Two Sigma Financial Modeling

Capstone Project

Machine Learning Engineer Nanodegree

Adimali Piyadasa
Sep 30, 2017

Definition

Project Overview

Financial forecasting is one of the most challenging predictive modeling tasks in today's world. Financial data in general considered as very volatile with low signal-to-noise ratio. Financial modeling gets more and more complex over time due to exponentially increasing volume of market and economic information. Most of the new information being generated is just noise further hindering the prediction process for making intelligent investment decisions. With ever increasing financial market information we need more powerful models to train on them. These models can be trained using variety of data inputs such as historical market data and various news feeds to predict the financial movements of different assets. Due to increasing availability and reduced cost of powerful computers, financial institutes are looking in to employing complex predictive models to forecast the market movements of their financial assets. Numerous studies have been conducted to forecast the financial market using different machine learning techniques and to evaluate their applicability in the financial domain. [1, 2, 3, 4] This project is focused on building several predictive models and evaluating their performance in forecasting the returns from multiple financial assets given in Kaggel 2sigma financial modeling competition data set. [5]

Problem Statement

The goal of this project is to create several predictive models to forecast the trend of future returns from financial assets using several features. Asset returns that need to be predicted are continuous floating point numbers in the range of -0.1 to 0.1, making this a regression problem. The Kaggle 2sigma financial modeling competition data set contains 110 features that can be used as inputs for the model. Two of these features are integers namely "Id" and "timestamp" while the rest are floating point numbers. The physical meaning of the features and how they were derived was kept confidential by the Two Sigma Investments to prevent the use of domain knowledge. Created models will then be evaluated and compared using R value between predicted and actual returns.

Metrics

I will use the R value between predicted and actual returns to evaluate the model performance, as used in the Kaggle competition. R value is calculated using R squared (coefficient of determination).

$$R^2 = 1 - \frac{\sum_i (y_i - \hat{y})^2}{\sum_i (y_i - \mu)^2}$$

$$R = sign(R^2)\sqrt{|(R^2)|}$$

Where y is the actual value, μ is the mean of the actual values, and $\hat{y}$ is the predicted value. In finance, given the high ratio of signal-to-noise, even a small R can deliver meaningful value!

Analysis

Data Exploration

The dataset being used is provided by the Two Sigma Investments for its Kaggle competition named "Two Sigma Financial Modeling Challenge". [5] Data set consists of different assets (Ids) and their returns over a certain time period. Complete dataset has 1.7 million examples each with 110 features and one asset return value. In this dataset there are 1424 unique assets and 1813 unique timestamps. Portfolio of assets is constantly being changed by selling and buying new assets. I will be using all the 110 features in the initial step for exploitative data analysis.

Some of the features are not available for certain assets and denoted by NaN. All the features are continuous floating point numbers (except for id and timestamp) and are distinctively named in three groups "derived", "fundamental" and "technical". Further information on the features such as how they are obtained or the physical meanings of them are kept confidential to prevent use of any domain knowledge in the predictive model. Data set will be split based on the "timestamp" to prevent the look ahead bias, first 75% of the examples will be used as training and validation (k-fold cross validation) set while the last 25% will be used as the testing set.

Some of the features have very skewed distribution with a large range as shown in Table 1. These features will have to be transformed before training the model.

	count	**mean**	**std**	**min**	**25%**	**50%**	**75%**	**max**
fundamental_17	1.6E+06	7.1E+13	8.1E+15	-3.6E+16	-2.6E-01	-1.8E-02	1.2E-01	1.0E+18
derived_1	1.6E+06	7.7E+11	7.6E+13	-7.4E-02	-3.0E-02	5.5E-03	1.1E-01	1.1E+16
fundamental_61	1.0E+06	9.3E+11	1.2E+14	-1.8E-01	-8.9E-02	-4.2E-04	1.6E-01	1.7E+16

Table 1: Several features with skewed distribution and large range.

Exploratory Visualization

Our goal in this project is to fit a predictive model for the asset return value y. From the Figure 1 and Table 2 it can be seen that the y values are normally distributed around 0 with maximum value of 0.093 and minimum value of -0.086. Two peaks at the tails of the histogram suggest that the y values might have been capped.

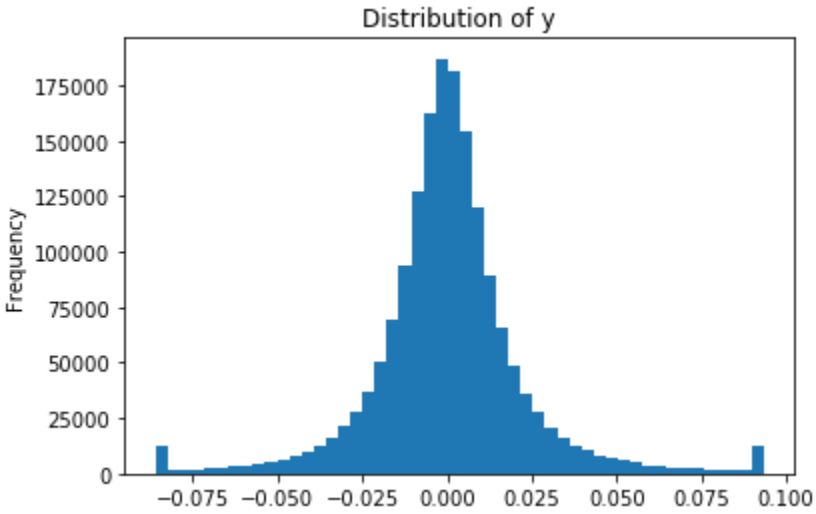


Figure 1: Distribution of asset returns

	count	mean	std	min	25%	50%	75%	max
y	1.7E+06	2.2E-04	2.2E-02	-8.6E-02	-9.6E-03	-1.6E-04	9.5E-03	9.3E-02

