MCD	0.3691	0.9734	0	0.0811	0.6266
HBOS	0.1781	0.9929	0.1956	0.3738	0.7518
ABOD	0.3747	0.0372	0.0190	0.0923	0.1723
Feature Bagging	0.3296	0.3107	0.1161	0.0920	0.0287
OCSVM	0.3887	0.9931	0	0.3828	0.9394
LOF	0.4301	1	0.1796	0.1011	0.9164
Iforest	0.2983	0.986	0.0359	0.3884	0.8982
AutoEncoder	0.3861	1	0	0.368	0.8655
DeepSVDD	0.2416	0.172	0.0047	0.0823	0.202
Mymodel	0.4461	1	0.2875	0.4271	0.9198

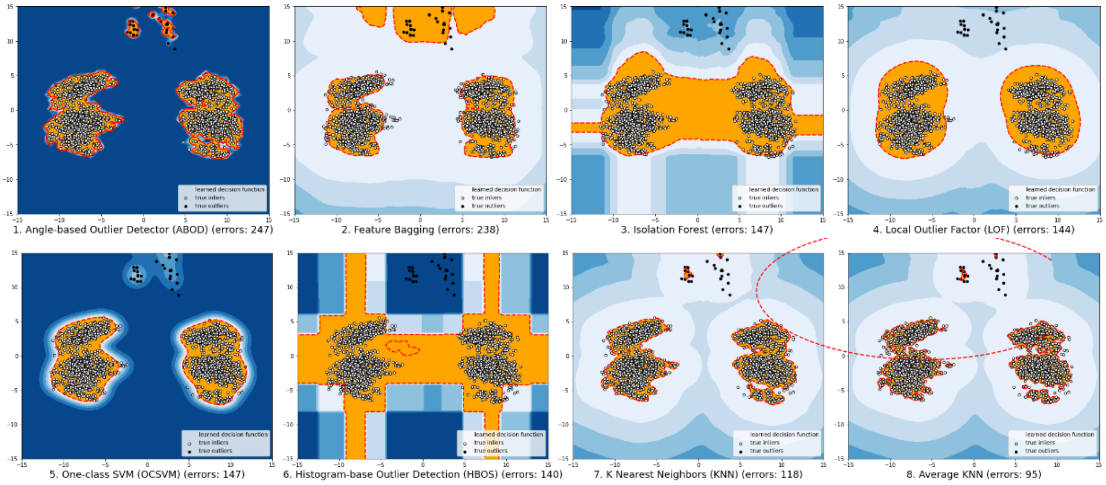
Visualization

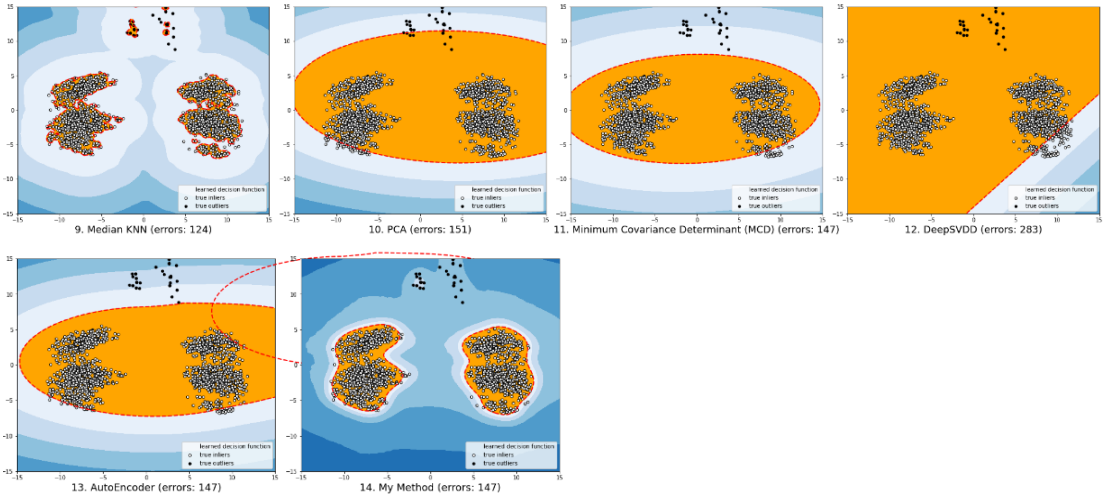
我們使用 PCA 方法降維，將資料點標示在第一主成分和第二主成分的座標圖上。由於資料集的變數從原本的數量降成只有兩個變數，因此每個異常值偵測器的表現可能和原本的結果不一樣。

