Stock_ML

April 24, 2020

[1]:
```
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.preprocessing import StandardScaler, LabelEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.svm import SVC
from sklearn import svm
from sklearn.neural_network import MLPClassifier
from sklearn.metrics import confusion_matrix, classification_report
from sklearn.model_selection import GridSearchCV
```

0.1 Data Wrangling

[2]:
```
stocks = pd.read_csv('data.csv', index_col=0)
```

[3]:
```
del stocks['date']
stocks.head()
```

[3]:

	high	low	open	close	volume	adj_close	\
0	3.937500	3.781250	3.812500	3.925781	53843200.0	2.518801	
1	3.945312	3.875000	3.937500	3.875977	36739200.0	2.486846	
2	3.890625	3.812500	3.851562	3.859375	52459200.0	2.476195	
3	3.812500	3.734375	3.804688	3.742188	80836800.0	2.401006	
4	3.765625	3.734375	3.757812	3.742188	58376000.0	2.401006	

	sp_percent_change	percent_change	SMA_41_period_SMA	SMM_9_period_SMM	\
0	0.003021	0.019270	3.888529	3.925781	
1	0.001338	-0.012686	3.889315	3.925781	
2	0.002674	-0.004283	3.888553	3.925781	
3	0.001000	-0.030365	3.883217	3.925781	
4	0.003996	0.000000	3.876739	3.875977	

	…	PPO_HISTO	VW_MACD_MACD	VW_MACD_SIGNAL	EV_MACD_MACD	EV_MACD_SIGNAL	\
0	…	0.326573	0.027446	0.019021	0.084081	0.085987	
1	…	0.165085	0.023622	0.019941	0.082570	0.085304	
2	…	0.025102	0.017531	0.019459	0.079963	0.084236	

```
3  …  -0.259541     -0.001684        0.015230      0.071205        0.081629
4  …  -0.427895     -0.011353        0.009914      0.065696        0.078443

    MOM_MOM   ROC_ROC    RSI_RSI  IFT_RSI_IFT_RSI  short_result
0  0.160156  5.345912  52.326346         0.128566             0
1  0.063477  2.293814  46.603432         0.042299             6
2  0.023438  2.489627  44.722182         0.004378             6
3 -0.085938 -1.844262  33.656509         2.017359             9
4 -0.179688 -2.443992  33.656509       678.794701            10

[5 rows x 40 columns]
```

[4]:
```
stocks.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 261856 entries, 0 to 261855
Data columns (total 40 columns):
 #   Column                  Non-Null Count   Dtype
---  ------                  --------------   -----
 0   high                    261856 non-null  float64
 1   low                     261856 non-null  float64
 2   open                    261856 non-null  float64
 3   close                   261856 non-null  float64
 4   volume                  261856 non-null  float64
 5   adj_close               261856 non-null  float64
 6   sp_percent_change       261856 non-null  float64
 7   percent_change          261856 non-null  float64
 8   SMA_41_period_SMA       261856 non-null  float64
 9   SMM_9_period_SMM        261856 non-null  float64
 10  SSMA_9_period_SSMA      261856 non-null  float64
 11  EMA_9_period_EMA        261856 non-null  float64
 12  DEMA_9_period_DEMA      261856 non-null  float64
 13  TEMA_9_period_TEMA      261856 non-null  float64
 14  TRIMA_18_period_TRIMA   261856 non-null  float64
 15  TRIX_20_period_TRIX     261856 non-null  float64
 16  VAMA_8_period_VAMA      261856 non-null  float64
 17  ER_10_period_ER         261856 non-null  float64
 18  KAMA_20_period_KAMA.    261856 non-null  float64
 19  ZLEMA_26_period_ZLEMA   261856 non-null  float64
 20  WMA_9_period_WMA.       261856 non-null  float64
 21  deltawma                261856 non-null  float64
 22  HMA_16_period_HMA.      261856 non-null  float64
 23  EVWMA_20_period_EVWMA.  261856 non-null  float64
 24  VWAP_VWAP.              261856 non-null  float64
 25  SMMA_SMMA               261856 non-null  float64
 26  MACD_MACD               261856 non-null  float64
 27  MACD_SIGNAL             261856 non-null  float64
```

```
 28  PPO_PPO                 261856 non-null  float64
 29  PPO_SIGNAL              261856 non-null  float64
 30  PPO_HISTO               261856 non-null  float64
 31  VW_MACD_MACD            261856 non-null  float64
 32  VW_MACD_SIGNAL          261856 non-null  float64
 33  EV_MACD_MACD            261856 non-null  float64
 34  EV_MACD_SIGNAL          261856 non-null  float64
 35  MOM_MOM                 261856 non-null  float64
 36  ROC_ROC                 261856 non-null  float64
 37  RSI_RSI                 261856 non-null  float64
 38  IFT_RSI_IFT_RSI         261856 non-null  float64
 39  short_result            261856 non-null  int64
dtypes: float64(39), int64(1)
memory usage: 81.9 MB
```

