DSP 2021 fall

Project

姓名:吳佩玲
學號:r10944054

Part 1

1. I used:
   PyTorch / Lightning
   CPU / CUDA
   Colab / PC / Other:工作站

2. Kaggle result:

Accuracy	1.00

3. Preprocessing details. (15pts)
   因為資料有兩個維度，因此兩維分別做標準化（使用 StandardScaler function），實驗結果是：固定模型架構與其他所有變項，有做前處理的話正確率可以達到 100%，沒做前處理，正確率只有 0.33，因此前處理非常重要。

4. Method, training details (model arch., why did you select this arch?), experimental setting (hyper-parameters) (30pts):
   模型架構如下圖所示：

```
----------------------------------------------------------------
       Layer (type)               Output Shape         Param #
================================================================
           Conv1d-1             [-1, 20, 2284]             540
             ReLU-2             [-1, 20, 2284]               0
           Conv1d-3              [-1, 40, 325]           8,840
             ReLU-4              [-1, 40, 325]               0
           Conv1d-5               [-1, 80, 64]          28,880
             ReLU-6               [-1, 80, 64]               0
           Conv1d-7              [-1, 160, 12]          89,760
           Linear-8                    [-1, 3]           5,763
================================================================
Total params: 133,783
Trainable params: 133,783
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 0.99
Params size (MB): 0.51
Estimated Total Size (MB): 1.62
----------------------------------------------------------------
```

此架構下，30 epochs 訓練，正確率 1.0。

除此之外我也嘗試使用其他架構，在查閱論文時發現一篇論文，” An Improved Fault Diagnosis Using 1D-Convolutional Neural Network Model”也是做 bearing fault detection 的，我就依照論文的描述架構模型，架構如下：

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv1d-1           [-1, 128, 1999]           4,224
              Tanh-2           [-1, 128, 1999]               0
         MaxPool1d-3           [-1, 128, 1998]               0
           Dropout-4           [-1, 128, 1998]               0
            Conv1d-5             [-1, 64, 498]          65,600
              Tanh-6             [-1, 64, 498]               0
         MaxPool1d-7             [-1, 64, 497]               0
