Kaggle Competition Final Report

Team 22: P76074347 王晉鋒, E54036219 張富嘉

Competition

Two Sigma: Using News to Predict Stock Movements

Dataset

本次競賽基於以下兩種資料：

1. Market data（2007 年至今）包含各家公司的金融信息，如開盤價，收盤價，交易量，計算回報等。共 16 欄。
2. News data（2007 年至今）包含有關資產的新聞文章/警報信息，如文章詳情，情緒和其他評論。共 35 欄。

Evaluation (Objective)

預測值為每日後 10 天的漲跌信心指數 $\hat{y}_{ti} \in [-1,1]$，每天(t)會根據每檔資產(i)計算出x_t值，算法如下：

$$x_t = \sum_i \hat{y}_{ti}\, r_{ti} u_{ti}$$

其中r_{ti}是後 10 天的市場調整後收益率，u_{ti}記錄當天該筆資產是否列入計算。最後的分數將以以下算法得出：

$$\text{score} = \frac{\bar{x_t}}{\sigma(x_t)}$$

由上述算分方式即可得知，本場競賽的目的即是在預測資產的夏普比率(sharpe ratio)。

Competition result

目前第一階段的評分結果顯示，排行榜上第一名分數約為 1.5，排名前 50%的分數為大約 0.64，我們上傳得到最高的成績為 0.6703 排名前 20%。

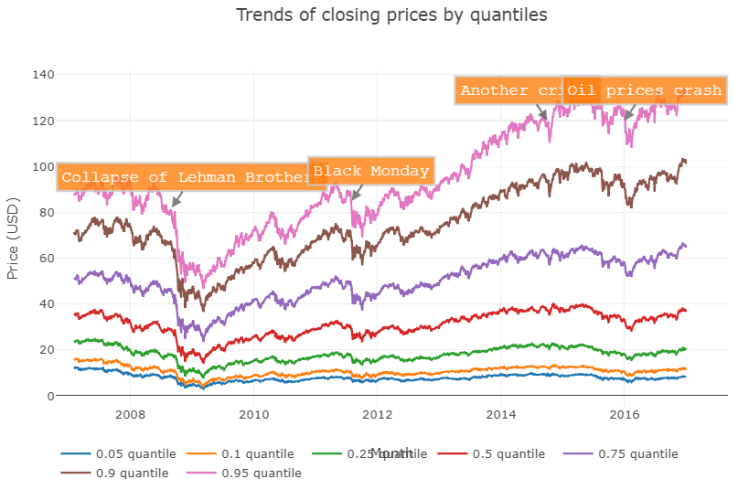
Preliminary:

在開始實作前，我們做過以下假設及觀察：

1. Guess/Investigate important features

新聞資料方面，資料集提供的 sentiment*等資訊，代表每則新聞對於所提及資產的正反面評價，我們認為是良好的指標，另外我們也認為 volumeCounts*等聲量資料也可能會造成資產價格的波動性增加。市場資料方面則提供了前 1 天及前 10 天的原始報酬和市場殘餘報酬(扣除整體市場波動後的報酬)，相對於未處理過的原始價格時間序列資料，這幾項資料將可大幅增加預測夏普比率的準確性和方便性。

2. Some serious events affected the stock price



由上圖可以看見，平均股價在抹些特定時段會因為重大事件而有較大的起伏，而這些資料往往會嚴重影響預測結果，因為這些非常規的資料分布很有可能會破壞模型訓練時所學習到的規則。所以我們決定只選擇 2010 年以後的資料，捨棄掉 2008 年到 2009 年金融海嘯的特例資料，以期較為符合 validation data 的分布狀況。

Model

選擇使用 LightGBM，一種 boosting 的隨機森林算法，可以做分類及預測信心指數。會使用此 model 是因為在 Kaggle 平台諸多的討論中，我們發現此 model 可達到最好的效果，實作上也比較容易，若是使用 Neural Network based 的算法，一方面是網路層數，learning rate 及 batch size 的調整都需耗費大量時間使其收斂到好的成果，另外一方面是能使用的特徵會受到限制，基於 Neural Network based 算法只能使用連續數值而不能使用離散類別或是非數值資料去做分類或預測，即使對上述資料轉換成連續數值，資料的可用資訊也會降低或是無法使用。

