Machine Learning Homework

Problem Set 2

Submitted by Gaurav Shah (gns274)

Install the package rugarch in an R session and import the dataset dji30ret (showing the Dow Jones 30 Constituents daily log returns from March 16 1987 to February 3 2009) into a data frame. Split your sample into a Training Set: from March 16, 1987 to December 31, 2002 and a Test Set from January 1st 2003 to February 3, 2009.

1. For each stock you will try to predict log return at time t + 1 using log returns from time t to t 90. Accordingly create a data frame that includes the target and explanatory variables for all stocks as well as dummy variables for each stock. Print the summary() and head() of this data frame. Explain why it makes sense to use one model for all stocks instead of 30 individual models.

TrainingDummyStacked.head()

	0	1	2	3	4	5	6	7	8	9	10	11	12	13
0	0.011429	0.000000	0.009238	-0.002317	-0.002312	-0.004608	0.039865	0.024214	-0.028988	-0.016529	-0.009324	0.000000	0.023475	0.019185
1	0.037909	0.011429	0.000000	0.009238	-0.002317	-0.002312	-0.004608	0.039865	0.024214	-0.028988	-0.016529	-0.009324	0.000000	0.023475
2	-0.008791	0.037909	0.011429	0.000000	0.009238	-0.002317	-0.002312	-0.004608	0.039865	0.024214	-0.028988	-0.016529	-0.009324	0.000000
3	-0.006645	-0.008791	0.037909	0.011429	0.000000	0.009238	-0.002317	-0.002312	-0.004608	0.039865	0.024214	-0.028988	-0.016529	-0.009324
4	0.015436	-0.006645	-0.008791	0.037909	0.011429	0.000000	0.009238	-0.002317	-0.002312	-0.004608	0.039865	0.024214	-0.028988	-0.016529

5 rows × 120 columns

9	10	11	12	13	14	...	DV21	DV22	DV23	DV24	DV25	DV26	DV27	DV28	DV3	DV4	DV5	DV6	DV7	DV8	DV9
29	-0.009324	0.000000	0.023475	0.019185	0.014634	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
88	-0.016529	-0.009324	0.000000	0.023475	0.019185	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
14	-0.028988	-0.016529	-0.009324	0.000000	0.023475	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
65	0.024214	-0.028988	-0.016529	-0.009324	0.000000	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
08	0.039865	0.024214	-0.028988	-0.016529	-0.009324	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

TrainingDummyStacked.describe().transpose()

	count	mean	std	min	25%	50%	75%	max
0	116940.0	0.000495	0.020534	-0.37949	-0.010234	0.0	0.011234	0.226528
1	116940.0	0.000495	0.020534	-0.37949	-0.010236	0.0	0.011234	0.226528
2	116940.0	0.000492	0.020534	-0.37949	-0.010242	0.0	0.011227	0.226528
3	116940.0	0.000494	0.020533	-0.37949	-0.010239	0.0	0.011228	0.226528
4	116940.0	0.000496	0.020534	-0.37949	-0.010237	0.0	0.011233	0.226528
...	...	...	...	...	...	...	...	...
DV5	116940.0	0.033333	0.179506	0.00000	0.000000	0.0	0.000000	1.000000
DV6	116940.0	0.033333	0.179506	0.00000	0.000000	0.0	0.000000	1.000000
DV7	116940.0	0.033333	0.179506	0.00000	0.000000	0.0	0.000000	1.000000
DV8	116940.0	0.033333	0.179506	0.00000	0.000000	0.0	0.000000	1.000000
DV9	116940.0	0.033333	0.179506	0.00000	0.000000	0.0	0.000000	1.000000