           Dropout-8             [-1, 64, 497]               0
            Conv1d-9             [-1, 32, 247]           8,224
             Tanh-10             [-1, 32, 247]               0
        MaxPool1d-11             [-1, 32, 246]               0
          Dropout-12             [-1, 32, 246]               0
           Conv1d-13             [-1, 16, 122]           2,064
             Tanh-14             [-1, 16, 122]               0
        MaxPool1d-15             [-1, 16, 121]               0
           Conv1d-16              [-1, 8, 59]             520
             Tanh-17              [-1, 8, 59]               0
          Dropout-18              [-1, 8, 59]               0
           Linear-19                  [-1, 3]           1,419
================================================================
Total params: 82,051
Trainable params: 82,051
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.12
Forward/backward pass size (MB): 9.07
Params size (MB): 0.31
Estimated Total Size (MB): 9.51
----------------------------------------------------------------
```

會想嘗試實作這篇的架構是因為他有實驗很多的參數，例如文中說：模型架構的 channel 數由大慢慢縮小會有較好表現，剛好跟第一個實作的模型相反，因此很有測試價值。我用 50 個 epoch 作訓練，Kaggle 上的正確率 0.99，並沒有比第一個架構用 30 epochs 的結果更好，因此沒有採用。

另外他在論文有提到一些不同超參數實驗結果，例如 activation function 用 Tanh 比用 ReLU 好、有 Dropout 也有較好表現等等，我在第一個模型架構下把 activation function 都換成 Tanh，正確率 0.62，結果並沒有比較好。

由於兩次測試都沒有比較好，我認為這篇論文在我的題目上參考價值較低，因此沒有繼續實驗下去。

5. What have you learned (Interesting Findings and Special Techniques)? (30pts)

第四小題的論述中，我原本以為，第一個架構雖然簡單，但也有好表現，那我用較複雜的第二個架構應該可以學到更全面的特徵，尤其又有用到 Dropout 層、MaxPooling 層等等目前大家建模型會用的技巧，感覺應該要有比較好的表現，至少跟第一個一樣好（正確率 1.00），但並沒有，因此得到結論：模型的好壞是 highly data dependent 的，在其他領域資料可以比現好的技巧不一定可以適用在我們想要的領域，還是要多做實驗。

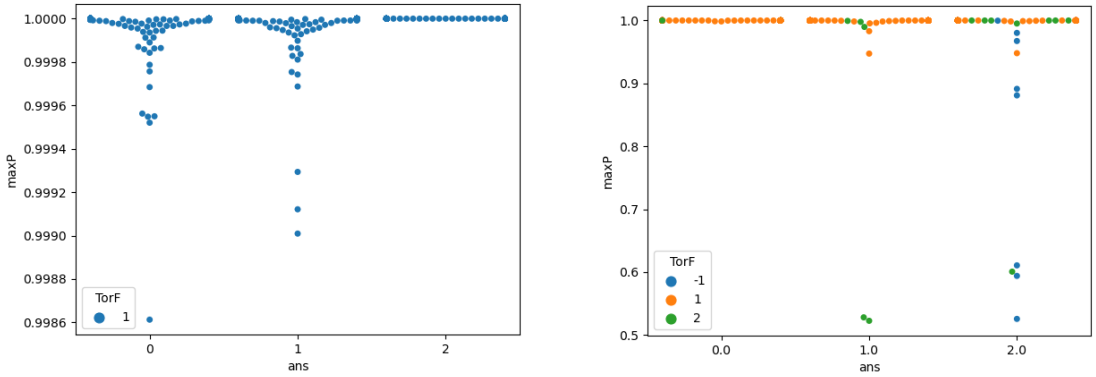
Part 2

1. Explain your method and result in detail. (7pts)

Accuracy	0.48

總結來說，我沒有達到很好的正確率。

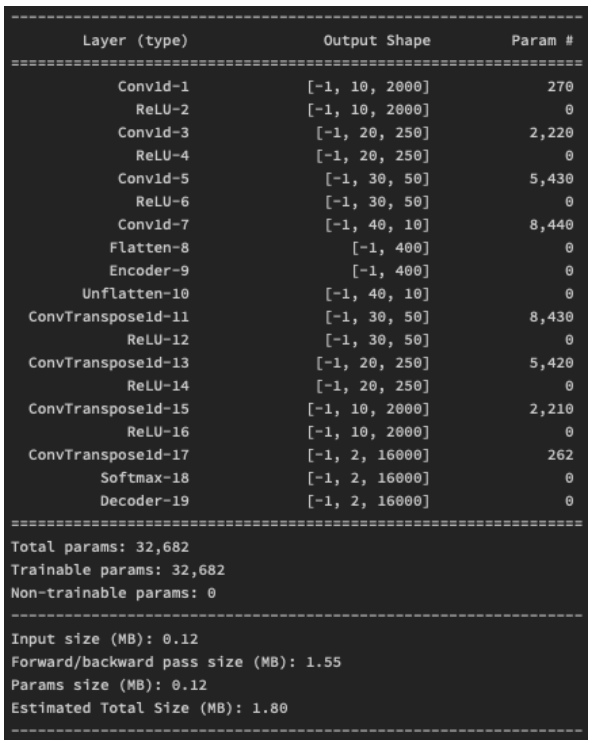
- 一開始我去檢查 task 1 的預測機率，發現模型在預測 class 時都是很有自信的，大部分都有高機率在某個 class，另外兩個的機率就很小，因此我猜想如果丟進 anomaly data，模型或許在各類別的機率就會相對平均一點，如此則我可以透過他預測的最大機率來判斷是否為 anomaly data。如下圖所示，左圖是 training set，右圖為 testing set & anomaly set。橫軸是取最大機率值，縱軸是 class，TorF 是答對或答錯，1 代表答對，-1 代表答錯，2 代表 anomaly class。



