MTH-9899

Final Project Presentation

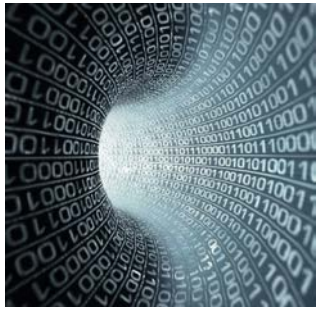
Raveesh Soni, Mukesh Pant, Jose Ferreira

Overview: Problem Description

Future stock return prediction

- Goal: Apply the various ML algorithms you have learned in this class to a real-world financial dataset.
- Data: The dataset you will be given consists of approximately 400k rows of data with 7 features. Each row represents an observation. It is your discretion whether to keep them all, consider a subset, or maybe even enlarge it.
- Columns
  - 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 cols with names that start with 'X' - those will be the features that you might consider using for your models

Overview: Our Approach



Data Analysis

- Split by security
- K-means clustering

Data Cleaning

- Discard incomplete data
- Fill blanks

Feature Selection

- Stationarity and fractional differentiation
- Polynomial features
- MDA and RFE selection

Model Selection

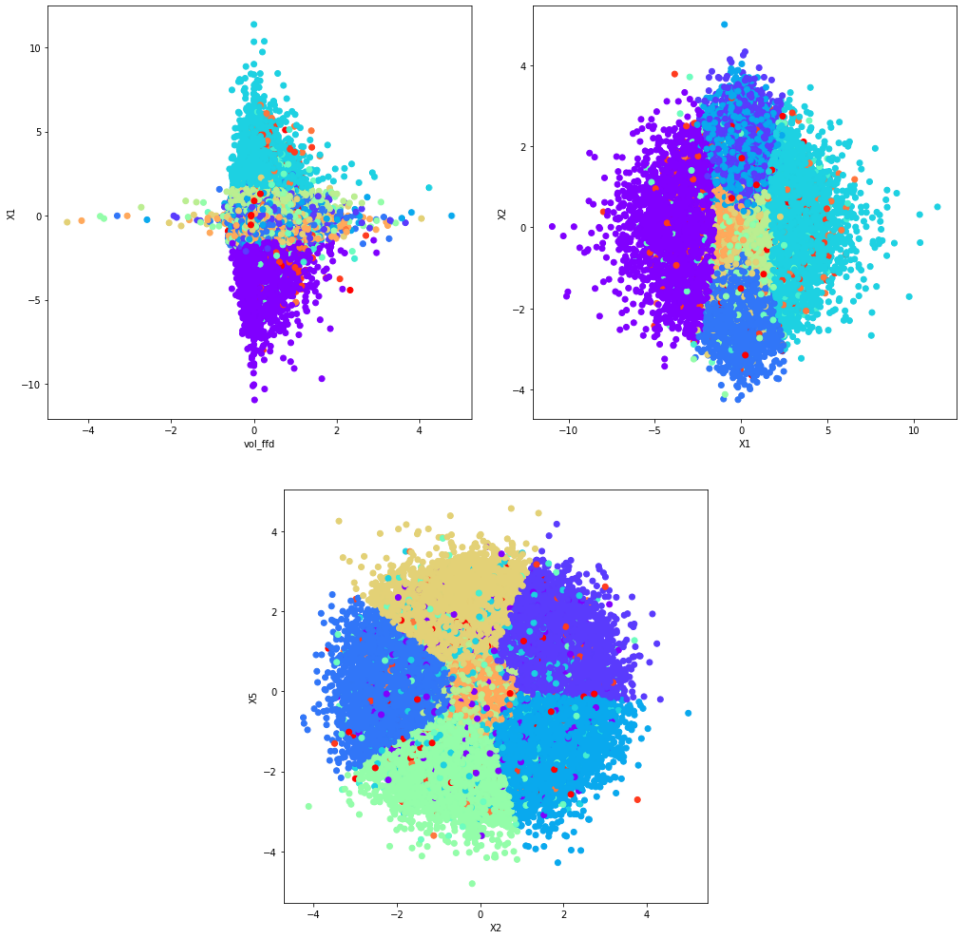
- Cross-validation
- Model evaluation

Parameter Tuning

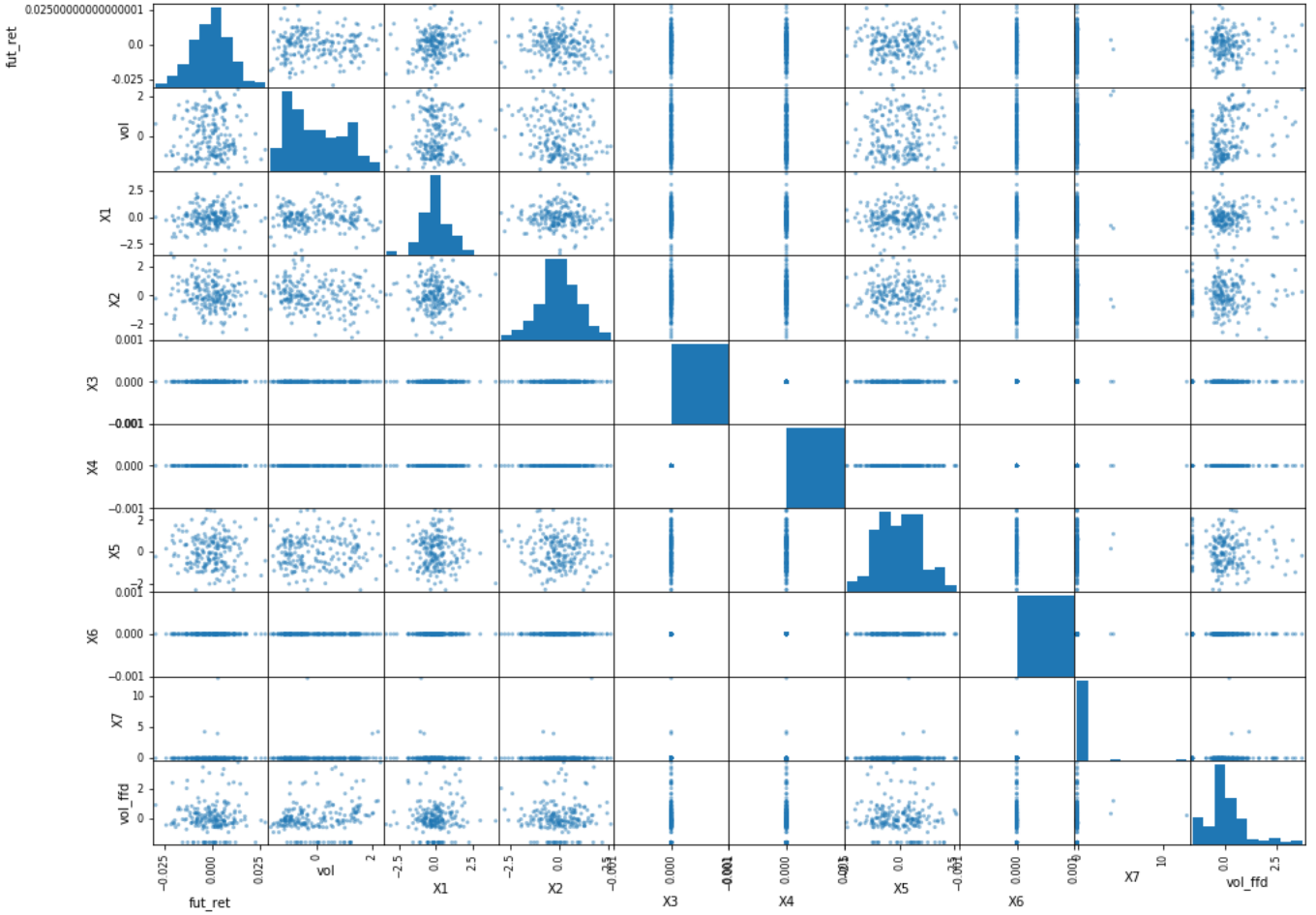
- Grid search
- Paramter Search

Data Analysis

Time-series by security view, clustering for cross-sectional view



K-means clustering (k=8) of cross-sectional data

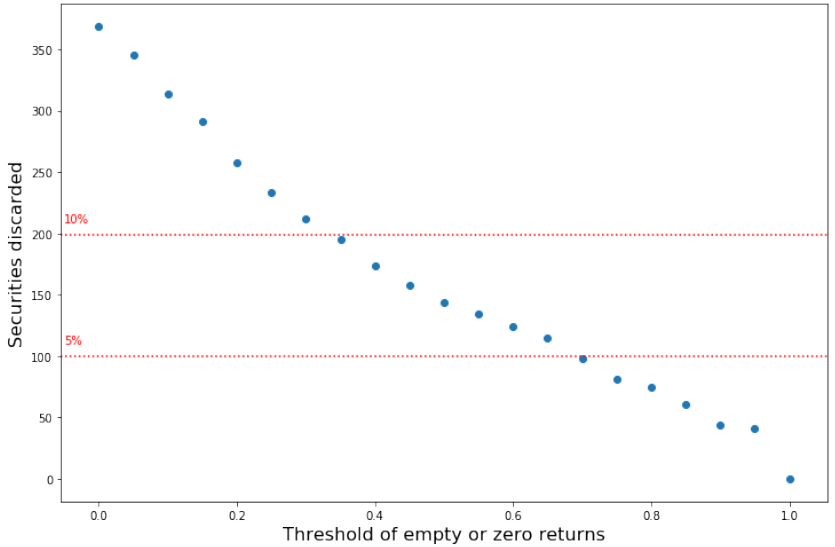


