MTH-9899: Final Project

Raveesh Soni, Mukesh Pant, Jose Ferreira
City University of New York, Baruch College

Overview

The goal of the project is to apply the various ML algorithms learned in this class in a real-world financial dataset.

Dataset description

The dataset provided consists of approximately 400,000 rows of data with 7 features. Each row represents an observation. The columns in the data file are:

- Date: The date is an int - it does correspond to real dates in the same order.
- Time: Can be ignored. All of the points occur at the same time and the future return is measured roughly 1 day in the future.
- fut ret: This is the dependent variable.
- sec id: This is an identifier for the security that will not change over time.
- vol: This is a proxy for the stock's volatility.
- X*: There are 7 columns with names that start with 'X' - those will be the features that you might consider using for your models.

Data Analysis

Data Cleaning

Data Discard Threshold

The data for some security time-series is not complete, some are missing returns, vol or have a significant amount of zero values. Under the expectation that the hold-out test data is going to be relatively clean, we experimented with a discard threshold that applies on a security level.

If the time-series for a given variable in a security time-series has more than a certain percentage of data missing, we discard the data for this security entirely. A balance between discarding too many securities risking the loss of important examples and keeping a lot of securities having too many empty values needs to be achieved.

An impact assessment on the prediction model out-of-sample scores using the full dataset (including the discarded securities) was performed (Figure 1). We found our model to be relatively robust to discarding securities. Furthermore, we found the scores to be better in average when we trained our model with the cleanest data possible.

To avoid risking the loss of generality we decided for a low threshold (0.25) rather than removing all the securities with empty data.

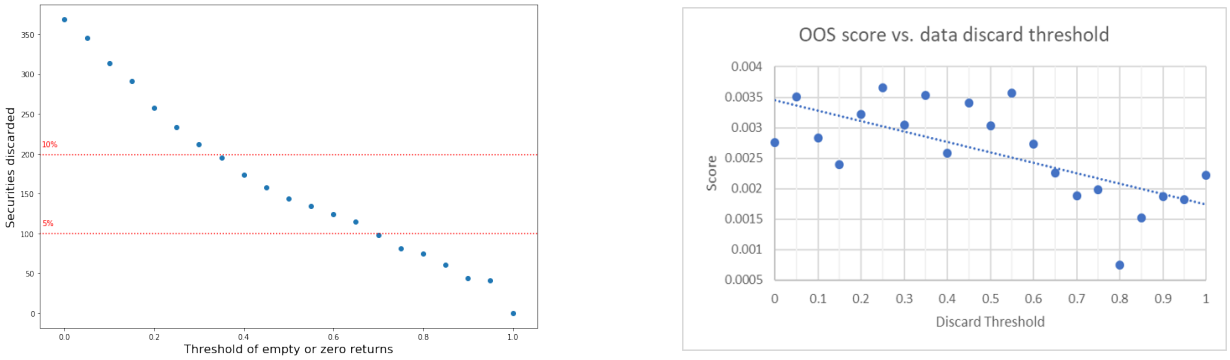

Figure 1: Number of discarded securities (left) and out-of-sample scores(right) per threshold level

For the N/A values remaining after the discard threshold exercise, we fill the space with the previous value in the time-series when it exists or with the dataset's median value for that variable when it doesn't.

Normalization

We first normalize the original variables using the time-series data per security rather than the cross-sectional data. Once the complete set of variables fed to the estimator model is decided, a normalization across the entire dataset is performed.

After the two normalization processes, a visual inspection (Figure 2) indicates the results are reasonable. Some of the data has jumps introduced by the N/A value filling procedure which possibly explains the lower R-square scores in data with high discard thresholds.