Data Preprocessing

Remove outliers

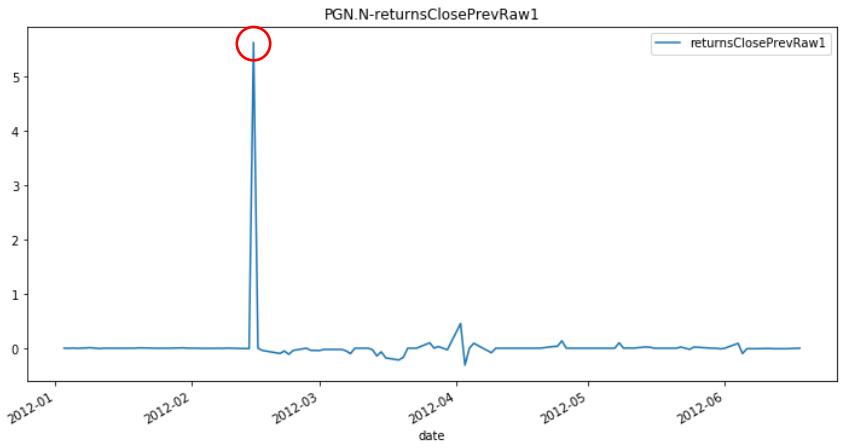
Training data 會有錯誤的資料混雜在其中，這會造成嚴重的 outlier，且不單單是移除掉這筆資料就行，因為有些特徵彼此之間有時間相依性，這裡所做的處置是去填補正確的數值 (實際去搜尋的)，再去對前後十天的報酬特徵去計算相對正確的數值，如此降低離譜的異常值所造成這段時間其他特徵計算上的錯誤。