120 rows × 8 columns

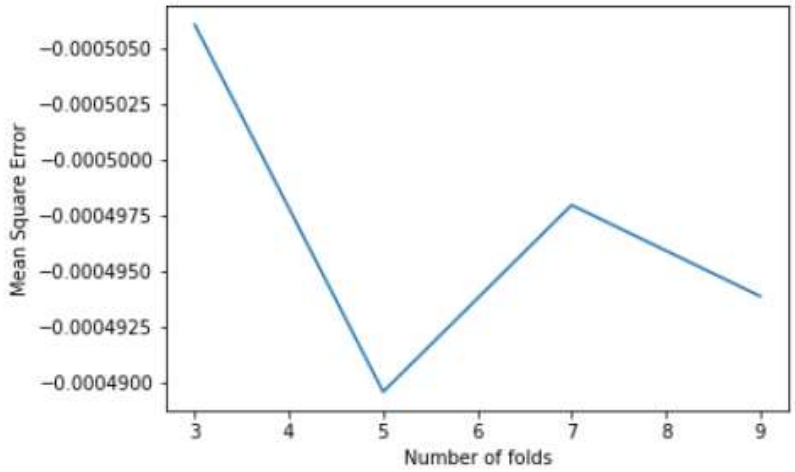
- It makes sense to use one model for all dji30 stocks instead of 30 different models because this reduces model complexity and trouble of managing & maintaining 30 different models.
- These stocks are constituents of DJIA 30, also have high correlation amongst themselves, thus a factor affecting one stock would also be affecting other stocks in the DJI30 basket.
- Eg: DJI Index ETF's when either buying or selling would trade in all the 30 stocks together. Thus, the stocks would have tendencies to move together. Also, the 30 stocks, are large cap stocks which act and move in tandem with the broader indices. Thus, they would be having common factors that would also drive their momentum.

2. Using the Training Set, fit an SVM with a radial kernel to predict the t+1 individual stock return. Use cross-validation to select the relevant parameters.

Cross validated Gamma (Kernel coefficient for 'rbf') and C (Penalty parameter C of the error term)

```
C : 0.01,0.03162278,0.1,0.31622777,1 {np.logspace(-2,0,5)}
Gamma: 1.e-05,1.e-04,1.e-03,1.e-02,1.e-01 {np.logspace(-5,-1,5)}
```

Folds	HyperParameters	MSE
[3 fold,	{'C': 0.3162277660168379, 'gamma': 0.001},	-0.00050602979190098]
[5 fold,	{'C': 0.01, 'gamma': 0.1},	-0.0004895798377606294]
[7 fold,	{'C': 0.01, 'gamma': 1e-05},	-0.0004979474879402921]
[9 fold,	{'C': 0.3162277660168379, 'gamma': 0.001},	-0.00049386710362071]




```
Minimum MSE corresponds to k_folds= 5
Optimal Hyperparamaters: {'C': 0.01, 'gamma': 0.1}
```

Most optimal model:

```
SVR(C=0.01, cache_size=5000, coef0=0.0, degree=3, epsilon=0.1, gamma=0.1,
  kernel='rbf', max_iter=-1, shrinking=True, tol=0.001, verbose=False)

MSE of Train Set:  0.0004913612356904043
MSE of Test Set:  0.0005456991905597548
```

3. Using the Training Set, fit a Random Forest to predict the t + 1 individual stock return.

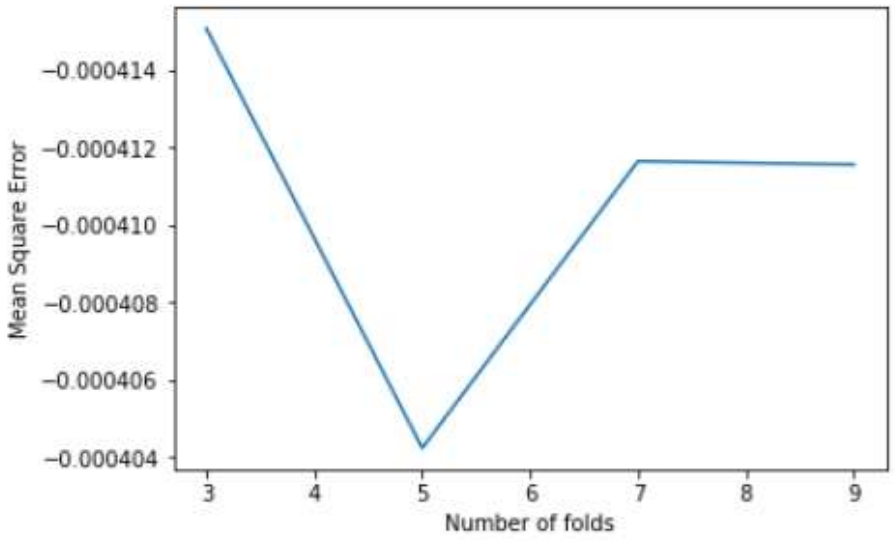
```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
                      max_features='sqrt', max_leaf_nodes=None,
                      min_impurity_decrease=0.0, min_impurity_split=None,
                      min_samples_leaf=1, min_samples_split=2,
                min_weight_fraction_leaf=0.0,n_estimators=100,n_jobs=-1,
                oob_score=False, random_state=5,verbose=0,warm_start=True)

MSE of Train Set:  5.6370824575738186e-05
MSE of Test Set:  0.0004984648104199768
```