可以發現最大機率值並沒有辦法很好的區分 anomaly data，即使有比較多點機率是比較小的，但有很多點他的預測機率還是逼近 1，因此推論，task 1 的 model 沒辦法直接拿來找 anomaly class。

- 接著我嘗試幫 class0, 1, 2 建 autoencoder model，所以如果 anomaly data 進來，經過 encoder 和 decoder 後重建的結果應該會和原始資料很不像，藉此發現異常資料。我第一個嘗試的是 PCA，PCA 的結果是第一軸有 30%左右的解釋力，但以下的都只有 0.x%的解釋力。我猜想是不是 16000*2=32000 維太大，無法收斂，因此我使用第一個模型將資料降維到 1920 維，再做 PCA，結果差不多，eigenspace 在辨識 training data 無法有好表現，因此放棄這個方法。

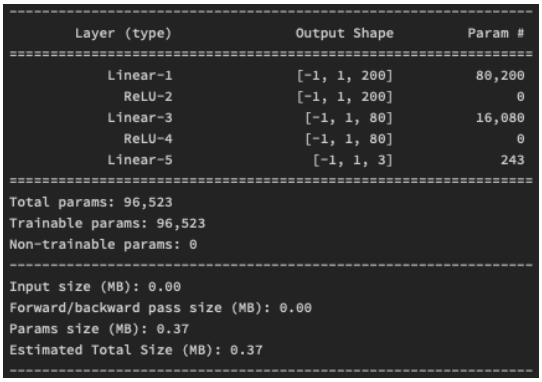
- 第三個嘗試的方法是 Autoencoder，設計架構如下：左圖是 Autoencoder，右圖是輸入 encoder 輸出的 400 維，輸出 3 維的 classifier。



Layer (type)	Output Shape	Param #
Conv1d-1	[-1, 10, 2000]	270
ReLU-2	[-1, 10, 2000]	0
Conv1d-3	[-1, 20, 250]	2,220
ReLU-4	[-1, 20, 250]	0
Conv1d-5	[-1, 30, 50]	5,430
ReLU-6	[-1, 30, 50]	0
Conv1d-7	[-1, 40, 10]	8,440
Flatten-8	[-1, 400]	0
Encoder-9	[-1, 400]	0
Unflatten-10	[-1, 40, 10]	0
ConvTranspose1d-11	[-1, 30, 50]	8,430
ReLU-12	[-1, 30, 50]	0
ConvTranspose1d-13	[-1, 20, 250]	5,420
ReLU-14	[-1, 20, 250]	0
ConvTranspose1d-15	[-1, 10, 2000]	2,210