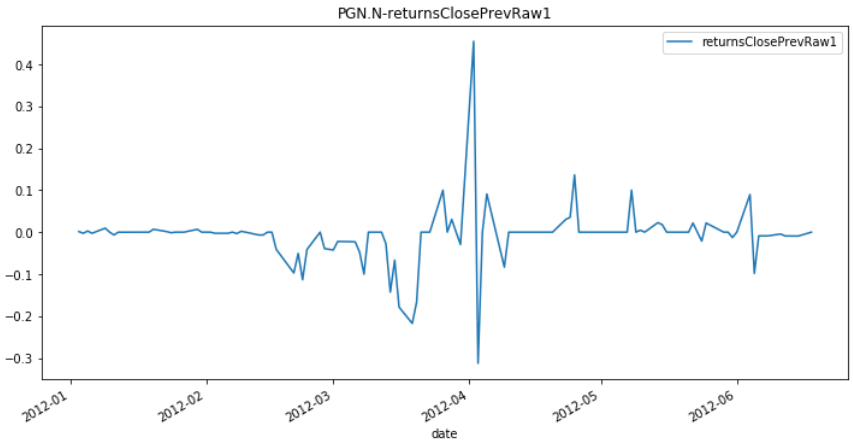
下列為其中兩個修正異常值的例子：

1. 在 2016/07/06，有檔資產的 close 值有誤，原始值不是 123.45 就是 123.47，比對 open 值會發現不合理，必須做修正。

time	assetCode	assetName	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1
2016-07-06 22:00:00+00:00	BBBY.O	Bed Bath & Beyond Inc	50303.0	123.45	30.68	1.921202	-0.295522
2016-07-06 22:00:00+00:00	DISH.O	DISH Network Corp	87466.0	123.47	63.29	1.430033	0.206903
2016-07-06 22:00:00+00:00	FLEX.O	Flex Ltd	175451.0	123.45	16.44	9.587479	0.405128
2016-07-06 22:00:00+00:00	MAT.O	Mattel Inc	56994.0	123.45	21.54	2.904175	-0.315321
2016-07-06 22:00:00+00:00	NDAQ.O	Nasdaq Inc	113350.0	123.47	71.51	0.911596	0.122254
2016-07-06 22:00:00+00:00	PCAR.O	Paccar Inc	63374.0	123.45	66.41	1.445523	0.285521
2016-07-06 22:00:00+00:00	PZZA.O	Papa John's International Inc	25050.0	123.45	71.89	0.817580	0.056429
2016-07-06 22:00:00+00:00	SHLD.O	Sears Holdings Corp	80940.0	123.47	8.84	8.512327	-0.351431
2016-07-06 22:00:00+00:00	ZNGA.O	Zynga Inc	418847.0	123.47	3.65	45.592453	0.403846

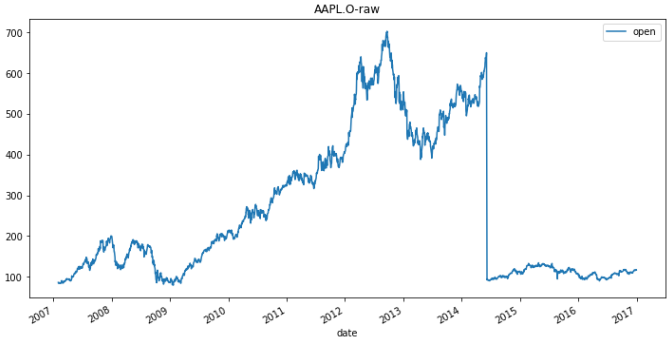
2. 如下圖所示，在 2012/02/15，PGN.N 的 returnsClosePrevRaw1 出現異常值，必須藉由過去的 open 和 close 來從新計算。



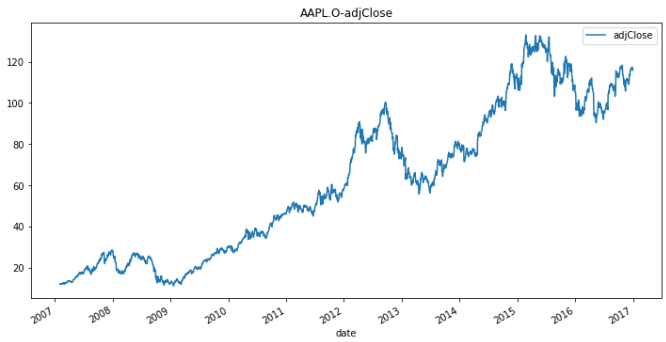


上圖為 PGN.N 經過修正之後的 returnsClosePrevRaw1，可以看到已回復正常，無異常的情形。

另外除了資料本身錯誤所造成的異常之外，我們也發現當資產發生股票分割(Stock Split)的時候，open 和 close 值會有極大的向下跳動出現，例如當分割數為 2 時，股價也會直接變成一半。由於許多資產皆有過股票分割，而且我們有使用 close 來產生其他重要的 features，所以必須對其進行處理。下圖為蘋果在 2014 年進行股票分割時股價從 6 百多下降到不到 1 百的情形，股票分割的訊息可以在 News data 中的 headline 找到，可以看到經過處理後價格還原成正常：




```
['Apple may be considering stock split: Bernstein',
 'Apple expands buybacks by $30 bln, OKs 7-for-1 stock split',
 'UPDATE 1-Apple expands buybacks by $30 bln, OKs 7-for-1 stock split',
 'UPDATE 2-Apple expands buybacks by $30 bln, OKs 7-for-1 stock split',
 'Reuters Insider - Apple stock jumps on results, stock split']
```



後續我們陸續參考了許多修正異常值的 kernel，並逐一將這些異常值還原。下表是在做處理之前，Market data 的資料大致分布情況：

	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1	returnsClosePrevMktres1
count	4.072956e+06	4072956.000	4072956.000	4072956.000	4072956.000	4056976.000
mean	2.665312e+06	39.712	39.712	0.001	0.010	0.000
std	7.687606e+06	42.288	42.611	0.037	7.084	0.033
min	0.000000e+00	0.070	0.010	-0.978	-1.000	-1.236
25%	4.657968e+05	17.250	17.250	-0.011	-0.011	-0.009
50%	9.821000e+05	30.300	30.290	0.000	0.000	-0.000
75%	2.403165e+06	49.860	49.850	0.012	0.012	0.008
max	1.226791e+09	1578.130	9998.990	45.592	9209.000	45.122