4. Using the Training Set, fit a Boosted Tree to predict the t+1 individual stock return. Use cross-validation to select the relevant parameters.

Cross validated learning rate (lambda) and n_estimators (No. of trees/boosting stages)
As fitting a boosted tree itself takes a lot of time, I used a small range of hyperparameters for CV. It took over 2 hours to run 3,5,7,9 fold cross validation with employing parallelization.

```
learning_rate_r : 0.01,0.1
n_estimators : 50,75,100
```

Folds	HyperParameters	MSE
[3,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004150851579774712]
[5,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004042369016882143]
[7,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004116567062804378]
[9,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004115662780722 2875]




```
Minimum MSE corresponds to k_folds= 5
Optimal Hyperparamaters: {'learning_rate': 0.1, 'n_estimators': 100}

GradientBoostingRegressor(alpha=0.9, criterion='friedman_mse', init=None,
             learning_rate=0.1, loss='ls', max_depth=3, max_features=None,
             max_leaf_nodes=None, min_impurity_decrease=0.0,
```

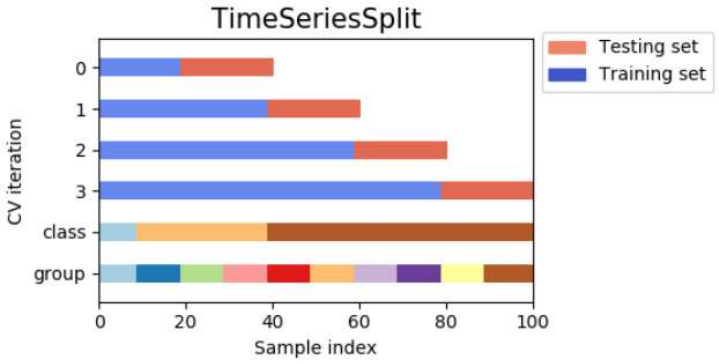
```
                min_impurity_split=None, min_samples_leaf=1,
                min_samples_split=2, min_weight_fraction_leaf=0.0,
                n_estimators=100, presort='auto', random_state=5,
                subsample=1.0, verbose=0, warm_start=True)

MSE of Train Set:  0.0004011487638158693
MSE of Test Set:  0.000510360120948804
```

5. Explain in detail how you performed cross-validation above, including how you ensured that there is no look ahead bias and how you chose the number of folds.

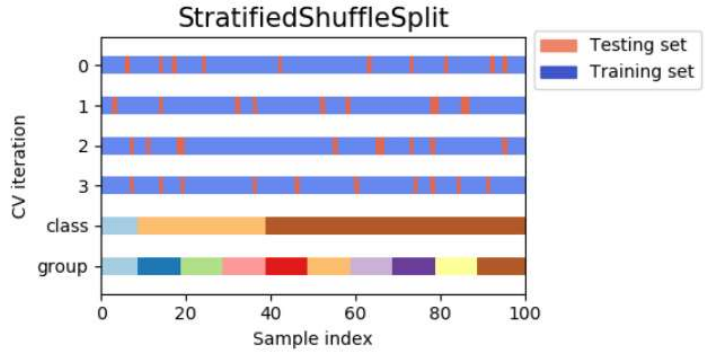
I took two approaches to Cross validaton:

1. **Time Series Split (MSE based)**
2. **Combination of Stratified Shuffle Split and Time Series Split (MSE based) – Could not do this**

Times Series Split is a walk forward approach.



Stratified Shuffle Split is a slicing approach. When combined with Times Series, it only Slices Test Data ahead of Training Data in a time series setting.



I could not perform the "Combination of Stratified Shuffle Split and Time Series Split" due to computational limitations. Time series split(Walk forward) itself took about 2 hours for SVR and about 3 hours for Boosted Trees where it was prone to giving time out errors.