Table 2: Description of target variable 'y'

As mentioned in the previous section we will split the data set such that first 75% of the data points will be used to train and cross validate the models while the last 25% will be kept as the testing set. Figure 2 shows the distribution of y in training and testing sets respectively. It can be seen that both training and testing sets have similar y value distribution.

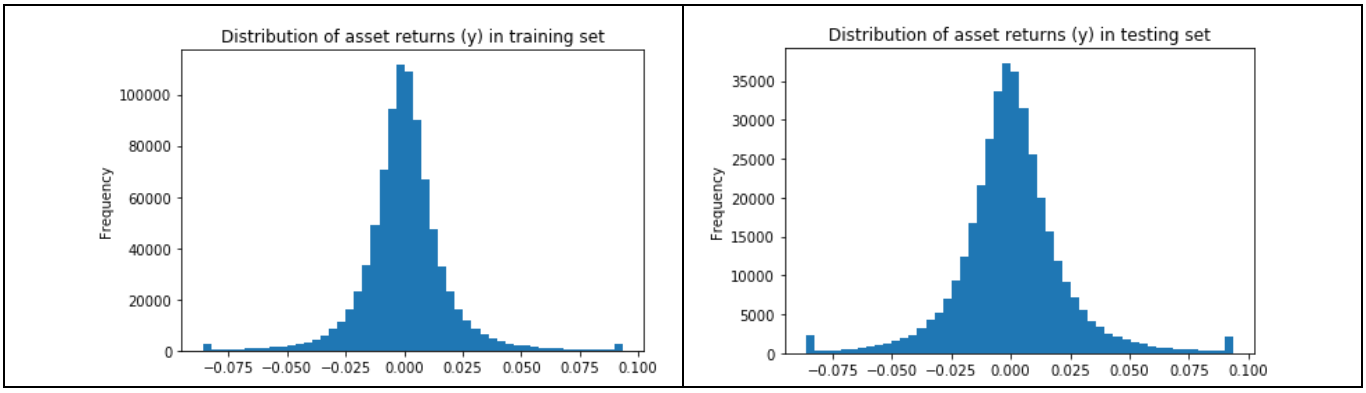


Figure 2: Distribution of target variable in train and test sets

As mentioned in the data exploration section some of the feature columns have highly skewed distributions as shown in Figure 3.

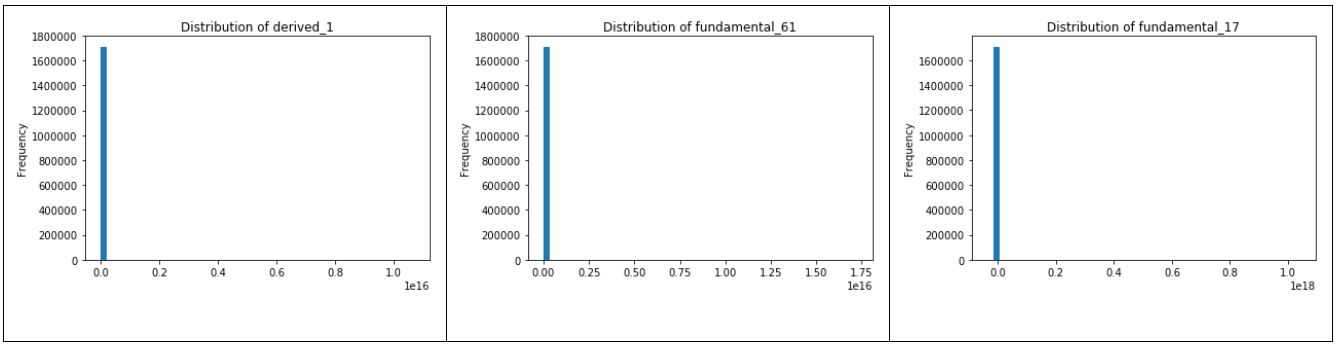


Figure 3: Distribution of highly skewed features

Algorithms and Techniques

Preprocessing: Following techniques and algorithms has been used in preprocessing the data.

Missing values: missing values in the data set are replaced by the median value of their respective columns using sklearn Imputer module. It was used so that median values from the training set can be used to fill the missing values of both train and test sets.