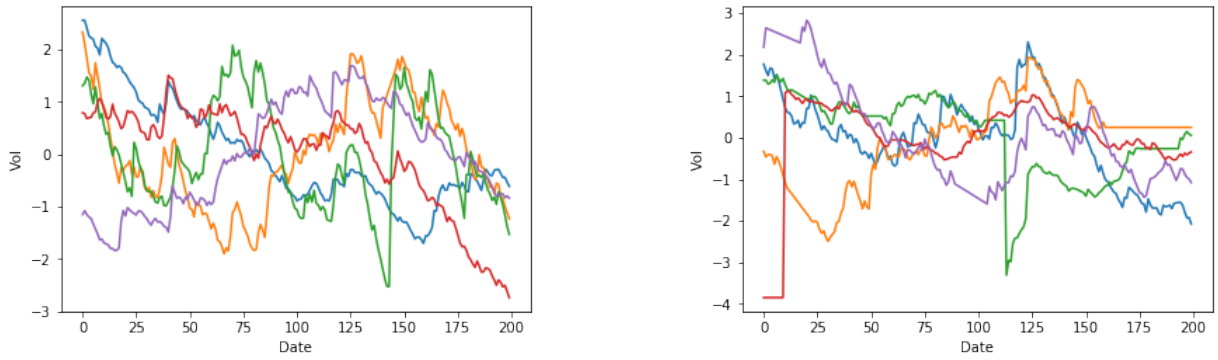

Figure 2: Some examples of the normalized vol time-series. On the right, some artifacts introduced by the N/A filling process can be observed

Feature Selection

Stationarity and Fractional Differentiation

As part of the data analysis stage, we performed a stationarity test on all variables per security and we determined that the vol variable is not stationary. A common approach is to use a lagged variable to make the feature stationary. However, this is not an optimal approach

since this procedure removes all memory and correlation with previous values.

A way to preserve some of the memory in the time-series is fractional differentiation [2] which is a generalization of the lag operator for non-integer steps.

Fractional Differentiation implementation

Let B denote the backshift operator as $BX_t = X_{t-1}$ for $t > 1$ a time series $X = X_1, ...$
A lag of first order can be expressed with the identity operator I as

$$(I - B)X_t = X_t - BX_t = X_t - X$$

We can extend this definition to include differentiation with a non-integer factor d by applying the binomial series expansion:

$$\begin{aligned}(1-B)^d =& \sum_{k=0}^{\infty} \binom{d}{k} (-B)^k \\ =& \sum_{k=0}^{\infty} (-B)^k \frac{\prod_{i=0}^{k-1}(d-i)}{k!} \\ =& 1 - dB + \frac{d(d-1)}{2!}B^2 - \frac{d(d-1)(d-2)}{3!}B^3 + \cdots\end{aligned}$$

We can implement an approximation to this operation by establishing a fixed-size window and calculating the corresponding weights for each value within this window for a given factor d as

$$w_k = -\frac{w_{k-1}(d-k+1)}{k}$$

Establishing the optimal fractional differentiation factor

We aim to make all the instances of the vol time-series stationary and to establish the minimum value of the fractional differentiation factor that achieves this as to be able to preserve

the maximum possible of correlation with the original series.

We ran a simple grid search over the entire dataset and generated the lagged series using different values of d. Then, tested for stationarity using the Augmented Dickey-Fuller unit root test. We found a factor of 0.7 to be enough to transform more than 98% of the vol time-series (Figure 3).

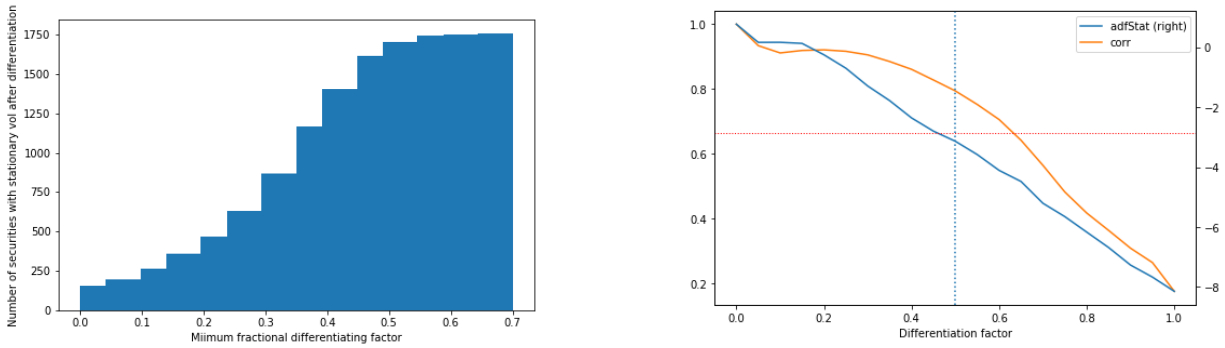

Figure 3: Histogram of minimum values to make the vol series stationary(left). Correlation preserved for a random security and its optimal differentiation factor (right)