經過一系列去除、還原等過程之後，可以看到下表的數值已經正常許多：

	volume	close	open	returnsClosePrevRaw1	returnsOpenPrevRaw1	returnsClosePrevMktres1
count	3.340086e+06	3340086.000	3340086.000	3340086.000	3340086.000	3328126.000
mean	2.639928e+06	40.326	40.316	0.001	0.001	0.000
std	7.816397e+06	44.588	44.583	0.026	0.026	0.021
min	0.000000e+00	0.070	0.080	-0.845	-0.862	-0.836
25%	4.584260e+05	16.940	16.940	-0.010	-0.010	-0.008
50%	9.758070e+05	30.300	30.290	0.001	0.001	-0.000
75%	2.392388e+06	50.550	50.530	0.011	0.011	0.008
max	1.226791e+09	1578.130	1584.440	7.178	7.178	6.981

可以看到不論是 close、returnsOpenPrevRaw1 都有很明顯的修正，且符合常理。在結果表現上，相同的 case 中，有經過前處理的分數是 0.6436，比未經處理的分數(0.6427)增加了約 0.001 分。

Filling missing values

最後須將 Market data 和 News data 依照時間和 assetCode 做合併，但在合併時，由於某天某間公司可能沒有任何相關的新聞報導參考(如下表所示)，所以 News data 會產生許多遺漏值，針對遺漏的 News data，我們想透過用 Market data 去訓練 kNN，再針對 News data 的每一項特徵去做預測，試圖增加資料集的大小且避免遺漏值影響模型分類的依據。

OpenNextMktres10	universe	date	sourceId_mean	urgency_mean	takeSequence_mean	provider_mean	bodySize_mean	compa
33	1.0	2009-01-02	NaN	NaN	NaN	NaN	NaN	NaN
76	0.0	2009-01-02	416.5	2.5	1.125	1.25	1840.0	1.5
82	1.0	2009-01-02	NaN	NaN	NaN	NaN	NaN	NaN

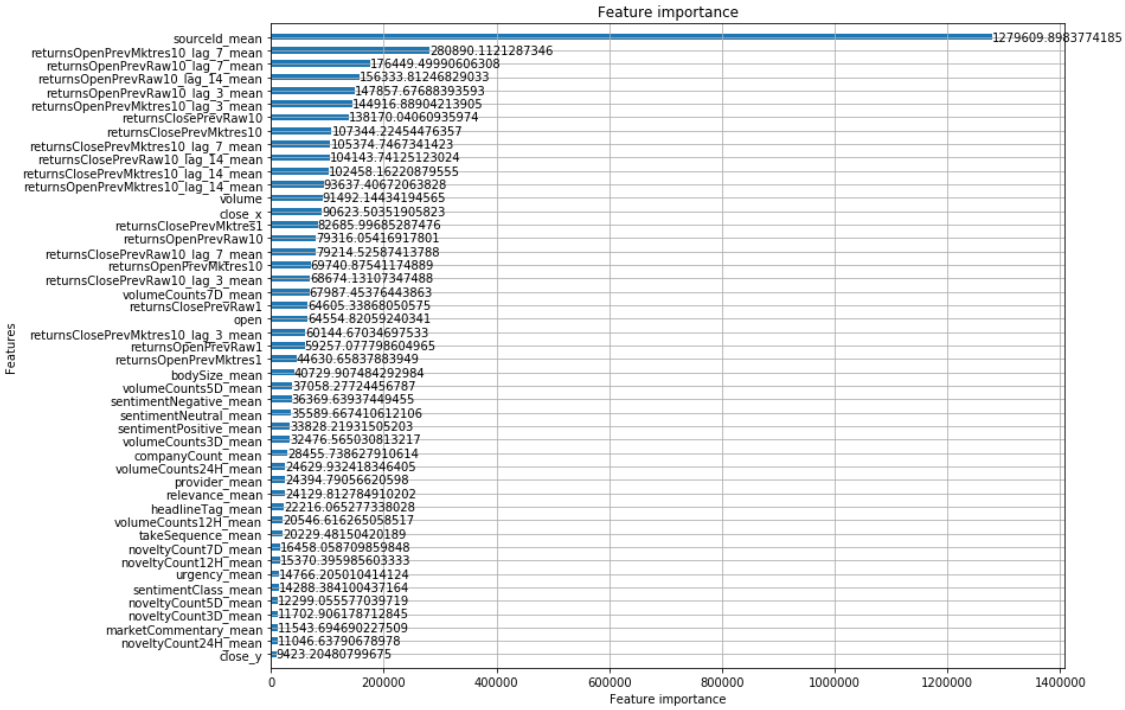
在其他條件不變下，最後的結果為 0.617 (做遺漏值填補，80%真實資料 20%填補資料)，相比之下表現較差，在此可以解釋說 kNN 所填的值並不妥當，甚至當填補的資料比例變大，其結果會更差。因此往後的 model 我們不採用 kNN 填補的方式，而改為採取填-1 或 0。