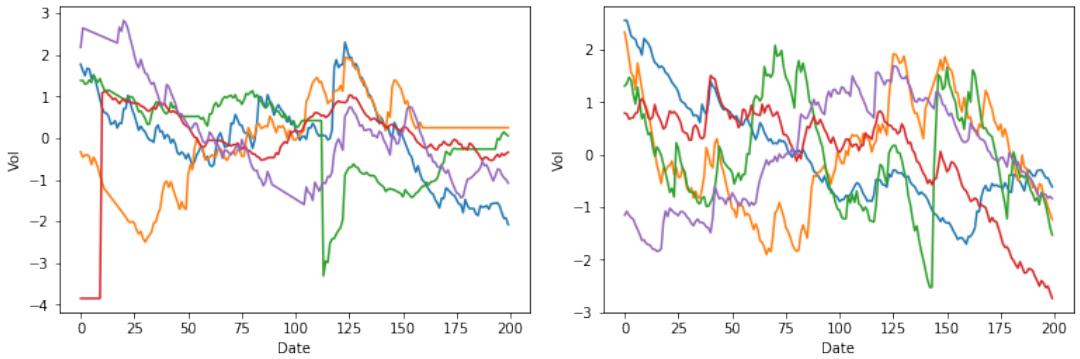
Time-series scatter matrix for a random security

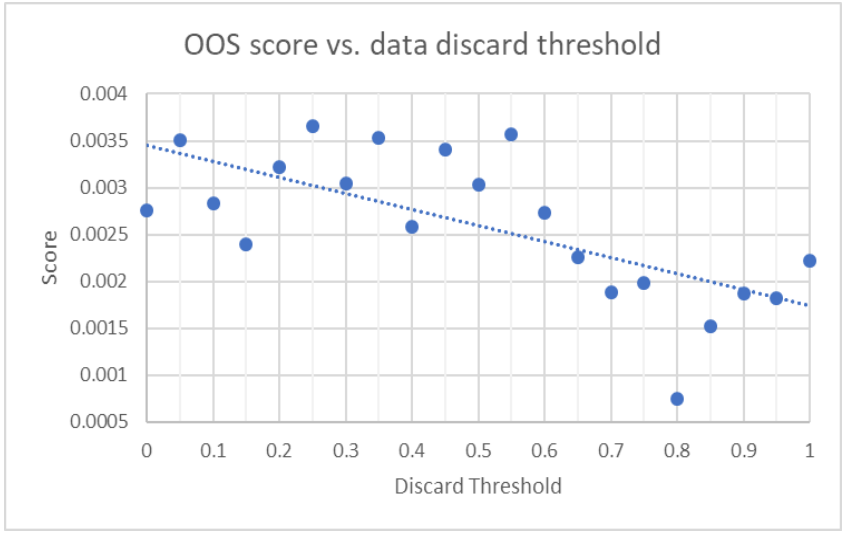
Data Cleaning

Data cleaning the most important piece before starting any Machine learning Algorithm

- Discard securities with more than a threshold percentage of returns missing (or zero).
- For the remaining N/A values, fill with the previous value in the time-series when it exists, fill with the dataset's median value for that variable when it doesn't.
- Normalize the variables using the time-series per security rather than the cross-sectional data.
- Normalize again across the entire dataset when all the features have been selected.