以下為各資料集的視覺化圖表：

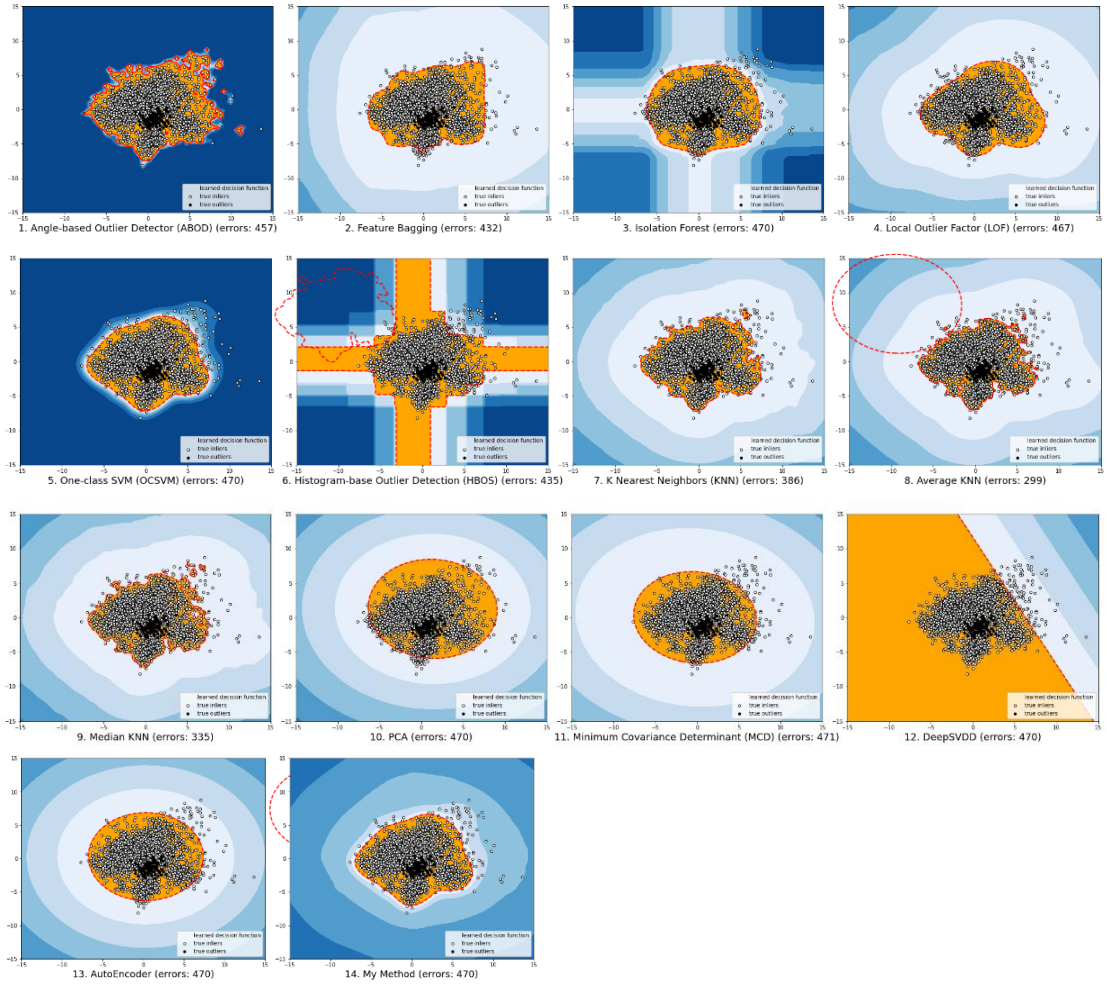
1. mnist

2. musk

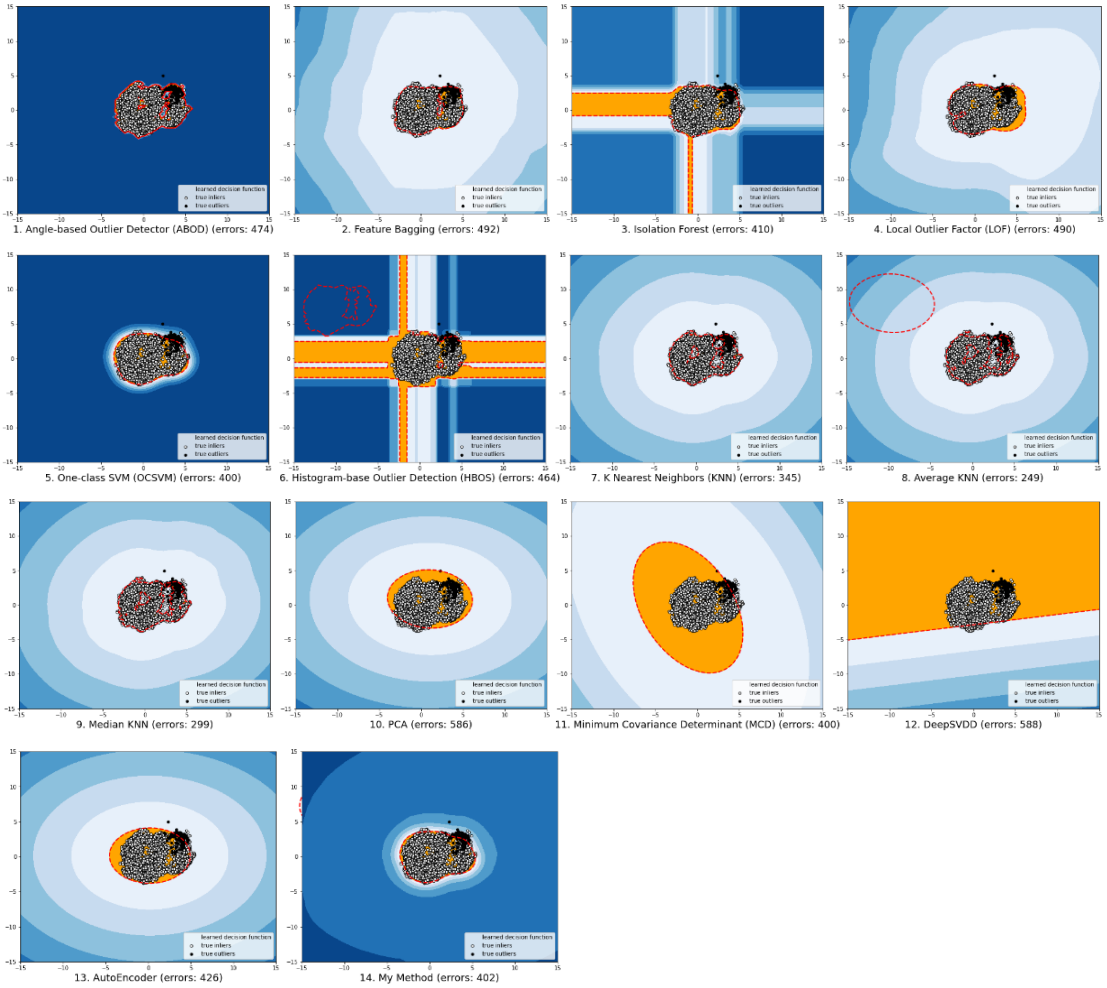



9. Median KNN (errors: 124)
10. PCA (errors: 151)
11. Minimum Covariance Determinant (MCD) (errors: 147)
12. DeepSVDD (errors: 283)
13. AutoEncoder (errors: 147)
14. My Method (errors: 147)