ReLU-16	[-1, 10, 2000]	0
ConvTranspose1d-17	[-1, 2, 16000]	262
Softmax-18	[-1, 2, 16000]	0
Decoder-19	[-1, 2, 16000]	0

Total params: 32,682
Trainable params: 32,682
Non-trainable params: 0

Input size (MB): 0.12
Forward/backward pass size (MB): 1.55
Params size (MB): 0.12
Estimated Total Size (MB): 1.80



Layer (type)	Output Shape	Param #
Linear-1	[-1, 1, 200]	80,200
ReLU-2	[-1, 1, 200]	0
Linear-3	[-1, 1, 80]	16,080
ReLU-4	[-1, 1, 80]	0
Linear-5	[-1, 1, 3]	243

Total params: 96,523
Trainable params: 96,523
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.00
Params size (MB): 0.37
Estimated Total Size (MB): 0.37

原本的想法是訓練一個 generator，且預期只用 training data 做訓練，anomaly data 解碼的效果就會很差，藉此先區別出 class 3，再抽取 bottleneck 的壓縮碼進行 task1 的分類，Autoencoder 訓練 100 epochs，classifier 訓練 30 epochs。但結果並不好，Autoencoder 的 training loss 到 1.04，classifier 的預測正確率也只有 0.63。

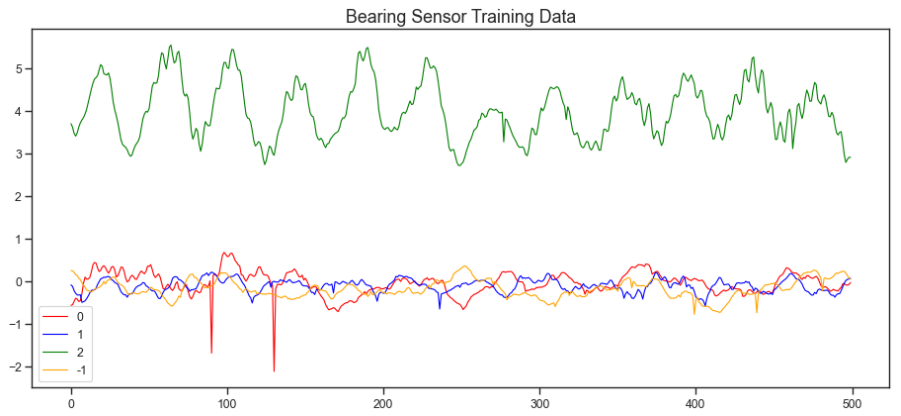
- 第四個嘗試的是 KNN，預期應該分三類，測試時有實驗 1~10 個 neighbor 的效果，但是正確率很慘，效果不佳。如下圖，我將訓練資料 7:3 分，三成作為 test，KNN 跑完好再回頭預測那七成的訓練資料，只有 0.62 的正確率，strong error 的意思是他預測錯誤並且是以大於五成的機率預測為該類，can't find class 即小於五成。可以看到 test 的正確率只剩 0.33，放棄此方法。