Feature Selection: Stationarity and Fractional Differentiation

Preserve memory while ensuring stationarity

- The ‘Vol’ variable is not stationary. A common approach is to use a lagged variable to make it stationary. However, this would remove all memory of previous values.
- A way to preserve some of the memory is fractional differentiation.

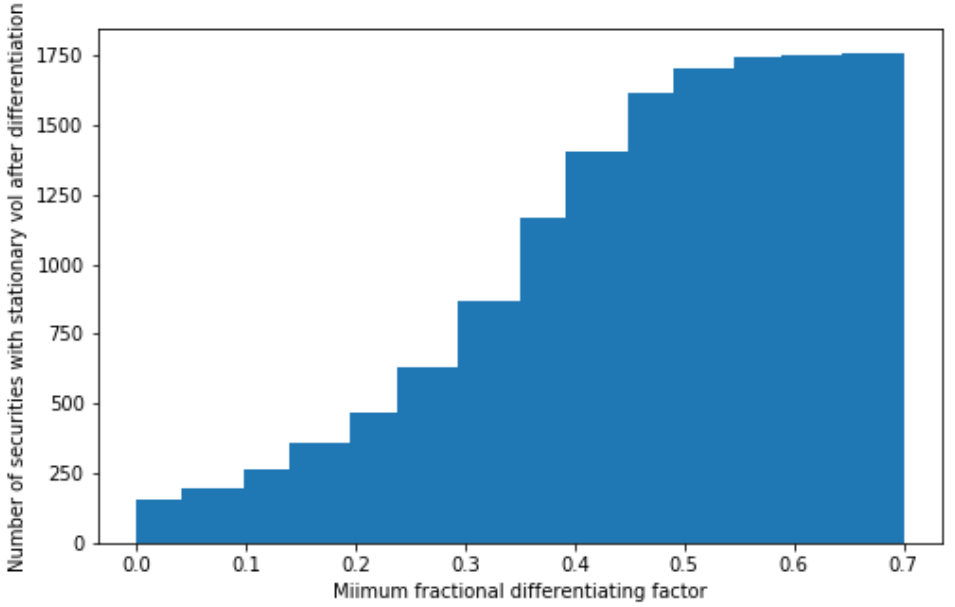
Let B denote the backshift operator as $BX_t = X_{t-1}$ for $t > 1$ a time series $X = X_1, ...$
A lag of first order can be expressed with the identity operator I as

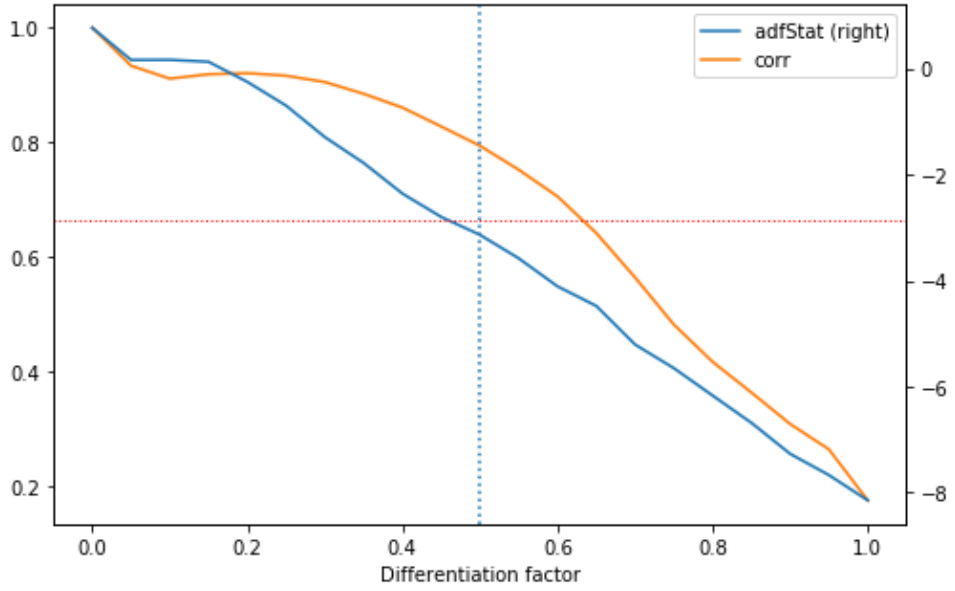
$$(I - B)X_t = X_t - BX_t = X_t - X$$

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 established the minimum value of the fractional differentiation factor that makes all the ‘Vol’ series per security stationary (determined by the Augmented Dickey-Fuller unit root test) to be 0.7.