[5]:
```
stocks.isnull().sum()
```

[5]:
```
high                    0
low                     0
open                    0
close                   0
volume                  0
adj_close               0
sp_percent_change       0
percent_change          0
SMA_41_period_SMA       0
SMM_9_period_SMM        0
SSMA_9_period_SSMA      0
EMA_9_period_EMA        0
DEMA_9_period_DEMA      0
TEMA_9_period_TEMA      0
TRIMA_18_period_TRIMA   0
TRIX_20_period_TRIX     0
VAMA_8_period_VAMA      0
ER_10_period_ER         0
KAMA_20_period_KAMA.    0
ZLEMA_26_period_ZLEMA   0
WMA_9_period_WMA.       0
deltawma                0
HMA_16_period_HMA.      0
EVWMA_20_period_EVWMA.  0
VWAP_VWAP.              0
SMMA_SMMA               0
MACD_MACD               0
MACD_SIGNAL             0
PPO_PPO                 0
PPO_SIGNAL              0
```

```
PPO_HISTO                0
VW_MACD_MACD             0
VW_MACD_SIGNAL           0
EV_MACD_MACD             0
EV_MACD_SIGNAL           0
MOM_MOM                  0
ROC_ROC                  0
RSI_RSI                  0
IFT_RSI_IFT_RSI          0
short_result             0
dtype: int64
```

0.1.1 Prepping data

[6]:
```
bins=[-300, -11, -5, 5, 11, 300]
group_names = ['strong sell', 'sell', 'hold', 'buy', 'strong buy']
stocks['short_result'] = pd.cut(stocks['short_result'], bins=bins,␣
 ↪labels=group_names)
stocks['short_result'].unique()
```

[6]:
```
[hold, buy, strong buy, sell, strong sell]
Categories (5, object): [strong sell < sell < hold < buy < strong buy]
```

[7]:
```
label_result = LabelEncoder()
```

[8]:
```
stocks['short_result'] = label_result.fit_transform(stocks['short_result'])
```

[9]:
```
stocks.head()
```

[9]:

	high	low	open	close	volume	adj_close	\
0	3.937500	3.781250	3.812500	3.925781	53843200.0	2.518801	
1	3.945312	3.875000	3.937500	3.875977	36739200.0	2.486846	
2	3.890625	3.812500	3.851562	3.859375	52459200.0	2.476195	
3	3.812500	3.734375	3.804688	3.742188	80836800.0	2.401006	
4	3.765625	3.734375	3.757812	3.742188	58376000.0	2.401006	

	sp_percent_change	percent_change	SMA_41_period_SMA	SMM_9_period_SMM	\
0	0.003021	0.019270	3.888529	3.925781	
1	0.001338	-0.012686	3.889315	3.925781	
2	0.002674	-0.004283	3.888553	3.925781	
3	0.001000	-0.030365	3.883217	3.925781	
4	0.003996	0.000000	3.876739	3.875977	

	…	PPO_HISTO	VW_MACD_MACD	VW_MACD_SIGNAL	EV_MACD_MACD	EV_MACD_SIGNAL	\
0	…	0.326573	0.027446	0.019021	0.084081	0.085987	
1	…	0.165085	0.023622	0.019941	0.082570	0.085304	
2	…	0.025102	0.017531	0.019459	0.079963	0.084236	