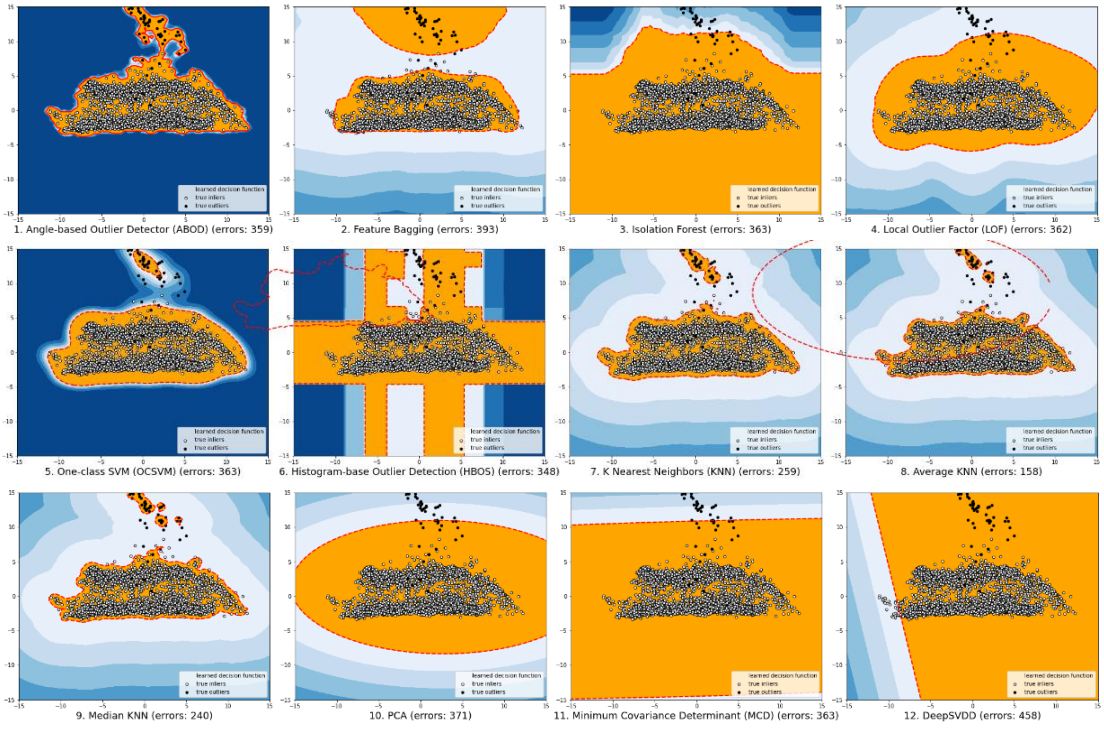
3. optdigits

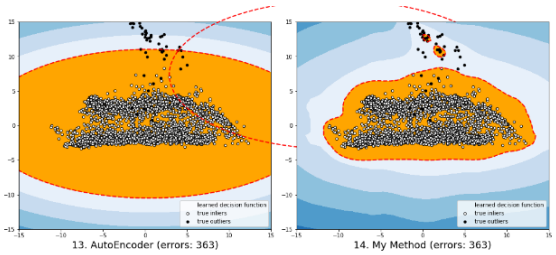

1. Angle-based Outlier Detector (ABOD) (errors: 457)
2. Feature Bagging (errors: 432)
3. Isolation Forest (errors: 470)
4. Local Outlier Factor (LOF) (errors: 467)
5. One-class SVM (OCSVM) (errors: 470)
6. Histogram-base Outlier Detection (HBOS) (errors: 435)
7. K Nearest Neighbors (KNN) (errors: 386)
8. Average KNN (errors: 299)
9. Median KNN (errors: 335)
10. PCA (errors: 470)
11. Minimum Covariance Determinant (MCD) (errors: 471)
12. DeepSVDD (errors: 470)
13. AutoEncoder (errors: 470)
14. My Method (errors: 470)

4. pendigits



5. satimage





Deeper Analysis

a. Do different methods lead to consistent outcomes over 5 datasets? Why or why not? Where are the inconsistent parts? Any insights?

同一個方法在不同的資料集表現有極大的差異。舉例來說，我們可以看到幾乎每個方法在 optdigits 這個資料集無法準確地預測異常值。以 mnist 和 musk 來看，以 KNN 為主的方法(包含 AvgKNN、MedKNN)在 mnist 的表現比 musk 好，但是 PCA、LOF、AutoEncoder 在 musk 的表現比 mnist 還好。