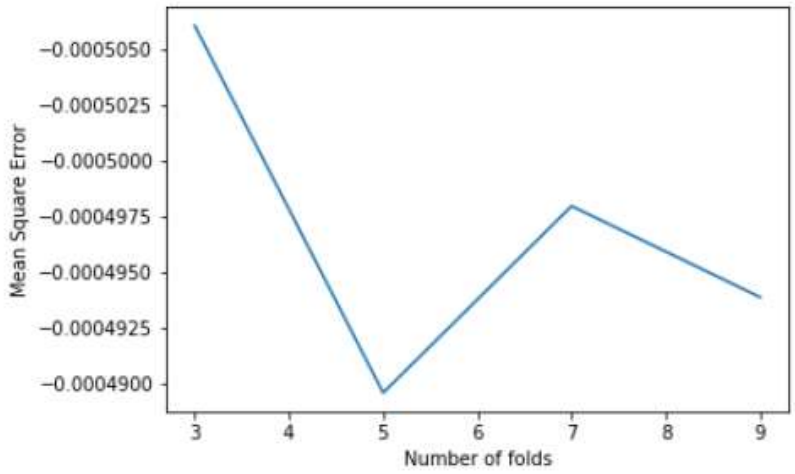
```
learning_rate_r=np.array([0.01,0.1])
n_estimators_r = np.array([50,75,100])
param_grid = dict(learning_rate=learning_rate_r, n_estimators=n_estimators_r)
cv = TimeSeriesSplit(n_splits=i)
grid = GridSearchCV(GradientBoostingRegressor(max_depth=3, random_state=5, warm_start=True), param_grid=param_grid, cv=cv, sc
```

I ensured that there is no look ahead bias by using Times Series Split which tests only on future times test data and trains on the previous data.

I ran cross validation for 3,5,7,9 number of folds for SVM and Boosted Trees. I chose the fold which gave me the lowest MSE.

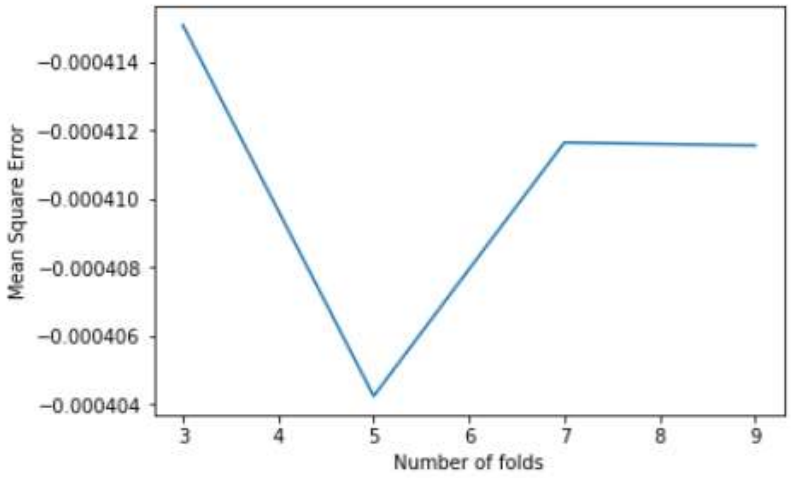
Radial SVM:

Folds	HyperParameters	MSE
[3 fold,	{'C': 0.31622776601683794, 'gamma': 0.001},	-0.00050602979190098]
[5 fold,	{'C': 0.01, 'gamma': 0.1},	-0.0004895798377606294]
[7 fold,	{'C': 0.01, 'gamma': 1e-05},	-0.0004979474879402921]
[9 fold,	{'C': 0.31622776601683794, 'gamma': 0.001},	-0.00049386710362071]



Boosted Trees

Folds	HyperParameters	MSE
[3,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004150851579774712]
[5,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004042369016882143]
[7,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.0004116567062804378]
[9,	{'learning_rate': 0.1, 'n_estimators': 100},	-0.00041156627807222875]



6. Compare the MSE of the different estimated models on the Test Set (do not ret after December 31, 2002). What are your conclusions?

```
Radial Kernel : MSE of Test Set:  0.0005456991905597548
Random Forest : MSE of Test Set:  0.0004984648104199768
Boosted Trees : MSE of Test Set:  0.000510360120948804
```

MSE for Random forest is the lowest among Radial SVM, Random Forest and Boosted Trees. Radial SVM and Boosted Trees are very well fitted with lot of cross validation and carefully choosing the number of folds. Even then, they tend to underperform and cannot beat Random Forest. I feel that Radial Kernel and Boosted Trees fall prey to overfitting. I could not run Boosting for large number of n_estimators due to computational limitations as Boosting was a very hardware intensive activity.