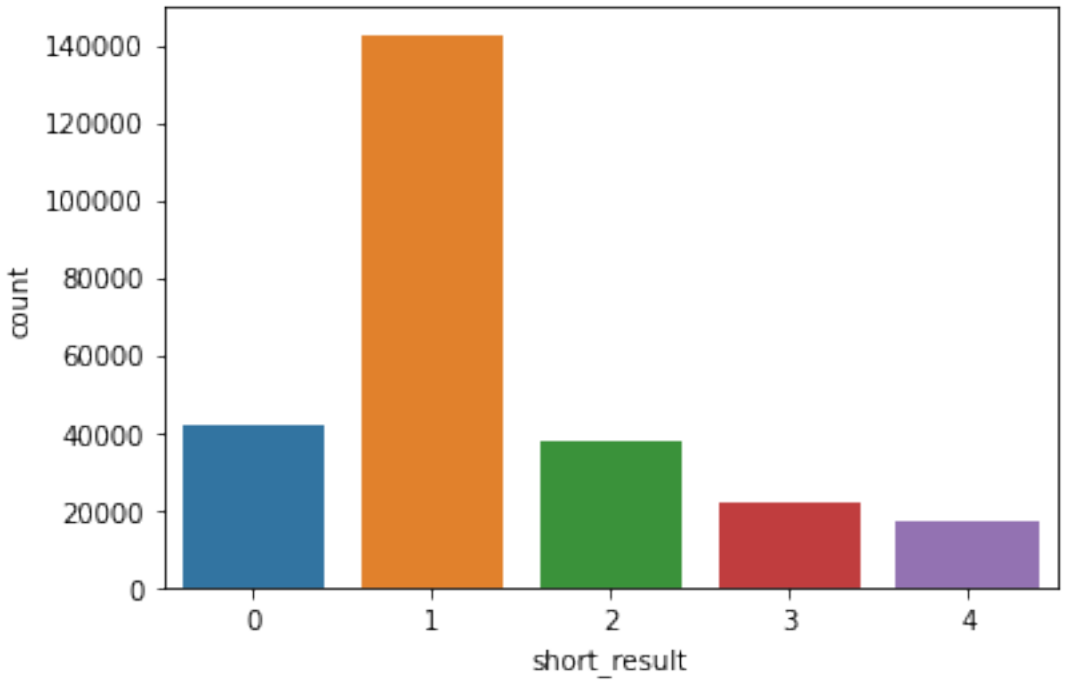
```
3  …  -0.259541     -0.001684        0.015230      0.071205        0.081629
4  …  -0.427895     -0.011353        0.009914      0.065696        0.078443

    MOM_MOM   ROC_ROC    RSI_RSI  IFT_RSI_IFT_RSI  short_result
0  0.160156  5.345912  52.326346         0.128566             1
1  0.063477  2.293814  46.603432         0.042299             0
2  0.023438  2.489627  44.722182         0.004378             0
3 -0.085938 -1.844262  33.656509         2.017359             0
4 -0.179688 -2.443992  33.656509       678.794701             0

[5 rows x 40 columns]
```

[10]:
```
stocks['short_result'].value_counts()
```

[10]:
```
1    142579
0     42264
2     38083
3     21801
4     17129
Name: short_result, dtype: int64
```

[11]:
```
sns.countplot(stocks['short_result'])
```

[11]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11a441710>
```



[12]:
```
X = stocks.drop('short_result', axis=1)
y = stocks['short_result']
```

0.1.2 Training

[13]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

[14]:
```
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.fit_transform(X_test)
```

1 Random Forest Classifer

[15]:
```
rfc = RandomForestClassifier(n_estimators=100)
rfc.fit(X_train, y_train)
pred_rfc = rfc.predict(X_test)
```

[16]:
```
# See how our model performed
print(classification_report(y_test, pred_rfc))
print(confusion_matrix(y_test, pred_rfc))
```

```
              precision    recall  f1-score   support

           0       0.65      0.33      0.43      8294
           1       0.68      0.94      0.79     28595
           2       0.72      0.29      0.41      7674
           3       0.58      0.54      0.56      4370
           4       0.73      0.38      0.50      3439

    accuracy                           0.67     52372
   macro avg       0.67      0.49      0.54     52372
weighted avg       0.68      0.67      0.64     52372

[[ 2701  4855    56   621    61]
 [  746 26777   423   521   128]
 [  133  4868  2199   257   217]
 [  481  1426    34  2344    85]
 [   80  1430   332   289  1308]]
```

2 SVM Classifier

[ ]:
```
clf=svm.SVC()
clf.fit(X_train, y_train)
pred_clf = clf.predict(X_test)
```

[ ]:
```
# See how our model performed
print(classification_report(y_test, pred_clf))
print(confusion_matrix(y_test, pred_clf))
```

3 Neural Network

[ ]:
```
mlpc = MLPClassifier(hidden_layer_sizes=(10,10,10), max_iter=200)
mlpc.fit(X_train, y_train)
mlpc_clf = mlpc.predict(X_test)
```

[ ]:
```
# See how our model performed
print(classification_report(y_test, mlpc_clf))
print(confusion_matrix(y_test, mlpc_clf))
```