Polynomial Features

After analysis of the input variables, we noticed that, for instance, variables X3, X4 and X6 seem to represent decay functions starting from an event in the security's time series. Also, variable X7 seems to indicate another type of punctual event in time.

We determined that combinations of variables might be more expressive than the individual features and tried generating polynomial features of second degree by pairing two features, see Figure 4 for an example of such combination.

We integrated these combinations in the model and selected the most significant features using the MDA and RFE feature selection algorithms. Including the best polynomial combinations increased our R2 scores by approximately 3-5 bps.

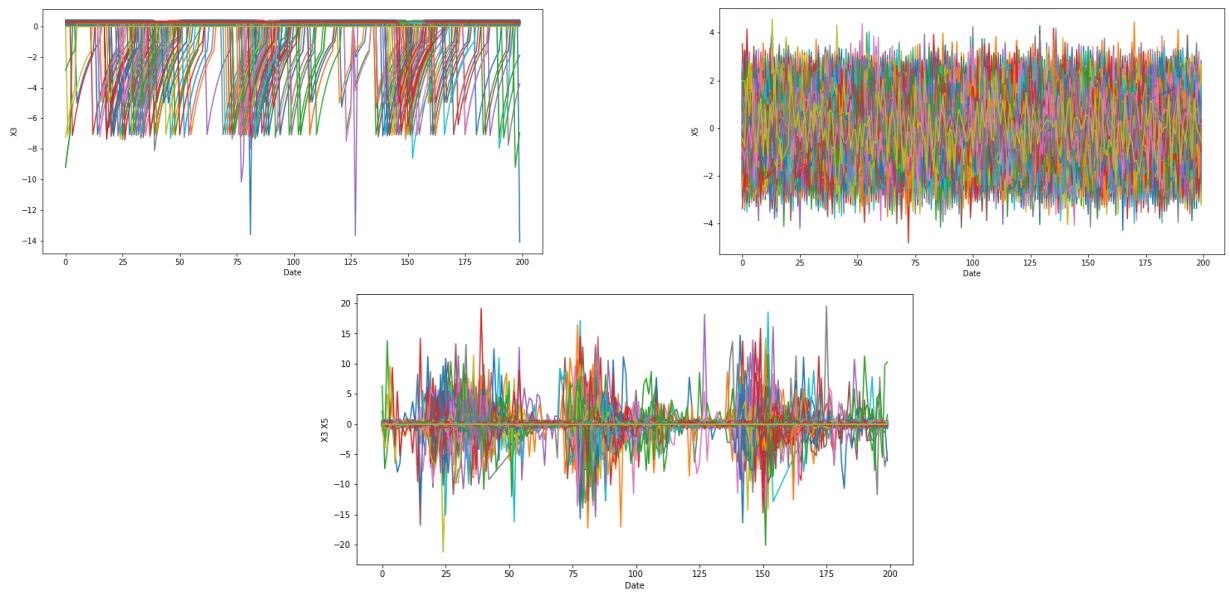

Figure 4: Two individual features (top) and the result of multiplying them (bottom)

Mean Decrease Accuracy

A method to select the most significant features for an estimation and simplifying the model as a result called Mean Decrease Accuracy (MDA) [2] was applied to the set of features extended by including the polynomial features described in the previous subsection. The method consists of fitting an estimator and calculate its out-of-sample score. Then, permutate the values in each feature and calculate the score again with the same method.