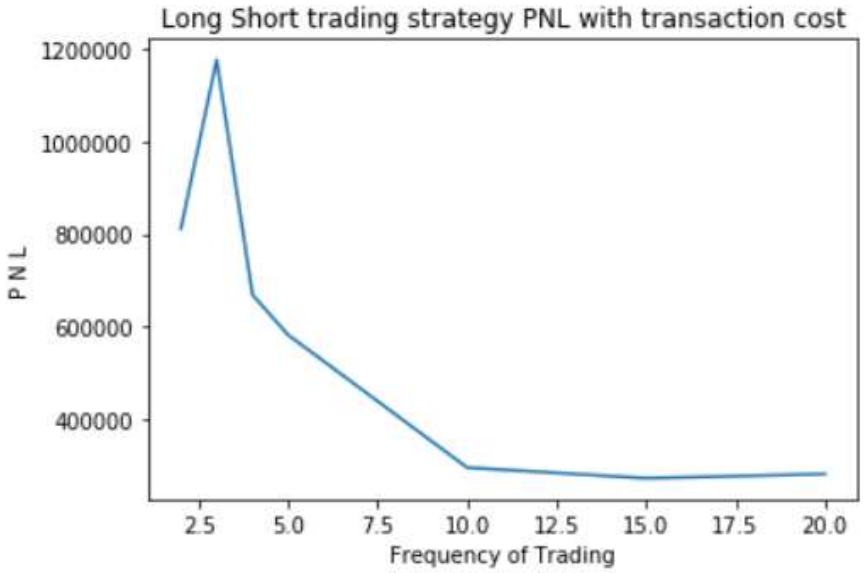
7. Use the best performing model above to construct portfolios on the Test Set (do not ret after December 31, 2002). Specifically, at each time period rank the forecasted returns and create a long/short portfolio comprising of the top 5 and bottom 5 forecasted returns. You should assume that transaction costs are 5bps of traded dollar amount and that at each period you allocate 1=10 of your capital to each of the 5 top and 5 bottom forecasts. Your starting capital is 1; 000; 000 Dollars, how does your strategy perform?

- **Assuming Transaction cost=5bps on both legs, i.e. 2.5bps on either leg**
- **I have coded the strategy to be flexible so that we can change the frequency of trading and plot and see for which frequency would have we gotten the maximum profit.**
- **I have used the following Random Forest Regressor as the predictor model which gave us the lowest Mean Square Error for test data**
- **After introducing transaction costs, the strategy tends to give very low returns and we understand the importance of trading cost and also get to know the reality in the developing a trading strategy.**
-

```
RandomForestRegressor(bootstrap=True, criterion='mse', max_depth=None,
```

```
max_features='sqrt', max_leaf_nodes=None,
min_impurity_decrease=0.0, min_impurity_split=None,
min_samples_leaf=1, min_samples_split=2,
min_weight_fraction_leaf=0.0, n_estimators=100, n_jobs=-1,
oob_score=False, random_state=5, verbose=0, warm_start=True)
```


```
1  day daily trading : Final portfolio value : 597257.2306657671
2  day freq. trading : Final portfolio value : 811641.489175154
3  day freq. trading : Final portfolio value : 1177469.3319886206
4  day freq. trading : Final portfolio value : 668238.6856799033
5  day freq. trading : Final portfolio value : 581070.0626878216
10  day freq. trading : Final portfolio value : 293750.1876725432
15  day freq. trading : Final portfolio value : 270585.7220611559
20  day freq. trading : Final portfolio value : 279836.6908005289
```



3 day frequency trading means that we keep our positions for 3 days as opposed to trading daily. 3 day gives us the maximum PNL and profit.


```
3 day freq. trading : Final portfolio value : 1177469.3319886206
```