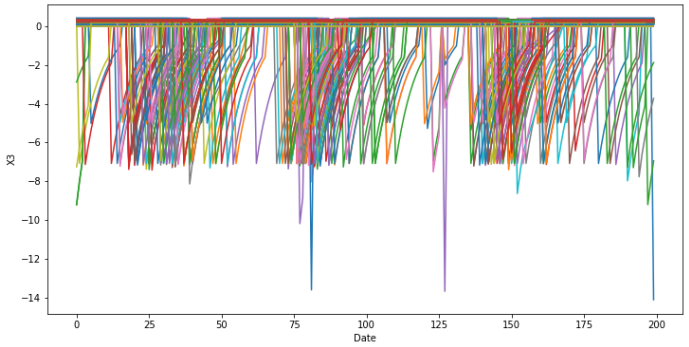


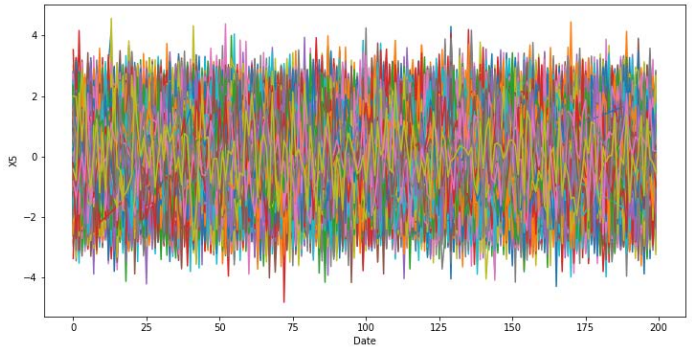


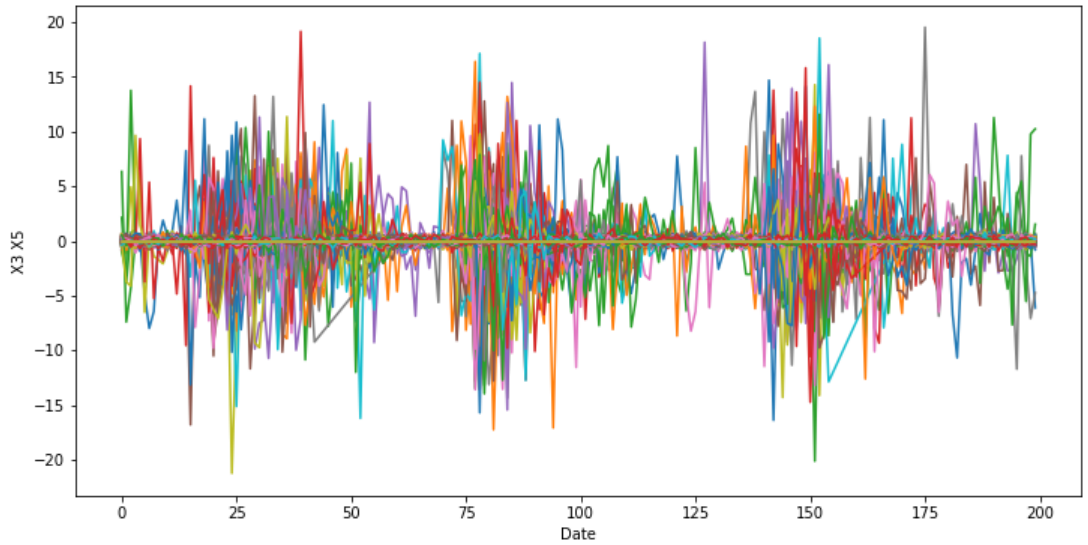
Feature Selection: Polynomial Features

More power of expression for the estimators

- After analysis of the input variables, we thought a combination of them could be more significant than using them independently.
- For instance, variables ‘X3’, ‘X4’ and ‘X6’ seem to represent decay functions starting from an event in the security’s time series. Variable ‘X7’ seems to indicate another type of punctual event in time. Multiplying them with other variables in the dataset can be beneficial.
- We included polynomial combinations of second degree in the model and selected the best using feature selection algorithms (to be explained in the following slides).
- Including the best polynomial combinations increased our R^2 scores by approximately 3-5 bps.