Feature Selection

News data or not news data?

我們做了幾個測試，討論是否需要使用 News data：

1. 只用 Market data (11 features) :　result 0.6369
2. Market + 部分 News (17 features):　result 0.6436
   (這 邊 的　news　是 取　'companyCount', 'sentimentClass', 'noveltyCount12H', 'noveltyCount7D', 'volumeCounts3D', 'volumeCounts7D'，是離散值且　feature importance　相對較高的幾個)



3. Market + News (39 features):　result 0.6184
4. Market + generate features (相對漲跌，開收盤平均，總收盤價，14 features): result: 0.6451

這裡我們得出以下幾點結論：

➔ Feature 越多不代表預測結果越好，但相對的也不能太少

➔ News data 在這邊可能降低整體的預測結果，根據同樣參加這次競賽的人之間的諸多討論，新聞資料往往不會立即性或是相對對股價造成嚴重的影響，且預測結果也顯示出 News data 並非必要當作訓練的特徵 (可能有幫助但很難，弄得好可能需要時間上的分析及文字向量做輔助，但大部分的情況都是降低 model 的表現)

➔ 著重在 market data 的 feature 生成或許是個不錯的方式增加訓練的資訊，之後的幾項測試將不再使用新聞資料而是將重心放在 market 的操作，並且嘗試使用原資料以外的生成資料去增進模型的學習能力。

Feature Generation

以下是針對 Market data 生成的一些新的特徵：

1. Create lag

由於我們是使用 LightGBM 來做時間序列的預測，而 LightGBM 是屬於 tree based 的訓練模型，一般而言若要讓 tree based 的模型做時間序列的預測，需要將歷史資料或時間列入特徵中，才能得到較佳的預測結果。在這次競賽中我們使用 create lag 的方式取得過去幾天的資料作為特徵，做法為分別將每個 assetCode 的 returnsClosePrevMktres10 和 returnsClosePrevRaw10 這兩個特徵取出，然後用 rolling 的方式算出 lag 分別為 3, 7, 14 步的平均值、最大值和最小值，最後會生成 18 (3*3*2) 個特徵，分別是：
returnsClosePrevMktres10_lag_3_mean 、 returnsClosePrevMktres10_lag_3_max 、 returnsClosePrevMktres10_lag_3_min 、 returnsClosePrevMktres10_lag_7_mean 、 returnsClosePrevMktres10_lag_7_max 、 returnsClosePrevMktres10_lag_7_min 、 returnsClosePrevMktres10_lag_14_mean、returnsClosePrevMktres10_lag_14_max、 returnsClosePrevMktres10_lag_14_min。經過測試，增加 lag 特徵能夠增加約 0.03 分的排行分數。

