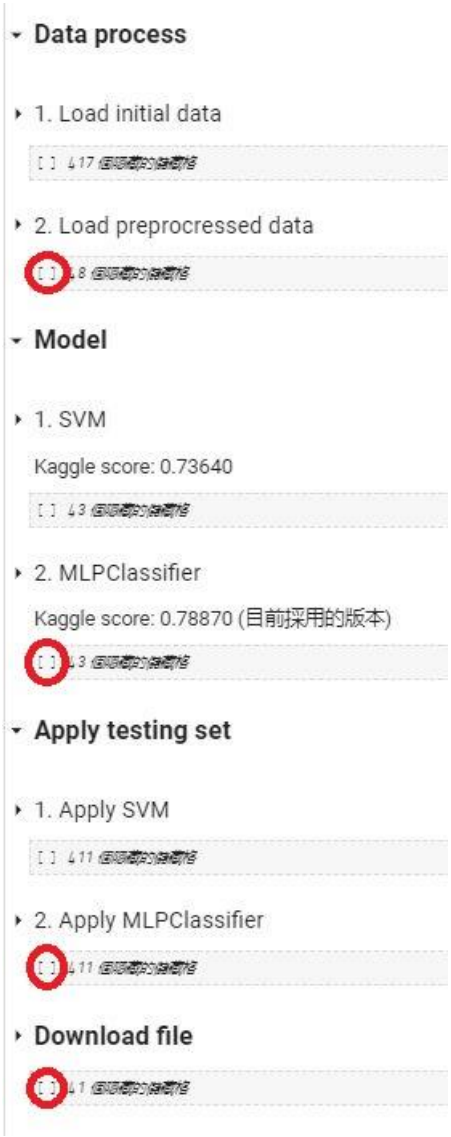
四資工三甲 B10715029 陳彥瑋

Data Science HW1 Report

1. 程式執行方式

程式是使用線上的 Google Colaboratory 撰寫的，網址是：https://colab.research.google.com/drive/1XaOE67rh7I7bzOp6UOFV46fcYQ_aqt6s?usp=sharing，剛進入後可先用 ctrl + [收合所有的區段以方便執行和閱讀。



一開始先打開 Data process 的部分，可以看到有 1. Load initial data 和 2. Load preprocessed data 兩個區段，1 的部分是把原始的檔案從頭執行讀取和預處理的部分，2 的部分則是讀取我已經先跑完預處理後儲存下來的檔案。基本上 1 跟 2 執行的結果都是相同的，執行其中一個即可（點擊區段開頭的方括號執行），只是說 2

的執行速度較快，所以建議執行 2 的部分。（1 的部分用來介紹 preprocess 的流程）

接下來在 Model 的部分有 1. SVM 和 2. MLPClassifier 兩個模型，目前在 kaggle 上 2 的分數比較高，所以執行 2 的部分即可（這邊大約要等 3 ~ 5 minutes）。

最後在 Apply testing set 的部分有 1. Apply SVM 和 2. Apply MLPClassifier 兩個部分，基本上是在前面 Model 的地方選什麼就選哪一個，所以這邊執行 2 的部分即可。

如果助教要下載執行完的結果的話，可以選擇左方檔案裡的 result.csv 或是執行 Download file 的部分。

2. 演算法簡介

在預處理的部分，首先我假設月份和會不會下雨有很大的關係（例如梅雨月份就容易下雨），所以先將 Attribute1（紀錄的日期）轉換成月份。

```
[ ]  #  Attribute1(date)  to  month
     #  Assumption:  raining  or  not  has  some  relation  with  month

     att1  =  train['Attribute1'].copy()
     for  i  in  range(len(att1)):
         att1[i]  =  att1[i].split('-')[1]
     train['Attribute1']  =  att1
```

再來，依據不同欄位的特性用不同的填補方式填補缺失值（像是風向資料我猜是因為沒有風所以才有缺失，所以用 NoWind 標籤來填）。

```
[ ]  #  Attribute  5,  6,  7,  9,  12,  13,  14,  15  fill  0
     #  Attribute  8,  10,  11  fill  NoWind  catalog

     fillzero  =  train[['Attribute5',  'Attribute6',  'Attribute7',  'Attribute9',  'Attribute12',  'Attribute13',  'Attribute14',  'Attribute15']].copy()
     fillzero  =  fillzero.fillna(0)

     fillNoWind  =  train[['Attribute8',  'Attribute10',  'Attribute11']].copy()
     fillNoWind  =  fillNoWind.fillna("NoWind")

     train[['Attribute5',  'Attribute6',  'Attribute7',  'Attribute9',  'Attribute12',  'Attribute13',  'Attribute14',  'Attribute15']]  =  fillzero
     train[['Attribute8',  'Attribute10',  'Attribute11']]  =  fillNoWind
```

```
[ ] # Attribute 3, 4, 16, 17, 18, 19, 20, 21, 22 fill average data

    ave = train[["Attribute3", "Attribute4", "Attribute16", "Attribute17", "Attribute18", "Attribute19", "Attribute20", "Attribute21"]].mean()
    most = train["Attribute22"].mode()[0]

    for i in range(train.shape[0]):
        currData = train.iloc[i]
        for col in ["Attribute3", "Attribute4", "Attribute16", "Attribute17", "Attribute18", "Attribute19", "Attribute20", "Attribute21", "Attribute22"]:
            if (pd.isnull(currData[col])):
                if (col == "Attribute22"):
                    train["Attribute22"][i] = most
                else:
                    train[col][i] = ave[col]
```