Feature selection: feature selection was done using univariate selection (sklearn: SelectKBest), and tree based methods (sklearn: ExtraTreeRegressor feature_importances_). In SelectKBest, I used f_regression model in which the correlation between each regressor and the target is computed and converted to an F score then to a p-value. In ExtraTreeRegressor, feature importance is calculated by averaging the total decrease in node impurity across all the trees in the ensemble. [6, 7]

After selecting the relevant features, multiple regression models were fitted to the training data using linear regression and Ridge regression modules in scikit-learn as the starting point, and evaluate their performance against the benchmark model. These regression models were selected as the starting point considering their simplicity and high noise levels in the input data. Linear regression module utilizes ordinary least squares linear regression while in Ridge regression l2-norm was used in the loss function.

Then moving on to the next step several tree based regression models (Random forest, AdaBoost and Extra trees) and a Support Vector Regression model was developed and their parameters were tuned using k- fold cross validation and grid search. In Random forest regressor module, multiple decision trees are fitted on sampled data set and output is averaged to improve the predictive accuracy and control over-fitting. In AdaBoost regressor, decision trees are fitted sequentially by improving the error of the current model, so that each new tree generated focus more on difficult cases based on the current model. Extra trees model utilizes a random splits in the trees making it computationally faster. SVMs use kernels to maximize the decision margin and SVR extend this to regression problems. I used the

Epsilon-Support Vector Regression module in sklearn with available ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’ kernals. SVR was selected as a trial mode, due to its reputation to have good accuracy.

These supervised learning regression models were chosen considering their simplicity, training time and applicability in the domain.

Benchmark

Asset return values in the dataset have been undergone some sort of normalization and capping process. This can be seen by the bell shaped distribution of return values around zero and two spikes at the end of the tails. Average of the return does not show a significant drift over time, thus allowing us to use it as a benchmark for the model. In this project we will use average return of all the assets over entire training dataset as the benchmark for testing data set. R score obtained using mean y value of the training set as the estimated value for test set was -0.012877.

Benchmark R score	-0.012877

Methodology

Data Preprocessing

Following steps were carried out in data preprocessing:

- test train split
- missing values replaced
- Handling large feature values
    - Large outliers removed
    - non-linear transformation

Test train split:

Test train split was carried out using sklearn test_train_split module with no shuffle to eliminate the look-ahead bias (data was sorted to timestamp)

Missing Values replaced:

Missing values were replaced using sklearn Imputer module. First the imputer object was trained on the training data and transformed them by replacing missing values with respective column medians. Then the test set was transformed using the previously trained Imputer object.

Handling large feature values:

As mentioned before, very large range and skewed distributions in some of the features result in bad models being trained on the data set. Following methods were used to overcome this.

- Outliers removal
- Outliers clipping
- Non-linear transformation

Removing outliers:

- Plot histograms for all the feature columns.
- Manually identify the suitable cutoff range
- Remove the data outside the identified cut off range.

Figure 4 shows histograms of several features in training data before and after the removal of outliers.