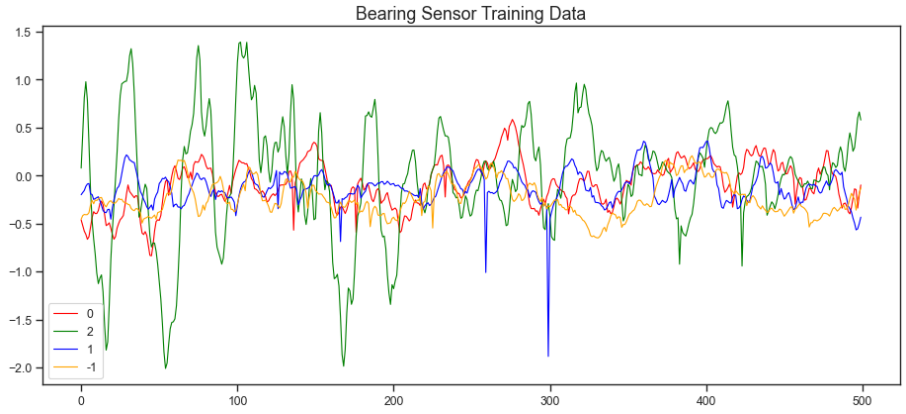
```
train error rate: 0.38095238095238093
train strong error:  581
train can't find class:  379
test error rate: 0.6722222222222223
test strong error:  560
test can't find class:  166
```

- 最後一個嘗試的是 One Class SVM，將訓練資料完全倒進去可以獲得約 0.68 的正確率，與同學討論之後嘗試去調整訓練資料的比例，例如丟進 1200 筆 class 0 的資料以及 300 筆 class 1 的資料，並設定 nu=0.2 去讓他學到 class 0 資料特徵，如此我可以 ensemble 多個 One Class SVM model 來預測。但是試過各個 class 的組合都沒有好的效果，三個 class 沒有辦法分開。猜想有可能是維度太大（32000 維）因此先對訓練資料做 convolution 後才丟進去分類，效果有變好一點，但還是沒辦法有像

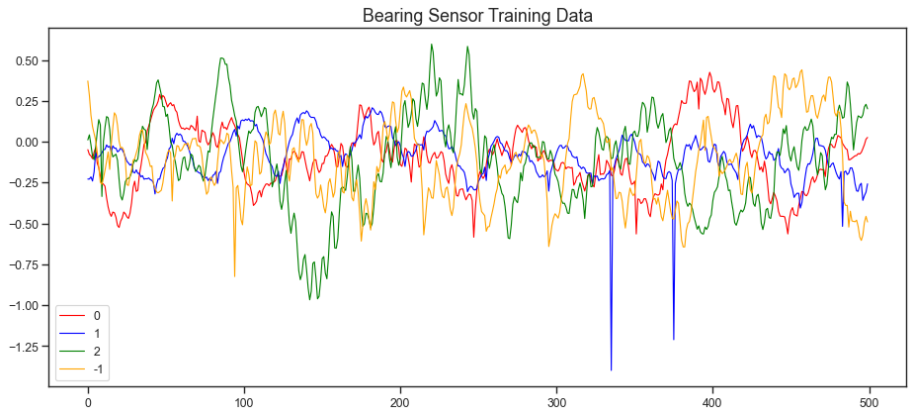
task 1 的模型一樣有非常顯著的效果。可以提的是原本我都有設定 gamma，但後來發現使用 default 的 gamma 會有比較好的效果。

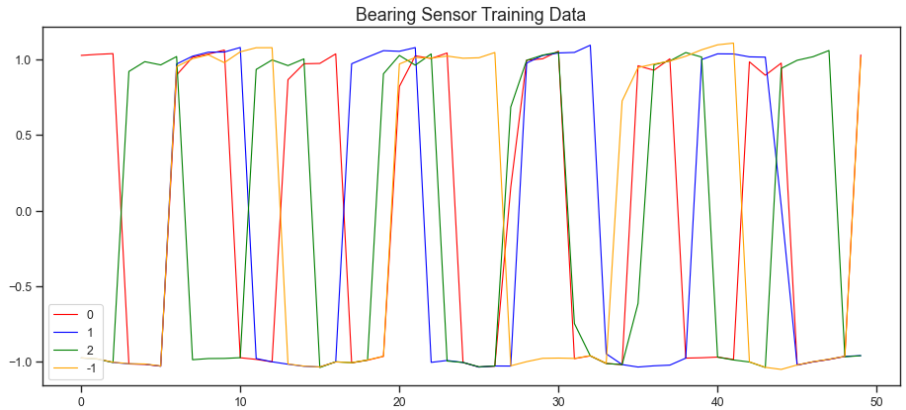
做了非常多嘗試但是效果都很差，我決定回頭來看資料的統計特性。

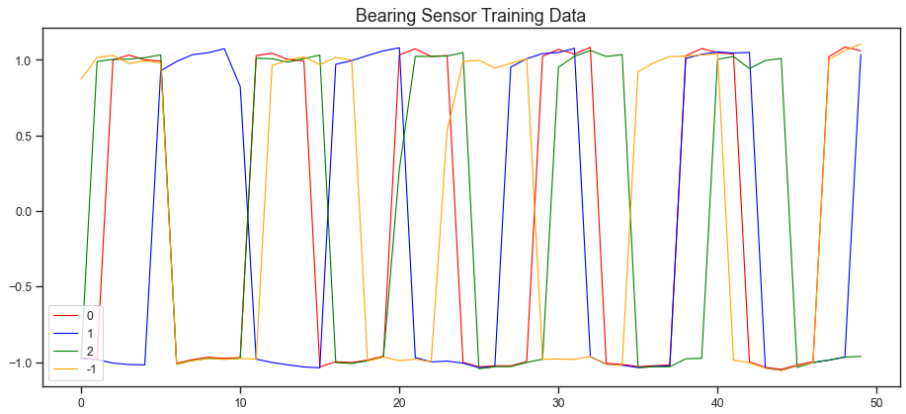


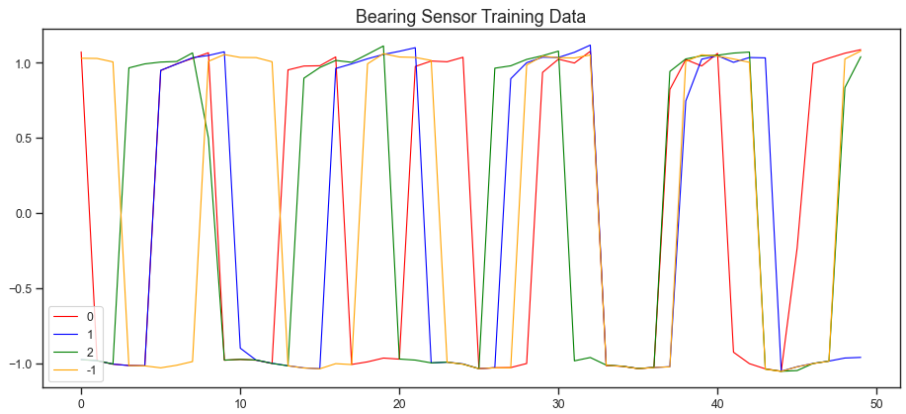


左邊的圖為隨機選取的 training sample（每一類選一筆資料，橘色-1 代表 anomaly），可以輕易地發現 class 2 有很大的起伏或者平均和其他組差很多。class 1 偶爾會出現極大的 spike（雖然正常組也會有），然而 anomaly 組在沒有背景知識下我看不出他有什麼特徵。

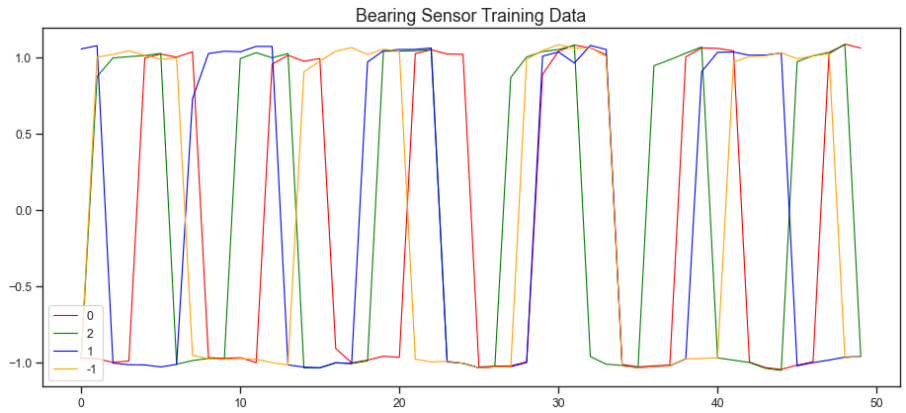