接下來把類別類型的資料用 one-hot encoding 的方式處理成數值。

```
[ ] # One-hot encoding

    train = pd.get_dummies(train, columns=["Attribute1", "Attribute2", "Attribute8", "Attribute10", "Attribute11"])
```

接者把 Attribute22（今天有沒有下雨）、Attribute23（明天會不會下雨，預測目標）轉換成 0 和 1 的數值。

```
[ ] # Attribute22, Attribute23 binarize

    for i in range(train.shape[0]):
        train["Attribute22"][i] = (train["Attribute22"][i] == "Yes")*1
        train["Attribute23"][i] = (train["Attribute23"][i] == "Yes")*1
```

然後，把 X, Y 分離。

```
[ ] # Split X, Y

    allKeys = train.keys().to_list()
    allKeys.remove("Attribute23")

    trainX = train[allKeys]
    trainY = train[["Attribute23"]]
```

最後，將 X 的資料做 z-score 的 normalization。

```
[ ] # z-score normalization

    scaler = preprocessing.MinMaxScaler()
    scaler.fit(trainX)
    trainX = scaler.transform(trainX)
```

做完預處理後，我把處理好的資料以 2:1 的比例分開成 training set 和 validation set，前者是用來訓練模型的，後者是用來輔助判斷模型在一般資料下的結果。

Split training set and validation set

```
[ ] from sklearn.model_selection import train_test_split

[ ] trainX, validX, trainY, validY = train_test_split(trainX, trainY, test_size=0.33, random_state=29)
```

在 model 的部分，我是使用 sklearn 的 MLPClassifier 這一個神經網路模型，其中，我設定它有 4 層 hidden layer，每一層的大小分別為 1000, 500, 200, 100 個神經元，每一層的輸出接的 activation function 是 relu。

在訓練這個模型時，我是使用 adam 的方法來解因為它的收斂速度最快而且是最常使用的 optimizer。此外，alpha 值代表的是 L2 regularization 的權重，越高代表模型越平滑，也就越不容易 overfitting，不過 training score 可能會下降。

最後，我設定 batch_size 是 100，也就是每一次取隨機 100 個樣本進行訓練。而 max_iter 的意思是最多只會跑這麼多次的訓練，不過這個模型在訓練過程中通常會因為 loss 的變化太小而提早結束訓練（雖然沒有加 early stopping）。

```
[ ] from sklearn.neural_network import MLPClassifier
    DNNmodel = MLPClassifier(hidden_layer_sizes=(1000,500,200,100), activation="relu", solver="adam", alpha=0.005, batch_size=100, max_iter=1000, random_state=29)
```

在預測 testing 上，預處理的部分差不多（除了不用補缺失值以及不用分離 X, Y），不過在 one-hot encoding 時我發現有些會在 training 時會生出來的 catalog 在 test 時不會有，原因是 training 時的資料比較多元所以在類別資料上會有 testing 沒有的類，舉例來說，在 testing 的 Attribute2（觀測站地區）就沒有 0 號觀測站的資料而 training 的有，所以 testing 就不會多出 Attribute_0 的 catalog。因此，為了要讓維度相同，所以要額外插入欄位。

```
[ ]  #  One-hot  encoding

     test  =  pd.get_dummies(test,  columns=["Attribute1",  "Attribute2",  "Attribute8",  "Attribute10",  "Attribute11"])

     #  Filling  lost  catalog
     test.insert(29,  "Attribute2_0",  0)
     test.insert(30,  "Attribute2_1",  0)
     test.insert(31,  "Attribute2_2",  0)
     test.insert(33,  "Attribute2_4",  0)
     test.insert(34,  "Attribute2_5",  0)
     test.insert(35,  "Attribute2_6",  0)
     test.insert(41,  "Attribute2_12",  0)
     test.insert(43,  "Attribute2_14",  0)
     test.insert(45,  "Attribute2_16",  0)
     test.insert(46,  "Attribute2_17",  0)
     test.insert(52,  "Attribute2_23",  0)
     test.insert(53,  "Attribute2_24",  0)
     test.insert(54,  "Attribute2_25",  0)
     test.insert(55,  "Attribute2_26",  0)
     test.insert(58,  "Attribute2_29",  0)
     test.insert(59,  "Attribute2_30",  0)
     test.insert(63,  "Attribute2_34",  0)
     test.insert(65,  "Attribute2_36",  0)
     test.insert(69,  "Attribute2_40",  0)
     test.insert(70,  "Attribute2_41",  0)
     test.insert(72,  "Attribute2_43",  0)
     test.insert(75,  "Attribute2_46",  0)
     test.insert(76,  "Attribute2_47",  0)
     test.insert(86,  "Attribute8_NoWind",  0)
     test.insert(103,  "Attribute10_NoWind",  0)
     test.insert(120,  "Attribute11_NoWind",  0)
```

3. 改進方向

```
[ ]  #  train  set  score
     print(DNNmodel.score(trainX,  trainY))

     #  valid  set  score
     print(DNNmodel.score(validX,  validY))
```

```
0.9951973454418442
0.8356965615030131
```

在 Model 的 MLPClassifier 裡可以看到 training score 大約是 0.99 而 validation score 大約只有 0.83 而已，可以很明顯地看到這個 model 已經 overfitting 了，目前有嘗試過 early stopping、增加 L2 權重、先做 PCA 降維等方法但是效果沒有很好，目前想到還可以改的方向是 dropout，但是 sklearn 的 MLPClassifier 不支援這個部分，所以或許要改用 keras 等專門的神經網路函式庫來搭建神經網路。