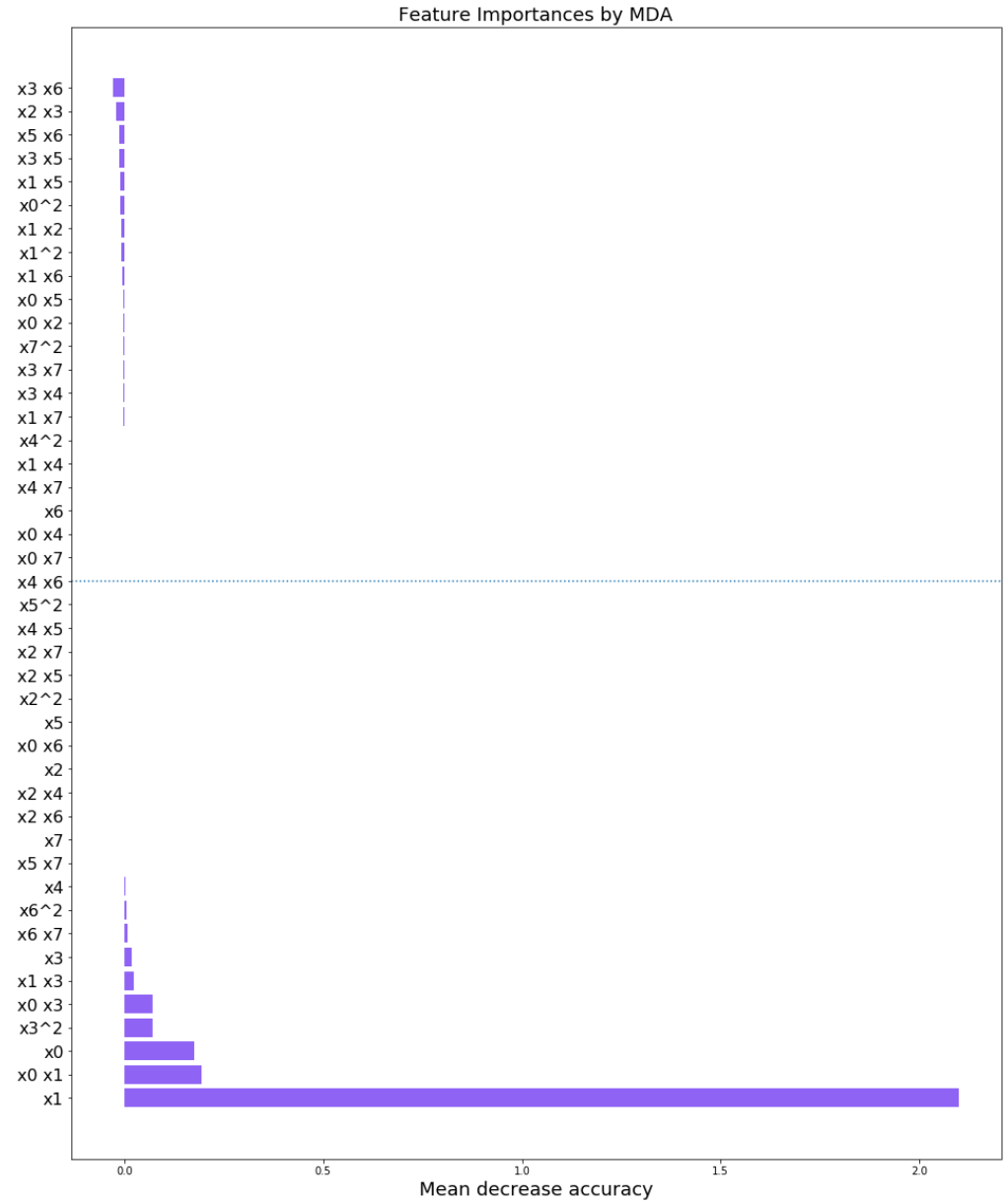


*Combination of features: X3, X5 and X3*X5*

Feature Selection: MDA (Mean Decrease Accuracy)