The difference between these two scores is a measure of how important the feature is since the permutation would have removed any relationship that the first run established. The method is run as times as possible and the results are averaged for accuracy. It's important to mention that this method does not discover the importance of variables being combined as the evaluation is done per individual feature and therefore it is fundamental to include these combinations as individual features in the input set.

In our case, for the estimator, we used Lasso linear regression as the L1-norm criteria effectively identifies unimportant features and the algorithm strikes a good balance between

accuracy and speed. Once the calculation of the MDA measure is completed for all the features, we selected the best half according to this importance score (Figure 5).

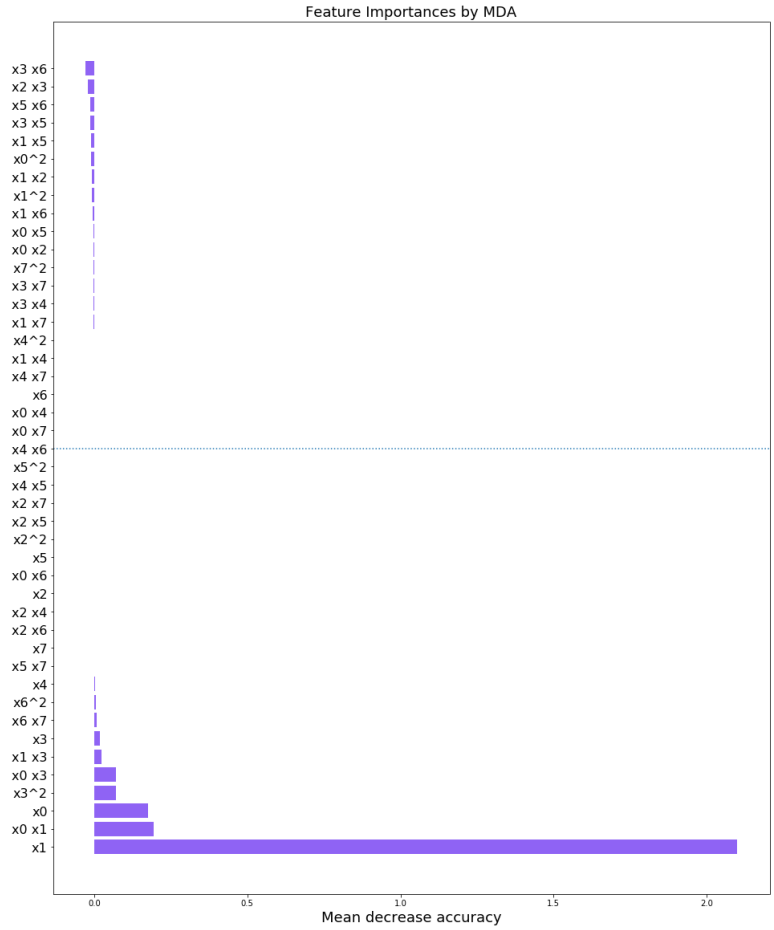

Figure 5: Features selected by the MDA algorithm using a Lasso estimator with cross-validation

Recursive Feature Elimination

We use Recursive Feature Elimination algorithm as a challenger to MDA to select features

Recursive feature elimination (RFE) is a feature selection method that fits a model and removes the weakest feature (or features) until the specified number of features is reached. Features are ranked by the model's coefficient (Linear Regression in our case) since we use Linear regression as a weak learner

RFE requires a specified number of features to keep, however it is often not known in advance how many features are valid. To find the optimal number of features we used cross-validation to score different feature subsets and select the best scoring collection of features

Model Selection

We tested various models to see which model gives us the best out of sample r square. We selected the parameters by cross-validation to avoid over fitting. Below are the results of running various models on the dataset

Model	Insample R^2	OOS R^2
Linear regression	6	8
Random Forest	78	54
K Nearest Neighbor	545	7
Gradient Boosting Machine	545	18

As can been seen above that Gradient Boosting nachine algorithm gave us most stable r squares so we go with GBM as a champion model. Below is some theory about GBM Model:

In any function estimation problem we wish to find a regression function $\hat{f}(x)$, that minimizes the expectation of some loss function $\psi(y, f)$