2. 指數移動平均 EMA

指數移動平均是以指數式遞減加權的移動平均，各數值的加權影響力隨時間而指數式遞減，越近期的數據加權影響力越重。其算法如下：

$$S_t = \alpha \times Y_t + (1 - \alpha) \times S_{t-1},$$

$$\alpha = \frac{2}{N + 1}$$

其中S_t為時間 t 的 EMA 值，Y_t為時間 t 的實際數值，α為加權值。我們為了計算下面會提到的 MACD 指標，所以計算了 lag 為 12 步和 26 步的 EMA，分別生成 close_26EMA 和 close_12EMA 兩個特徵。

3. MACD

MACD 的基本原理是運用兩條不同速度的指數平滑移動平均線來計算兩者之間的差離狀態(DIF)，然後再對 DIF 進行平滑移動平均即為 MACD，常用來判斷中短期的買賣訊號。在這邊為了方便計算，我們將先前算出的 close_26EMA 和 close_12EMA 相減當作 MACD 的值。即：MACD = close_26EMA - close_12EMA。

4. 相對強弱指標 RSI

相對強弱指標是一個比較價格升降運動以表達價格強度的技術分析工具。其算法如下：

$$\mathrm{RS} = \mathrm{UP}/\mathrm{DN},$$

$$\mathrm{RSI} = 100 - \frac{100}{1 + RS}$$

其中 UP 為期間內之絕對漲幅，DN 為期間內之絕對跌幅。我們分別計算了 14 天和 6 天的 RSI 值，生成 rsi_14 和 rsi_6 兩個特徵。

5. 布林帶 BBands

布林帶分為上軌、中軌和下軌，這邊我們將中軌設為 7 天的移動平均('MA_7MA')，則上下軌等於中軌$\pm \mathrm{K} * 7$ 天的移動標準差。即 MA_7MA_BB_high/low = MA_7MA $\pm$ 2* MA_7MA_std。

6. 歷史資料

我們將當天對應的公司名稱，取前幾天的資料(含 lag features)合併作為當天的特徵值。企圖增加時間的資訊在訓練資料集上。

Modeling

Parameters Tuning

對基於決策樹的模型，我們採用的參數調整方法可以大略歸類為以下幾個步驟：

1. 先選擇較高的 learning_rate(例如 0.1)
2. 調整基本參數
3. 調整參數以降低 overfitting
4. 調整正則化參數
5. 降低 learning_rate，提高準確率

下面將以此次競賽的其中一次參數調整為例(此次模型使用 LGBMClassifier，objective="binary"，metric="binary_logloss"，training data 為原本的 market data 加上 lag features)，說明並比較結果：

Step1. learning_rate

首先將 learning_rate 設為一個較大的值，此處設為 0.2，max_depth 設為 7，num_leaves 設為 80，可以得到 best iteration=503，loss=0.680922。接下來我們帶入 learning_rate=0.2，num_boost_round=503，接續調整其他參數。

Step2. max_depth 和 num_leaves

max_depth 是樹的深度，深度太大會造成 overfit，num_leaves 是做為調整樹的複雜度的參數，和 max_depth 大致的換算關係為 num_leaves < $2^{\mathrm{max\_depth}}$。在這邊我們

使用 GridSearchCV 工具來調整參數，得到初步的最佳解是在 max_depth=7，num_leaves=110 的時候，loss=0.683284，接下來進行參數微調。

```
params_test2={
    'max_depth': [6,7,8],
    'num_leaves': [80,95,110,125,140]
}
```

```
({'max_depth': 8, 'num_leaves': 125},
 -0.6819858501351308,
```

首先我們將參數範圍縮減到如上圖所示之範圍，總共比較了 15 個參數組合，得到最佳解是在 max_depth=8，num_leaves=125 的情況下，loss=0.6819858。

Step3. min_child_samples 和 min_child_weight

接下來處理 overfitting 的問題，將 min_child_samples 設得較大可以避免生成一個過深的樹。這邊一樣使用 GridSearchCV 作為調參的工具，min_child_samples 的測試範圍設在 18~22，min_child_weight 的測試範圍設在 0.001, 0.002。最後結果 min_child_samples 的最佳值為 20，min_child_weight 則為 0.001，與預設值相同。