Determine importance of features by permutation

- Fit an estimator and calculate a score. Then, permutate the values in each feature and calculate the score again. The difference between these two scores is a measure of how important the feature is. Run many times and accumulate the results for accuracy.
- Does not discover importance of combinations of variables.
- Can be used with any estimator, we used Lasso linear regression as this method is known to help selecting features and it offered a good balance between accuracy and speed.
- We selected the top half of the features according to the MDA importance score.

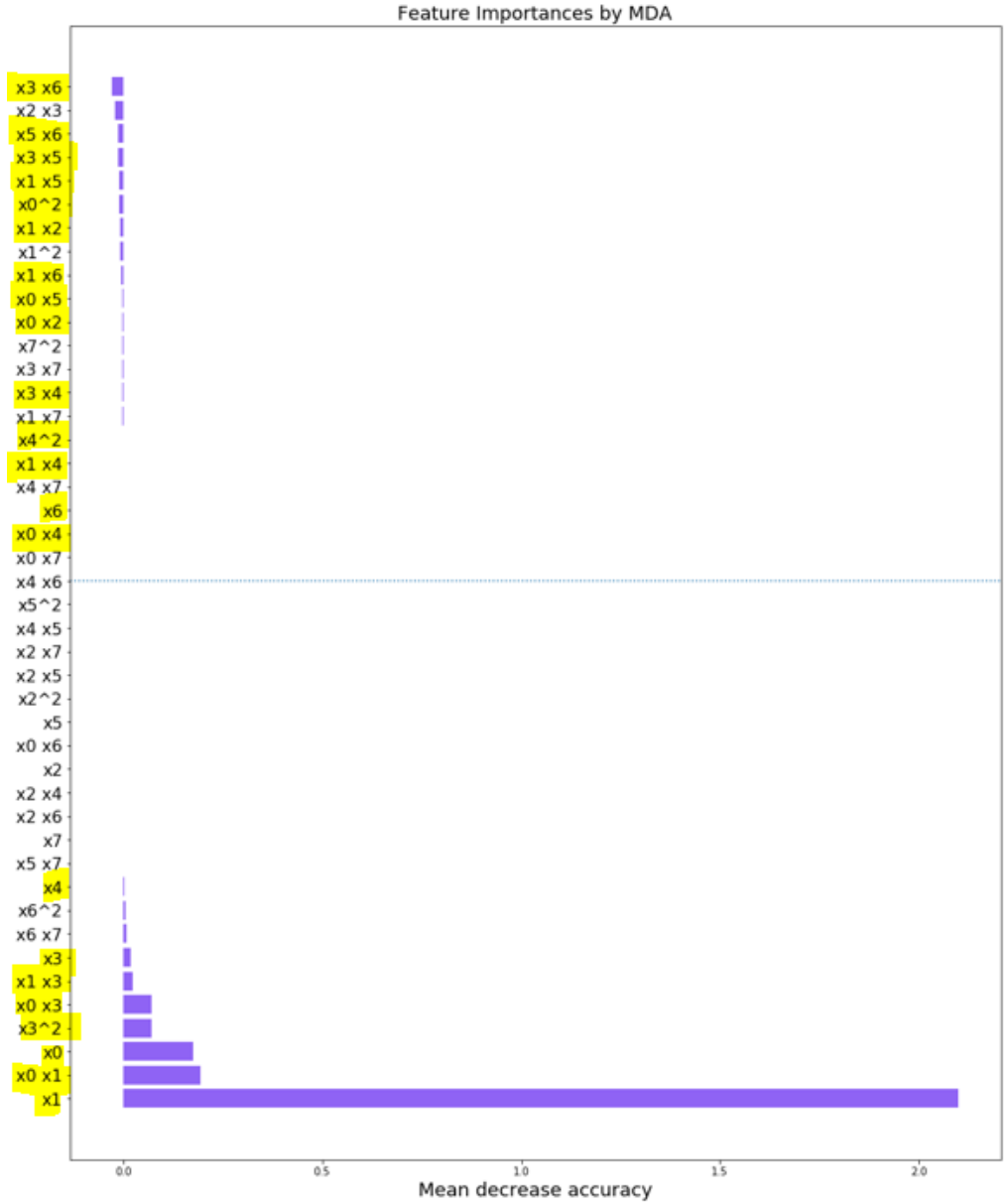


Feature Selection: RFE (Recursive Feature Elimination)

