# 討論

1.資料集的前處理？

觀察到在成交量上數字大小明顯高出其他變項，為了讓變相間數字的差異不要造成某個變項比重特別重的錯誤，因此將每個數據進行標準化處理，降低因為數據大小差異過大造成的影響，採用 Z 分數的處理程序，消除離群值的影響，並將的一行資料刪除，因為第一行無法看出漲幅變化。

2.哪個分類模型的正確率最高？為什麼？當資料集 改變時，仍然能保有較高的正確率嗎？

總共採計三種分類模型，分別為邏輯回歸（Logistic Regression）、類神經網路（Neural Network）和決策樹（Decision Tree），三者的初始參數如附圖一至三，其中以邏輯回歸在訓練資料時的正確率最高，為 0.72，其次類神經網路為 0.61，決策樹最差，為 0.56。猜測是因為超參數的調整沒有符合這筆資料的型態，因而造成類神經網路與決策樹的表現差，決策樹的部分又因為本身不適合解決變項為連續數值型資料的問題，因此表現更差，而邏輯回歸本身就適合拿來解決特徵不多的分類問題，恰好符合邏輯回歸本身的優勢，而拿訓練好的模型驗證測試資料時，也是在邏輯回歸的表現最好，為 0.82，神經網路為 0.77，決策樹為 0.51。

其中我有個比較不明白的點，為何在邏輯回歸和類神經網路中，測試資料的準確率皆高過訓練資料，網路上查不太到相關資訊，應該也不是程式碼寫

錯，問題待解。

```
model_lr=LogisticRegression()
```

附圖一：邏輯回歸參數 ver1

```
model_nn = Sequential()
model_nn.add(Dense(64,input_dim = 5,activation = 'relu'))
model_nn.add(Dense(32, activation='relu'))
model_nn.add(Dense(output_dim = 1,activation='sigmoid'))
model_nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])
```

附圖二：類神經網路參數 ver 1

```
model_dt = DecisionTreeClassifier(
        criterion='gini',
        max_depth=2,
        max_leaf_nodes=2**4,
        random_state =2020)
```

附圖三：決策樹參數 ver1

3.如何改進分類模型?

邏輯回歸

正規化方法改用 L1，solver 用 liblinear，用以處理特徵少的問題，訓練正確率提升至 0.93，測試正確率也提升至 0.83，整體變化不大，而且有 overfitting 傾向，使用原先的預設參數即可

```
model_lr=LogisticRegression(penalty = 'l1',solver = 'liblinear')
```

附圖四：邏輯回歸參數 ver2

類神經網路

增加隱藏層層數（三層，神經元數分別為 128、64、32），並增加訓練代數，訓練正確率提升至 0.88，測試正確率提升至 0.84，為參數調整後正確率改動最大的組別，由於無法直接看出此正確率的變化是源自隱藏層的增加還

是代數的提升，因此分別又側試了兩個模型，一個為僅調整隱藏層，一個為僅調整代數，後者的變化較為明顯，訓練正確率為 0.90，測試正確率為 0.84，前者則是和最初的差不多，訓練正確率為 0.63，測試正確率為 0.78，因此任為過低的代數無法讓模型學習，因此模型訓練中先提升訓練代數讓模型確實學習到問題再調整其他可能參數（層數、神經元數等等）

```
model_nn = Sequential()
model_nn.add(Dense(128,input_dim = 5,activation = 'relu'))
model_nn.add(Dense(64, activation='relu'))
model_nn.add(Dense(32, activation='relu'))
model_nn.add(Dense(output_dim = 1,activation='sigmoid'))
model_nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

train_acc_all_list = model_nn.fit(train_x, train_y,       #輸入 與 輸出
          nb_epoch = 100,          #子代數
          batch_size = 10,        #批量大小 一次參考多少的數據
          verbose = 1 ,           #是否顯示訓練過程
          validation_data=(train_x, train_y)) #拿來預測的資料
```

附圖五：類神經網路參數 ver2（調整層數、代數）

```
model_nn = Sequential()
model_nn.add(Dense(64,input_dim = 5,activation = 'relu'))
model_nn.add(Dense(32, activation='relu'))
model_nn.add(Dense(output_dim = 1,activation='sigmoid'))
model_nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

train_acc_all_list = model_nn.fit(train_x, train_y,       #輸入 與 輸出
          nb_epoch = 100,          #子代數
          batch_size = 10,        #批量大小 一次參考多少的數據
          verbose = 1 ,           #是否顯示訓練過程
          validation_data=(train_x, train_y)) #拿來預測的資料
```

附圖六：類神經網路參數 ver3（調整代數）

```
model_nn = Sequential()
model_nn.add(Dense(128,input_dim = 5,activation = 'relu'))
model_nn.add(Dense(64, activation='relu'))
model_nn.add(Dense(32, activation='relu'))
model_nn.add(Dense(output_dim = 1,activation='sigmoid'))
model_nn.compile(loss='mean_squared_error', optimizer='adam', metrics=['accuracy'])

train_acc_all_list = model_nn.fit(train_x, train_y,       #輸入 與 輸出
          nb_epoch = 10,          #子代數
          batch_size = 10,        #批量大小 一次參考多少的數據
          verbose = 1 ,           #是否顯示訓練過程
          validation_data=(train_x, train_y)) #拿來預測的資料
```

附圖七：類神經網路參數 ver4（調整層數）

決策樹

Splitter 調整為 best，最大深度為 10，最多節點數為 2 的 8 次方，將整顆數條大一點，也更適合處理特徵少的問題，雖然訓練與測試正確率都有提升，訓練正確率為 0.68，測試正確率為 0.57，但是還是可以看出決策樹在處理這個問題上的表現仍舊不理想。

```
model_dt = DecisionTreeClassifier(
        criterion='gini',
        splitter = 'best',
        max_depth=10,
        max_leaf_nodes=2**8,
        random_state =2020)
```

附圖八：決策樹參數 ver2