Step4. feature_fraction 和 bagging_fraction

設置這兩個參數同樣也是為了降低 overfitting 的，同時也可以加快訓練速度，使用 GridSearchCV 將 feature_fraction 的測試範圍設在 0.5~0.9，bagging_fraction 的測試範圍設在 0.6~1.0。得到最後結果 feature_fraction 的最佳值為 0.8，bagging_fraction 的最佳值為 1.0。

Step5. reg_alpha 和 reg_lambda

這兩個參數分別對應到 l1 正則化和 l2 正則化，使用 GridSearchCV 將 reg_alpha 和 reg_lambda 的測試範圍設定為[0, 0.001, 0.01, 0.03, 0.08, 0.3, 0.5]。最後得到 reg_alpha 的最佳值為 0.5，reg_lambda 的最佳值為 0.5。

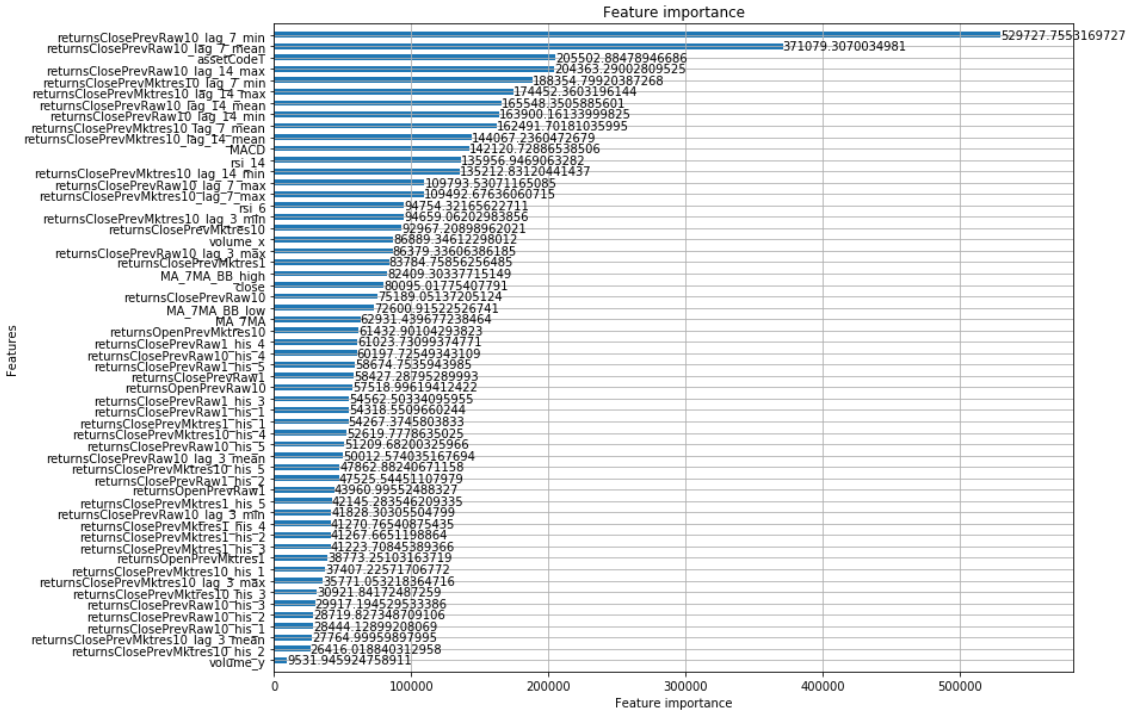
Step6. 降低 learning_rate

在找完以上參數之後，我們將 learning_rate 調整到 0.01，並使用上述參數帶入訓練模型，最終得到 log_loss 為 0.678677。

以上是這次競賽中為 lightGBM 調整參數的方法，過程中也有嘗試過其他的調參工具，例如 gp_minimize 等，方法皆和上述步驟一樣。

Experiment

下圖為 Market data 和新生成特徵的 feature importance 圖表，本次競賽中我們根據此圖測試了以下幾種案例：



1. Create lag：從上圖可以看到 lag feature 對模型的重要性相對都比較重要，所以我們對 returnsClosePrevMktres10 和 returnsClosePrevRaw10 生成總共 18 個 features，再跟 market data 結合，共 29 個 features。