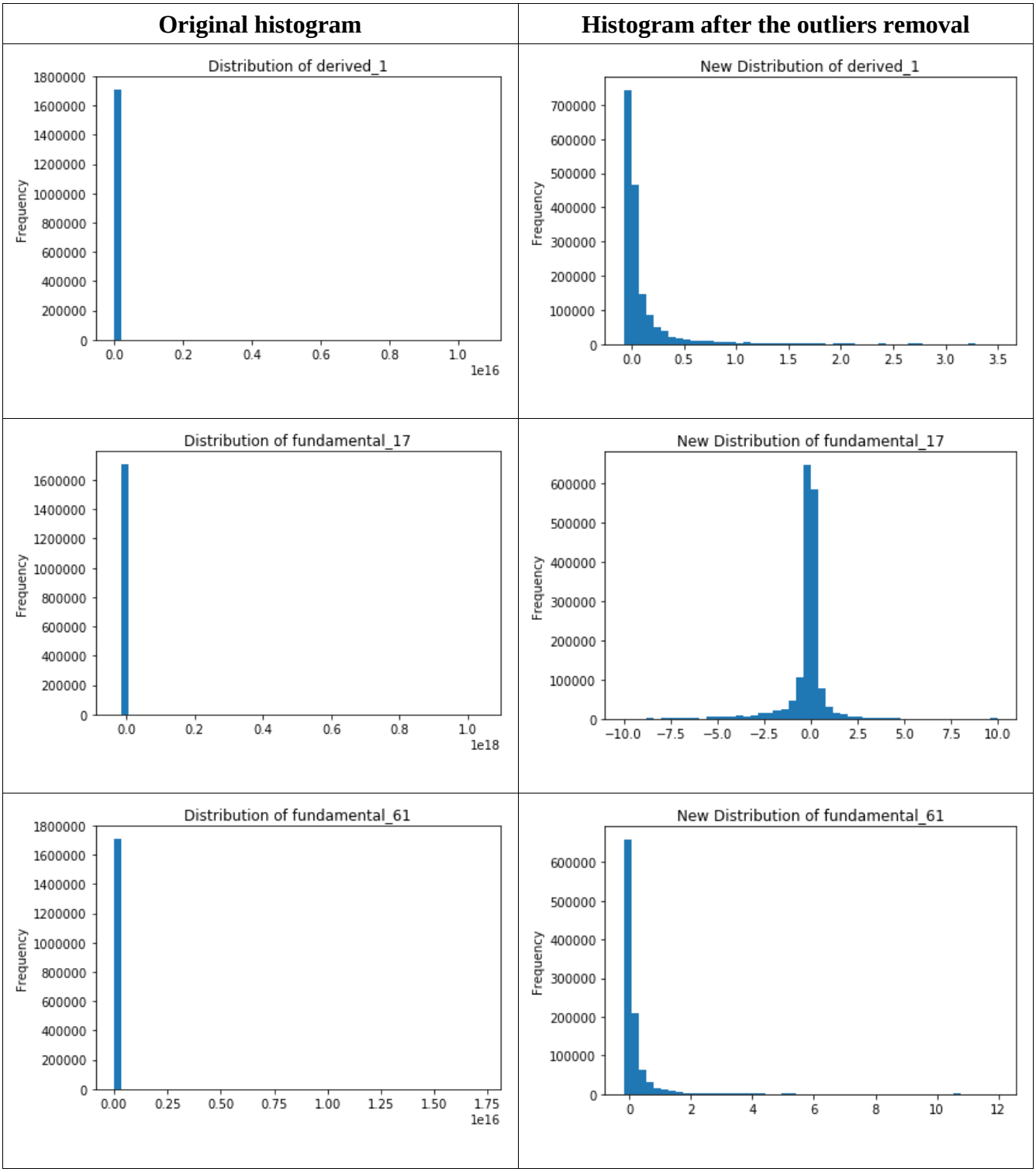


Figure 4: Distribution before and after outlier removal

Outliers clipping:

Instead of removing the outliers they are clipped at predefined values. Figure 5 shows histograms of several features in training data before and after the clipping of outliers.