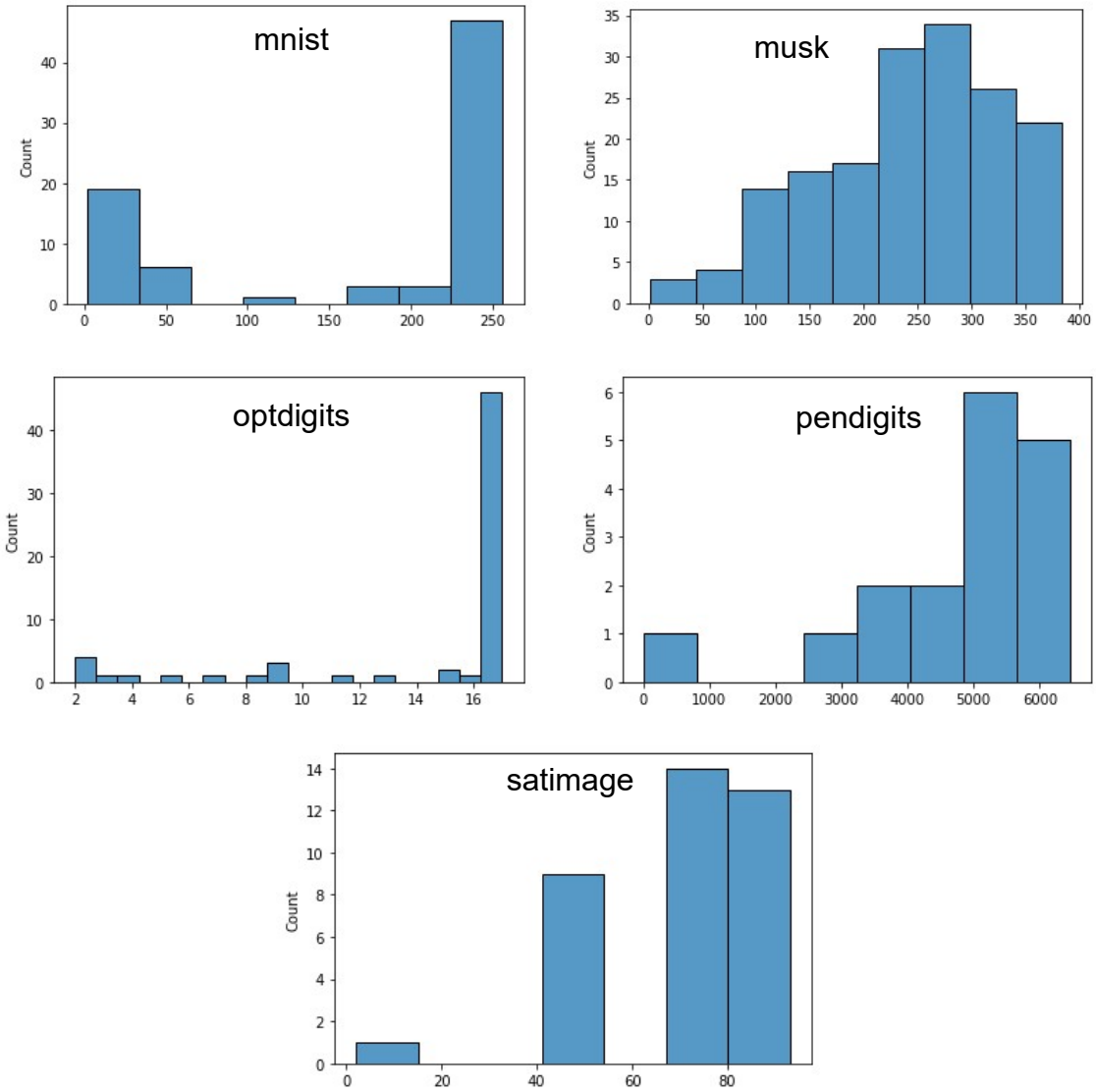
b. Which methods tend to have good results on all datasets? Why? Which tend to have worse results on all datasets? Why?

沒有一個模型是在所有的資料集中表現都很好。舉例來說，PCA 和 OCSVM 的表現很不錯但是在 optdigits 的資料集中他的 recall 是 0。又或者 HBOS 在 optdigits 和 pendigits 的表現很不錯，但是在 mnist 資料集中的 recall 僅只有 0.17 左右。LOF 的表現很不錯但是在 pendigits 的資料集中他的 recall 僅僅只有 0.1 左右。DeepSVDD 的表現在所有資料集中表現不佳，我想這是因為我們沒有花時間去調整他的參數例如說 hidden neurons 等等，導致表現不如預期。

c. Given various datasets have different characteristics, datasets with what characteristics tend to be easier for have better outlier detection results?