2. 對(1)的模型新增 MACD、 rsi_14、rsi_6 這三個 features。

3. 這邊想嘗試新增歷史資料的要素，因為記憶體大小有限制，無法複製太多 features，這次選擇 7 個 features 當作基底，合併前 5 天的資料，共 35 個 features 一併去做訓練。被選來作為歷史資料的 features 有：MACD、rsi_14、一日的 raw、十日的 close 跟 open 的 Mktres，以及這兩項生成的 lag=14 日 mean。

4. 不使用(3)中 lag 的特徵，對其他 5 個特徵取出前 9 天的資料，共 45 個 features 做訓練。

測試結果顯示，在相同參數之下，在上傳的分數上分別為 0.6403, 0.6533, 0.6703, 0.6138，也就是說，取適當數量的 feature 去做歷史資料的擴增，是有符合我們的假設的增進 model 的分數。可見得時間訊息對於用 lightGBM 訓練時間序列資料是重要且必要的。

```
    df_code['his_clo_mktres_2'] = df_code['returnsClosePrevMktres10_lag_14_mean'].shift(3)
    df_code['his_clo_raw_2'] = df_code['returnsClosePrevRaw10_lag_14_mean'].shift(3)
    df_code['his_clo_mktres_3'] = df_code['returnsClosePrevMktres10_lag_14_mean'].shift(7)
    df_code['his_clo_raw_3'] = df_code['returnsClosePrevRaw10_lag_14_mean'].shift(7)
    df_code['his_clo_mktres_4'] = df_code['returnsClosePrevMktres10_lag_14_mean'].shift(14)
    df_code['his_clo_raw_4'] = df_code['returnsClosePrevRaw10_lag_14_mean'].shift(14)

#     df_code['his_clo_mktres_1_min'] = df_code['returnsClosePrevMktres10_lag_14_min'].shift(1)
#     df_code['his_clo_raw_1_min'] = df_code['returnsClosePrevRaw10_lag_14_min'].shift(1)
    df_code['his_clo_mktres_2_min'] = df_code['returnsClosePrevMktres10_lag_14_min'].shift(3)
    df_code['his_clo_raw_2_min'] = df_code['returnsClosePrevRaw10_lag_14_min'].shift(3)
    df_code['his_clo_mktres_3_min'] = df_code['returnsClosePrevMktres10_lag_14_min'].shift(7)
    df_code['his_clo_raw_3_min'] = df_code['returnsClosePrevRaw10_lag_14_min'].shift(7)
    df_code['his_clo_mktres_4_min'] = df_code['returnsClosePrevMktres10_lag_14_min'].shift(14)
    df_code['his_clo_raw_4_min'] = df_code['returnsClosePrevRaw10_lag_14_min'].shift(14)
```

```
for col in return_features:
    for window in n_lag:
        rolled = df_code[col].shift(shift_size).rolling(window=window)
        lag_mean = rolled.mean()
        lag_max = rolled.max()
        lag_min = rolled.min()
        lag_std = rolled.std()
        df_code['%s_lag_%s_mean'%(col,window)] = lag_mean
        df_code['%s_lag_%s_max'%(col,window)] = lag_max
        df_code['%s_lag_%s_min'%(col,window)] = lag_min
```

(version 4/10)	0.67031	

Conclusion:

這次競賽，構想主要圍繞於如何透過適當的分析資料，去選取這次比賽最可被使用的特徵集，雖然測試的模型種類只有一種，但資料競賽最重要的我認為不在模型，而是操作這些資料的技術。這次我們花了很多時間去討論，參考網路上專家們的一些意見，然後去跑模型，縱使結果不是最好的，但在比賽過程中學習到很多東西。