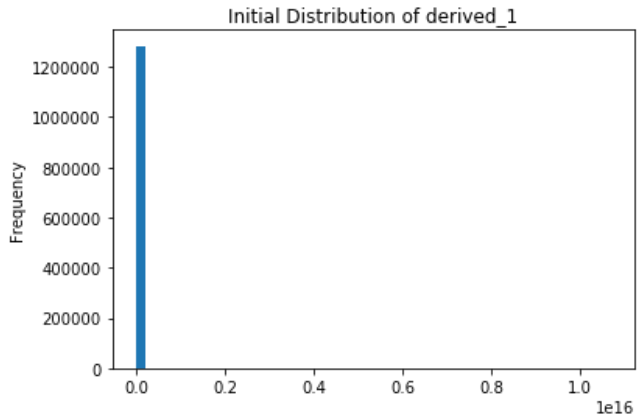
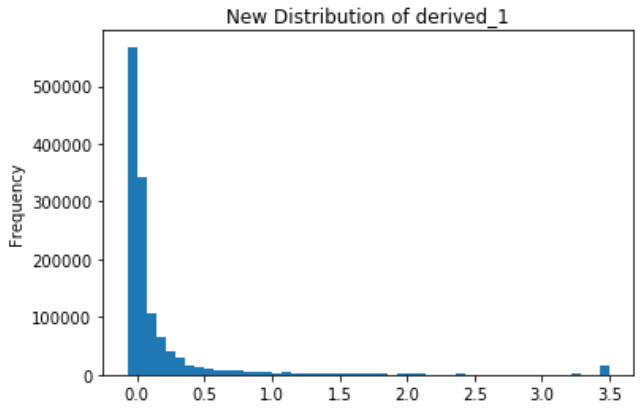
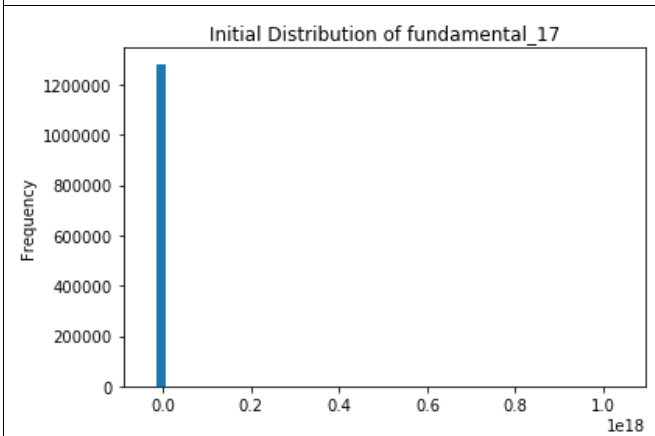
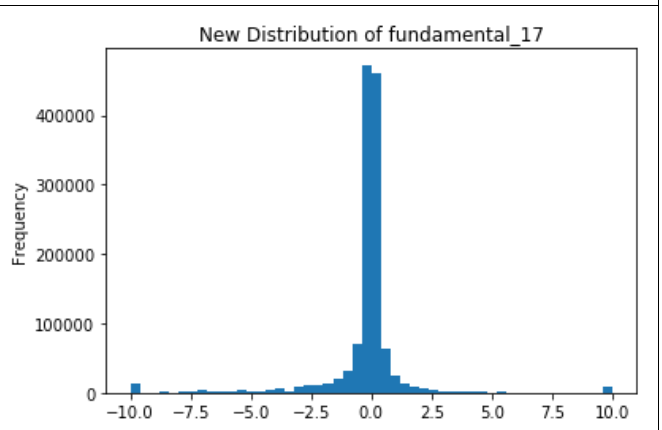
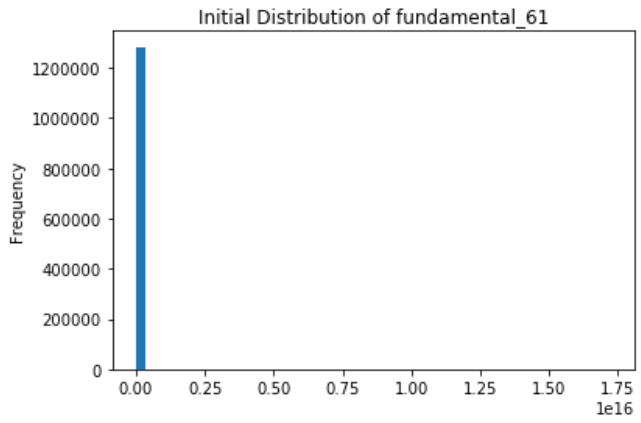
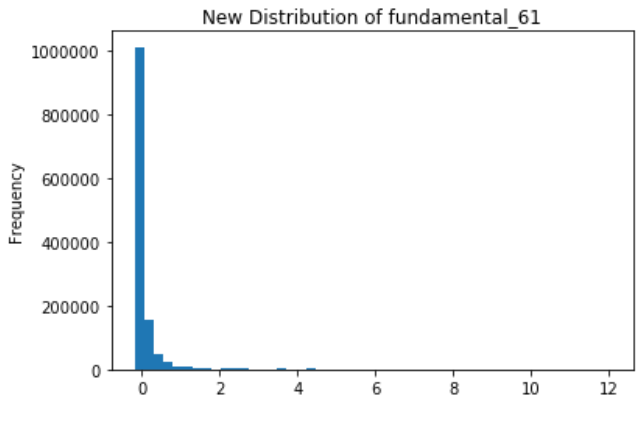
Original histogram	Histogram after the clipping outliers
	