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

Tried different models and ensembles and chose model with highest OOS r square. We also tried ensembles of Clustering (K means) and then running different models

Model/Ensemble	In sample R square	OOS R Square
Linear Regression	0.1218%	0.1716%
Random Forest (BEST)	0.5055%	0.3591%
Gradient Boosting (BEST)	**0.9609%**	**0.4852%**
K means cluster + Linear Regression	0.2640%	0.1254%
K means cluster + Gradient Boosting	1.4626%	0.1491%
LSTM	0.0722%	0.1605%

Parameter Tuning

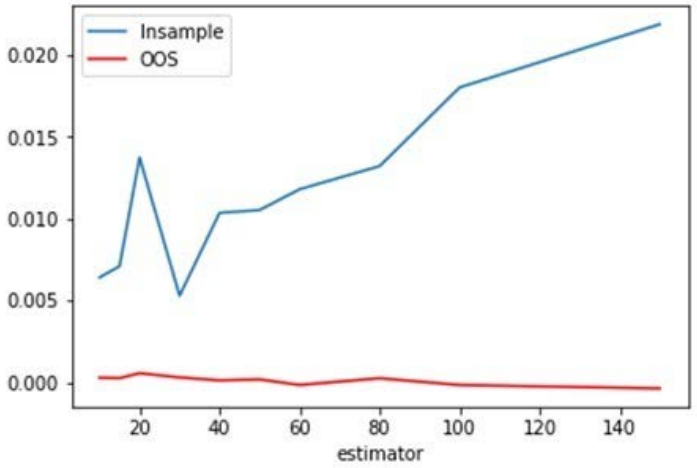
Grid Search and parameter search to select the best combination of model parameters. We need to tune the following parameters for Gradient Boosting Machine:

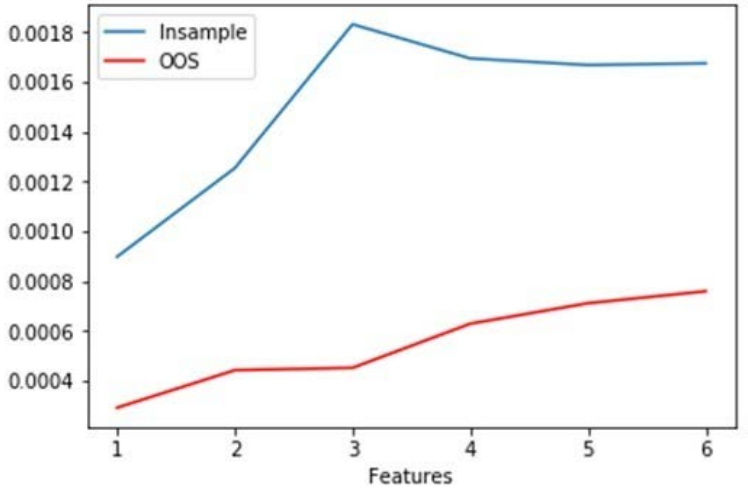
1. Learning Rate: Learning rate shrinks the contribution of each parameter, this helps to not overfit the model
2. No of estimators: This represents the number of trees in the forest, increasing the number of trees may lead to over fitting so we need to choose the parameter carefully
3. Max Depth: This 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
4. Min Samples Split: This represents the minimum number of samples required to split an internal node
5. Min Samples Leaf: This is the minimum number of samples required to be at the leaf node

Parameter Tuning(Contd….)

Parameter Search: We can run in sample and OOS r square by different parameters (keeping others constant) and see which parameter value produces highest OOS r square. See images below.

Although this method looks good but is naïve and arduous and does not test combinations





We finally use Grid Search CV from scikit learn to tune hyper parameters for GBM algorithm because it has below advantages

1. Tests all Combinations
2. Does Cross Validation

Finally GBM gives us an OOS r square of 25 bps which looks dismal but is as per expectations

Thank You