我們將五個資料集的變數個數畫成直方圖，從圖中可以發現資料集有較多的離散變數的情況下模型預測的結果比較差。舉例來說，optigits 的變數個數最多只有 17 個，代表說 optigits 基本上都是離散變數；同時可以發現許多方法在 optdigits 的 Recall @n 都低於 0.1，然而其他資料集基本上以連續型變數為主，且方法在其他四個資料集的最好的 Recall @n 高於 0.3，也驗證了連續型變數為主的資料集表現較佳的說法。

以下為各資料及變數個數分布圖表：



d. Which methods can successfully detect outliers that are hardly i dentified by other methods? Why?

在五個資料集之中，最難預測的兩個資料集是 optdigits 和 pendigits。相較於其他模型，HBOS 在這兩個資料集表現得非常的優異。

e. Can your own methods outperform existing methods? Why and why out? What is the novelty of your methods? Can your metho d identify difficult outlier points that cannot be detected by existi ng methods? Why?

在大部分資料集中，my model 表現可以超越 existing model。就算沒又超越也可以成為第二或第三名。My model 表現較好的原因之一我認為是偵測器的參數，例如 HBOS 在 my model 裡我把 n_bins 調成 30，LOF 也將 n_neighbors 調成 300。另一個原因是 my model 是一個更泛化的模型，從

這次作業中我們看出資料的型態時常影響到偵測器的效果，而 my model 可以同時考慮多個模型，讓在該資料集表現較好的模型可以彌補表現較差的模型，達到截長補短的效果。

f. You own methods perform better on which datasets? Why?
在 AUC score 的部分，my model 在 optdigits 和 pendigits 兩個資料集中勝出，而在 musk 資料集中則並列第一。Recall @ n score 的部分，在 musk、optdigits 和 pendigits 三個資料集中勝出，而在 musk 資料集中則並列第一。可以發現 my model 尤其在 optdigits 和 pendigits 上效果最佳，這兩個資料集的共同點就是 existing method 表現都覺很低且兩極。尤其 optdigits 資料集中，有些 existing method 的 Recall @ n score 甚至為 0，很明顯有些偵測器不適合這個資料集。因此藉由 my model 的權重調整，可以將在這些資料集中表現較佳的偵測器權重調高，以降低 my model 受到表現差的模型的影響。

Conclusions & Thoughts

詠翰： 這次作業讓我了解到原來異常值檢測也是一門很深的學問，光是 pyod 這個套件裡面就有非常多的模型，這次作業所嘗試的模型也僅僅是冰山一角。由於時間因素，很可惜的是沒辦法仔細的調整 AutoEncoder 和 DeepSVDD 這兩個深度學習的模型的參數。除此之外，和前幾次作業一樣，要想出一個 generalized model 勝過指定的模型是需要花點時間的，也很感謝隊友的付出才能完成這次作業。

恩兆： 雖然 outlier 對統計系的學生來說是個耳熟能詳的概念，但我先前做模型時卻一直沒有將其放在心上，直到這次的作業讓我對異常值檢測有了更深的理解。不論是從統計常態分配、密度、角度等方面來看，都有許多方法可以用來找出異常值，進而精進模型表現。其中，這次最讓我驚豔的是 pyod 裡面就有內建許多異常值檢測的模型，自己也在構建 my model 時嘗試了像是 COF、CBLOF 等模型，都在不同資料集中有著不錯的表現，實在是方便又實用。

皓昀： 這次作業讓我了解到異常值偵測也是一門學問。以前做機器學習模型之前，我資料前處理通常只會做標準化的動作，很少會注意到異常值的

影響。Pyod 裡面有非常多的模型，從傳統的 KNN 到最新的 AutoEncoder 等深度學習，沒想到異常值偵測也有那麼多方法。這次作業最讓我意外的是許多傳統方法效果不輸深度學習模型，甚至勝過它們。比較可惜的是沒有更仔細的調整 AutoEncoder 等深度學習模型，不然我認為結果會比原本的更好。