	
	

Figure 5: Distribution before and after clipping outliers

Nonlinear transform:

Nonlinear transformation was carried out using sklearn QuantileTransformer module. Figure 6 shows the histograms before and after the transformation to uniform distribution.

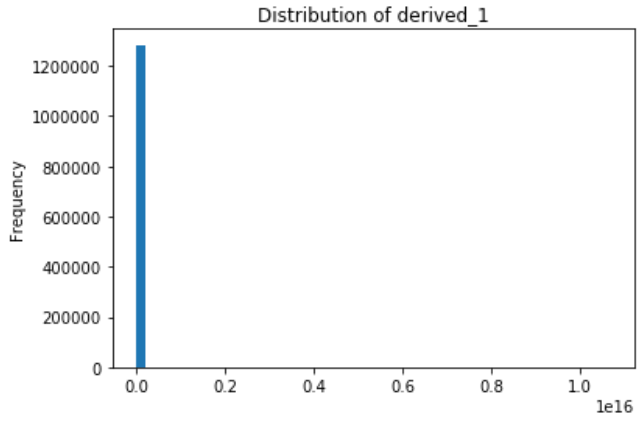
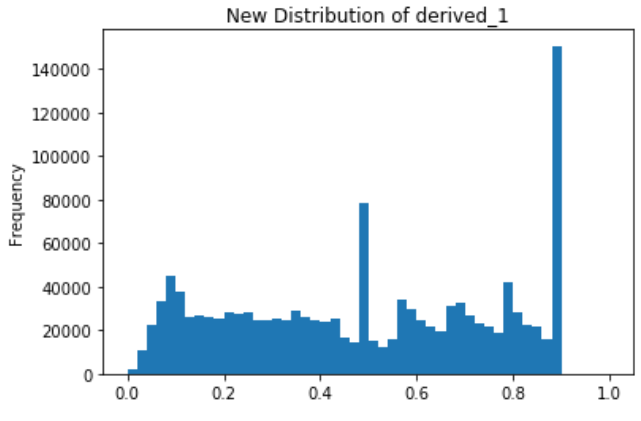
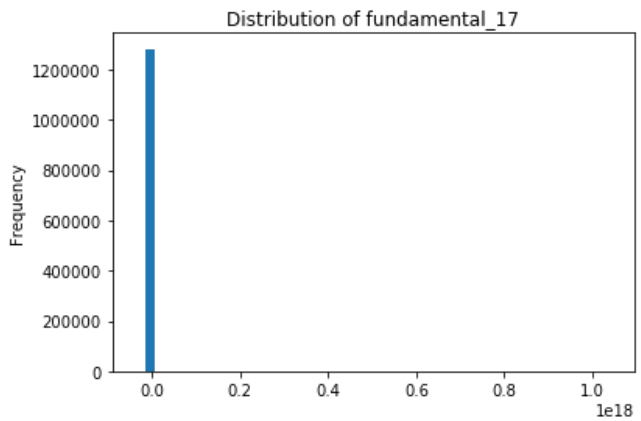
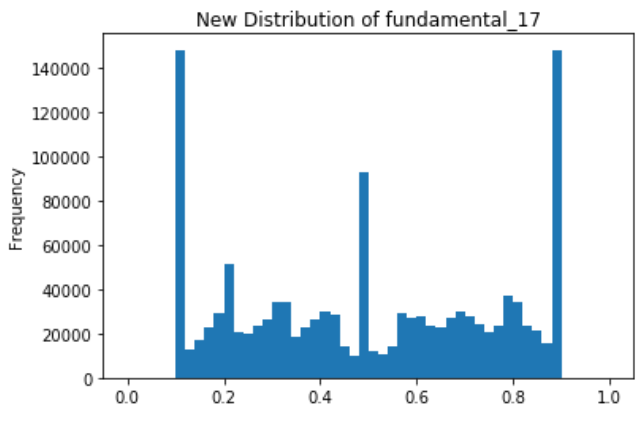
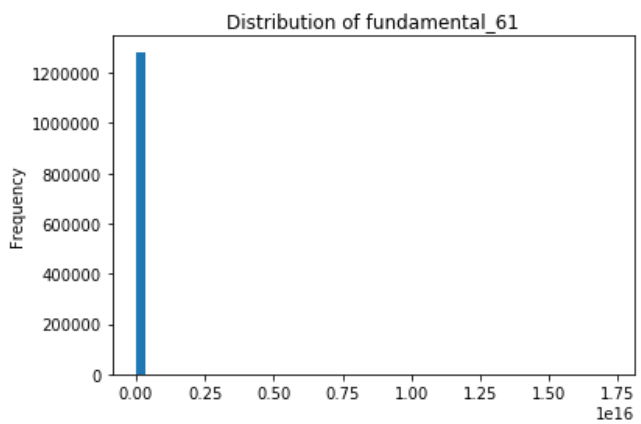
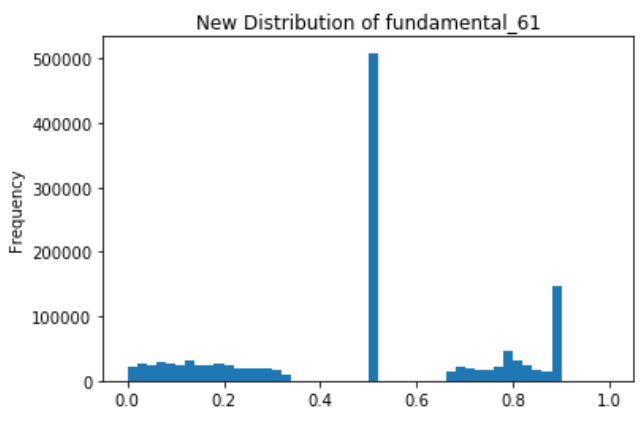
Original histogram	Histogram after the transformation to uniform distribution
	