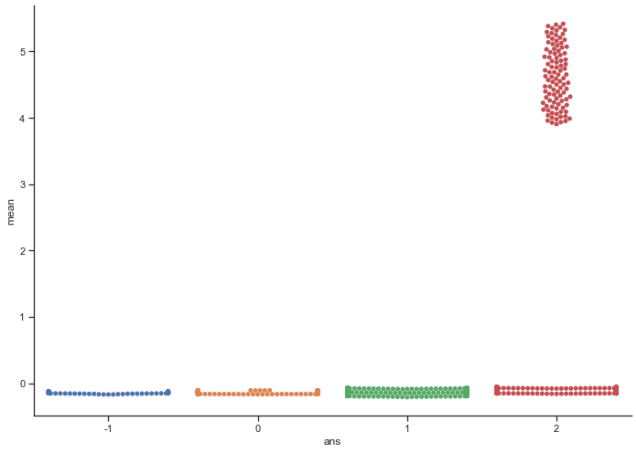




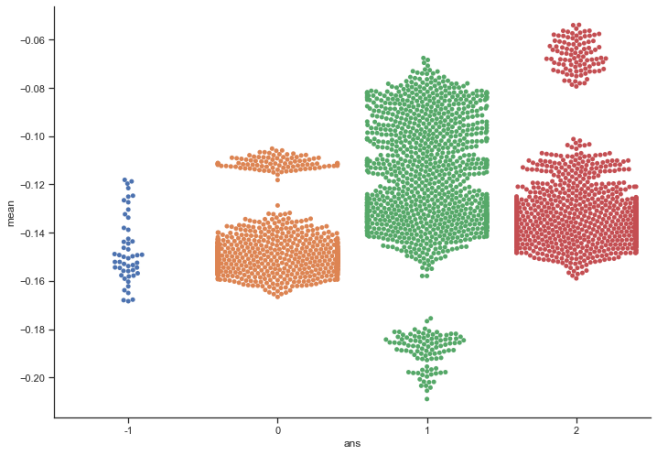
訓練資料第二個維度的資料可以發現 anomaly data 的類似方波的寬度比起其他還要更大，但是可以看到有一些 class 1 & class 2 的波寬也很大，有點難以區別。因此以下主要先來看第一維資料的統計特性。



首先統整各個 sample 的 mean & variance，如下圖：

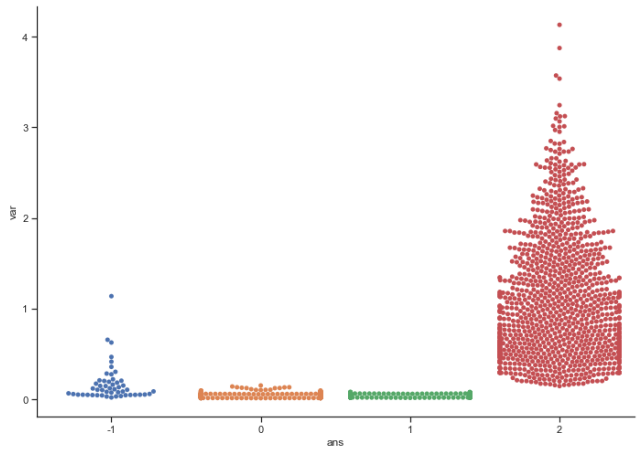


可以發現 class 2 有一群 mean 很高的資料點。去除高平均值的點，看在零附近的點的分佈狀況：

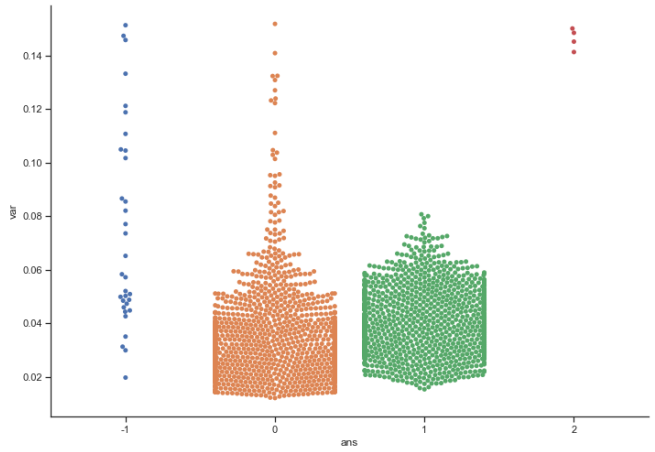


四個組別沒有差太多。

接著是 variance。

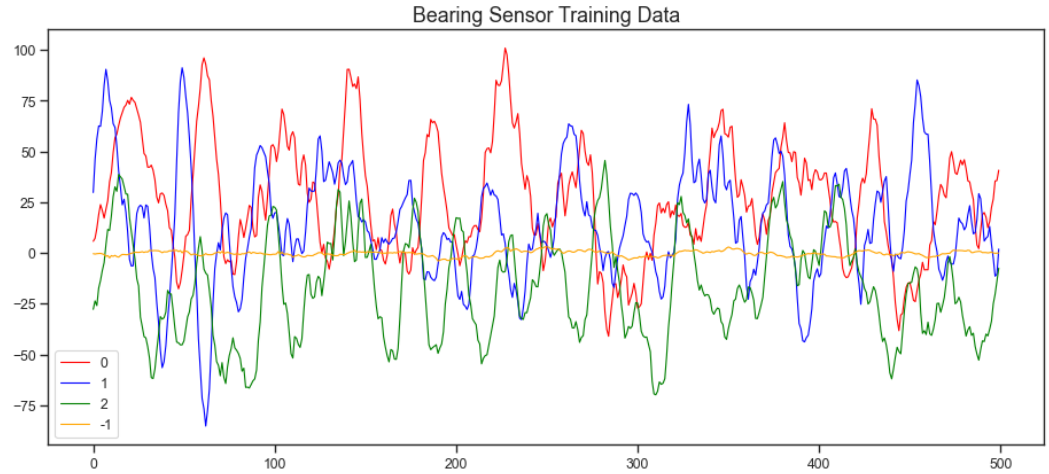


Class 2 的 variance 幾乎都恨大，然而 anomaly 組的 variance 在高與低之間都有分布。縱軸用 class 0 的最大值 0.1517 作為上界看看：

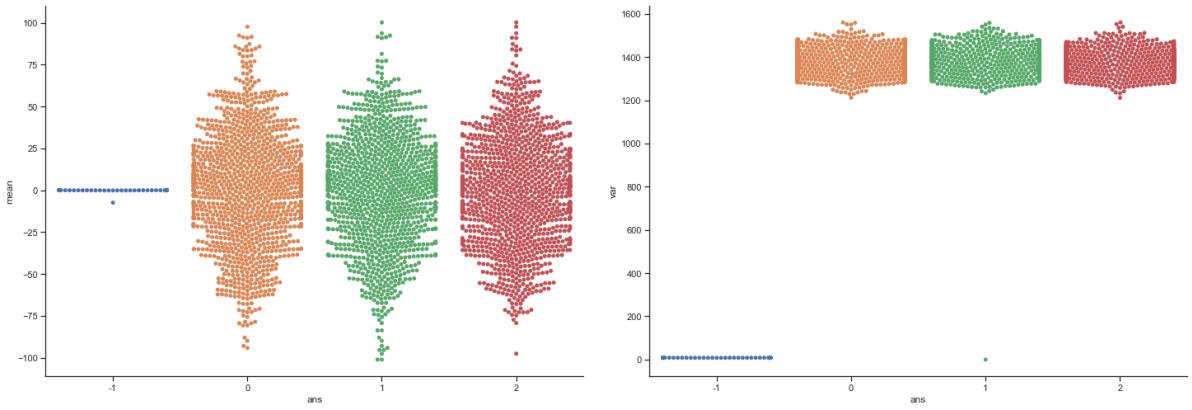


Class 2 只剩下四點在界線內，代表可以用 variance 很好的區隔 class 2 與另外兩組（測試發現上圖四點都是 mean 很大的點，因此用 mean 和 variance 可以 100%辨識 class2）不過若使用這兩個指標，也會有一些 anomaly 的資料被誤分到 class 2。

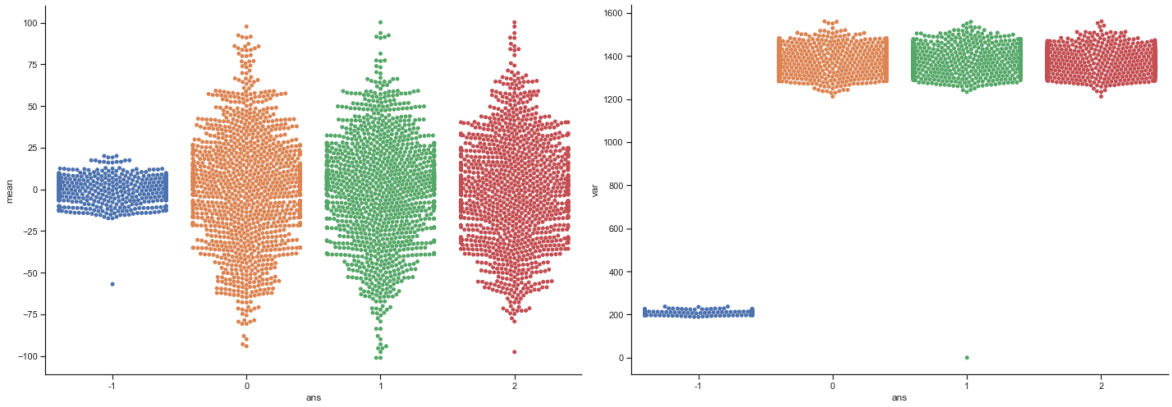
在找資料時有看到 github 上有人也有做 bearing fault detection，他有用到 fft 這個特性，因此我也嘗試畫畫看圖：



發現 anomaly 的資料起伏非常小，一樣用 mean(右) &variance(左)看看資料分布。



anomaly 組的資料經過 fft 轉換之後，比起其他三組有明顯較低的 variance。
但是檢查 anomalytestdata.npy 的部分發現也有這個特徵（label 為-1），懷疑測試資料和訓練資料來源不同。



由於時間關係，後面的分析來不及做。