$\hat{f}(x) = argmin_{f(x)} \mathbf{E}_{y,x} \psi(y, f(x))$

Initialize $\bar{f}(x)$ to be a constant, $\bar{f}(x) = argmin_{\rho} \sum_{i=1}^{\mathbf{N}} \psi(y_i, \rho)$

For t in $1, ..., T$ do

1. Compute the negative gradient as the working response

$$z_i = -\frac{\partial}{\partial f(x_i}\psi(y_i, f(x_i))|_{f(x_i)=\bar{f}(x_i)}$$

2. Fit a regression model, g(x), predicting z_i from the covariates x_i

3. Choose a gradient descent step size as

   $\rho = argmin_\rho \sum_{i=1}^{N} \psi(y_i, \hat{f}(x_i) + \rho g(x_i))$

4. Update the estimate of $f(x)$ as

   $\bar{f}(x) \leftarrow \hat{f}(x) + \rho g(x)$

   =$argmin_{f(x)} \mathbf{E}_x[\mathbf{E}_{y|x}\psi(y, f(x))|x]$

   We will focus on finding the estimates of $f(x)$ such that

   $\hat{f}(x) = argmin_{f(x)} \mathbf{E}_{y|x}[\mathbf{E}_{y|x}\psi(y, f(x))|x]$

Parametric regression models assume that $f(x)$ is a function with a finite number of parameter and estimates them by selecting those values that minimize a loss function (e.g. squared error loss) over a training sample of N observations on (y, x) pairs

$\hat{\beta} = argmin_\beta \sum_{i=1}^{\mathbf{N}} \psi(y_i, f(x_i; \beta))$
When we wish to estimate $f(x)$ non-parametrically the task becomes more difficult. Again we can modify our current estimate of $f(x)$ by adding a new function f(x) in a greedy fashion. Letting $f_i = f(x_i)$, we see that we want to decrease the N dimensional function
$J(f) = \sum_{i=1}^{\mathbf{N}} \psi(y_i, f(x_i))$

$= \sum_{i=1}^{\mathbf{N}} \psi(y_i, F_i))$

The negative gradient of J(f) indicates the direction of the locally greatest decrease in $J(f)$. Gradient descent would then have us modify f as

$\hat{f} \leftarrow \hat{f} - \rho\delta J(f)$

where rho is the size of the step along the direction of greatest descent. Clearly, this step alone is far from our desired goal. First, it only fits f at values of x for which we have observations. Second, it does not take into account that observations with similar x are likely to have similar values of $f(x)$. Both these problems would have disastrous effects on generalization error. However, we can choose a class of functions that use the covariate information to approximate the gradient, usually a regression tree. This line of reasoning produces his Gradient Boosting algorithm. At each iteration the algorithm determines the direction, the gradient, in which it needs to improve the fit to the data and selects a particular model from the allowable class of functions that is in most agreement with the direction. In the case of squared-error loss

$\psi(y_i, f(x_i)) = \sum_{i=1}^{\mathbf{N}} y_i - f(x_i))^2$

this algorithm corresponds exactly to residual fitting

Parameter Tuning

We use Grid search in scikit learn and parmeter searching to come up with best estimates for the GBM algorithm. We try to optimize the below parmaters to produce the best out of sample R-square

- **Learning Rate** Learning rate shrinks the contribution of each parameter, this helps to not overfit the model
- **No Of Estimator** This represents the number of trees in the forest, increasing the number of trees may lead to over fitting so we need to choose the parameter carefully
- **Max Depth** This indicates how deep the tree can be, the deeper the tree, the more splits it has and it could lead to over fitting of data

- **Min Samples Split** This represents the minimum number of samples required to split an internal node
- **Min Samples Leaf** This is the minimum number of samples required to be at the leaf node

Grid search CV tests a combinations of parmeters and uses cross-validation to come up with best parameter sets

References

[1] De Prado, Marcos Lopez. Advances in financial machine learning. John Wiley & Sons, 2018.

[2] the elements of statistical learning by hastie tibshirani and friedman &