	
	

Figure 6: Distribution before and after transforming to uniform distribution

Figure 7 shows the histograms before and after the transformation to normal distribution using QuantileTransformer module.

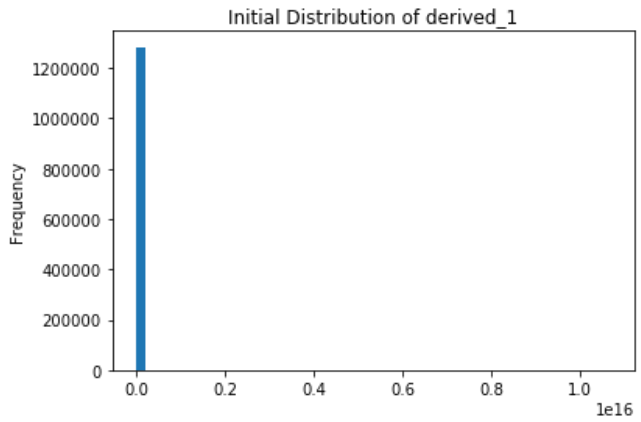
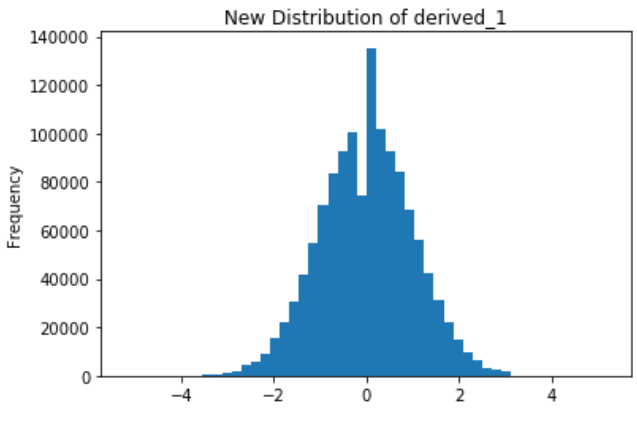
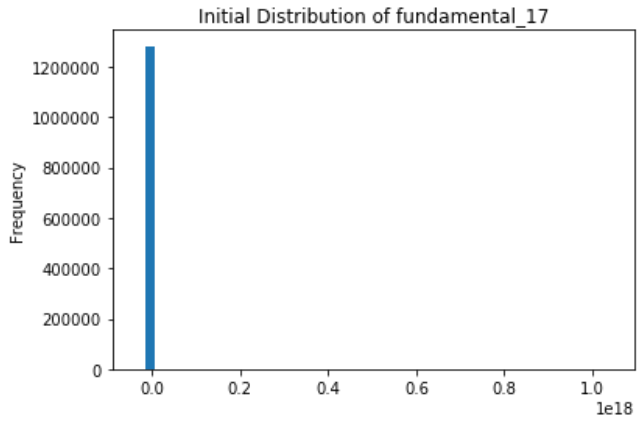
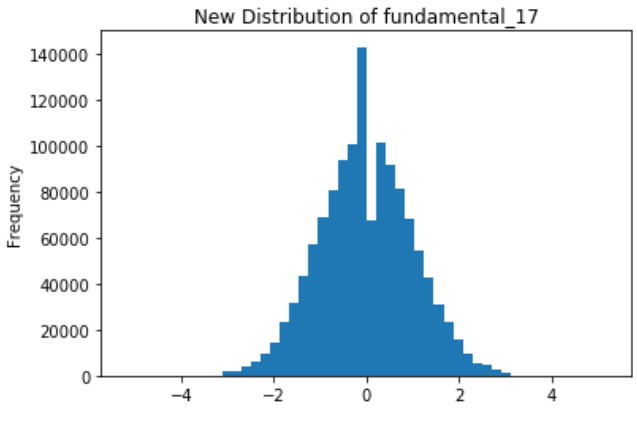
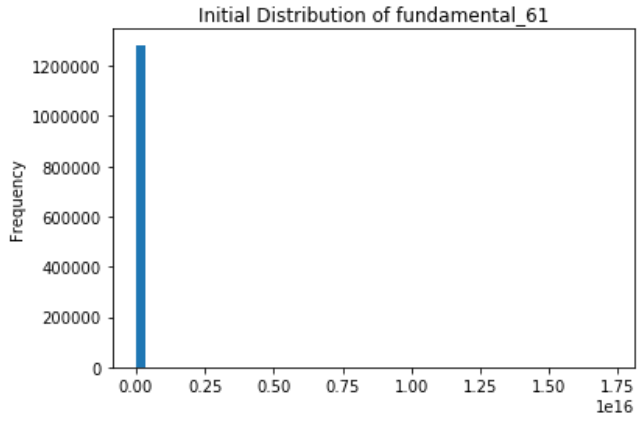
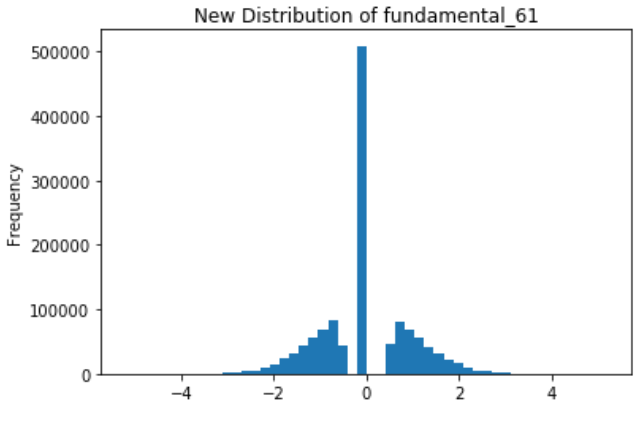
Original histogram	Histogram after the transformation to normal distribution
	
	
	

Figure 7: Distribution before and after transforming to normal distribution

Implementation and Refinement

In this section we will implement several models and refine them for better performance.

Following libraries were used in the implementation process:

- r_score module from sklearn.metrics was used to calculate the R^2 score for predictions.
- cross_val_score module from sklearn.model_selection was used to do the cross validation.
- GridSearchCV module from sklearn.model_selection was used to do the Grid search for parameter optimizations in Ridge regression, Random forest and SVR along with cross validation.
- LinearRegression module from sklearn.linear_model was used to create the multiple linear regression models.
- Ridge module from sklearn.linear_model was used to create the multiple Ridge regression models.
- RandomForestRegressor module from sklearn.ensemble was used to create the random forest regression model.
- AdaBoostRegressor module from sklearn.ensemble along with DecisionTreeRegressor module from sklearn.tree was used to create AdaBoost model.
- ExtraTreesRegressor module from sklearn.ensemble was used to create the Extra-trees regression model.
- SVR module from sklearn.svm was used to create the Epsilon-Support Vector Regression model.

Implementation code can be found in provided four Implementation_* notebooks. Long computation time was taken to cross validate the RandomForestRegressor, increasing the number of cores used (n_jobs) was greatly helpful.

First we will train multiple linear regression and Ridge regression models with all the features. Table 3 shows the results with 3 fold cross validation on training set.

Model parameters	**Outliers removal**	**Outliers clipping**	**Transform-uniform**	**Transform-normal**
Linear	-0.025667	-0.026117	-0.026684	-0.028783
Ridge, alpha=0.1	-0.025564	-0.026068	-0.026684	-0.028783
Ridge, alpha=1	-0.025026	-0.025499	-0.026683	-0.028783
Ridge, alpha=100	-0.026878	-0.025838	-0.02655	-0.027756
Ridge, alpha=10000	-0.023014	-0.02312	-0.02207	-0.028126

Table 3: Linear and Ridge regression R scores with all the features.

Next step we will select a set of features to improve the R score. Following methods were used to select the features.

- Univariate selection using sklearn SelectKBest f_classif score function
- Tree based methods: sklearn ExtraTreeRegressor feature_importances_

SelectKBest:

Table 4 shows the 10 features with highest score calculated by SelectKBest module in sklearn. It can be seen that the rankings are as same as the correlation method.

Rank	Outliers removal	Outliers clipping	Transform-uniform	Transform-normal
1	technical_20	technical_20	timestamp	timestamp
2	technical_30	technical_30	technical_20	technical_20
3	timestamp	timestamp	fundamental_55	fundamental_55
4	fundamental_55	fundamental_53	technical_30	fundamental_11
5	fundamental_53	technical_40	fundamental_11	technical_30
6	technical_19	fundamental_11	fundamental_56	fundamental_2
7	fundamental_8	technical_7	fundamental_2	fundamental_56
8	fundamental_56	fundamental_48	fundamental_53	fundamental_53
9	fundamental_60	fundamental_56	fundamental_48	fundamental_48
10	fundamental_11	fundamental_55	technical_40	technical_40

Table 4: Features with highest score calculated by SelectKBest module

Feature importance from extra tree regressors:

Table 5 shows the feature importance ranked by ExtraTreesRegressor module in sklearn.

Rank	Outliers removal	Outliers clipping	Transform-uniform	Transform-normal
1	technical_30	technical_33	technical_33	technical_33
2	technical_33	technical_30	technical_41	technical_41
3	technical_41	technical_41	technical_30	technical_30
4	technical_24	timestamp	technical_24	technical_24
5	technical_20	technical_24	technical_3	technical_3
6	technical_3	technical_3	timestamp	technical_1
7	timestamp	technical_20	technical_1	technical_5
8	technical_1	technical_1	technical_20	technical_31
9	technical_5	technical_5	technical_5	technical_20
10	technical_13	technical_31	technical_31	technical_25

Table 5: Feature importance ranked by ExtraTreesRegressor module

After several cross validation trials following set of features were selected to continue with the models.

technical_20	technical_30	fundamental_11	technical_19	technical_24	id

Table 6 shows the cross validated R scores for linear and ridge regression models built with selected features.

Model parameters	**Outliers removal**	**Outliers clipping**	**Transform-uniform**	**Transform-normal**
Linear	-0.013704	-0.017387	-0.019312	-0.019981
Ridge, alpha=0.1	-0.013499	-0.017236	-0.019311	-0.019980
Ridge, alpha=1	-0.012722	-0.016331	-0.019311	-0.019980
Ridge, alpha=100	-0.016279	-0.017207	-0.019300	-0.019979
Ridge, alpha=10000	-0.016272	-0.017326	-0.018347	-0.019906

Table 6: Cross validated R score for linear and ridge regression models built with selected features

Next we built several other supervised learning models, and the results are tabulated in the Table 7. For Random forest and SVR models grid search with cross validation has been carried out to optimize the parameters and only the results from optimum model is shown in the Table 7.

Model parameters	**Outliers removal**	**Outliers clipping**	**Transform-uniform**	**Transform-normal**
Random forest	-0.005577	0.0009751	-0.0121511	-0.01190
AdaBoost	-0.049778	-0.019066	-0.058134	-0.041316
Extra trees	-0.611889	-0.496716	-0.62663	-0.495346
SVR	-0.212673	-0.195435	-0.195435	-0.195435

Table 7: R score for tree based regression models SVR built with selected features

From the above results we can see that Random forest regressor with outliers clipping is the optimum model with cross validated R score of 0.0009751. In order to compare with the other supervised learning methods we will evaluate best models obtained from linear and Ridge regression methods as well.

Results

Model evaluation, validation and Justification

In order to validate the robustness of the model with unseen data we use our test set. Test set was separated from the training set initially before building the models. Table 8 shows the Test set R score for selected 3 best models.

Model	**Test set R Score**
Random forest	0.01731
Linear	0.02122
Ridge	0.02016

Table 8: R score for test set

From the above results it can be seen that the selected models perform well with testing data. Good test scores also suggest that the final parameters obtained by grid search for random forest and Ridge regression models generalize well for unseen data. As observed from the grid search cross validation results for random forest regressor shown in Table 9, when increasing the max depth, R score decreases. This might be a result of overfitting due to high noise in the data.

Max depth	Cross validated R score
2	-0.0055772
4	-0.011345
6	-0.023468

Table 9: grid search cross validation results for random forest regressor with n_estimators=50

All three selected models perform well compared to the benchmark R score of -0.012877. Considering the very high noise to signal ratio in financial data it is safe to say that our three selected models perform satisfactorily within the given domain.

Conclusion

Free-Form Visualization

Considering the high volatility of the financial data our expectations are quite focused on predicting the fluctuation of the market rather than the actual return value. Figure 8 shows several comparison figures where our linear model was succeeded in doing this.

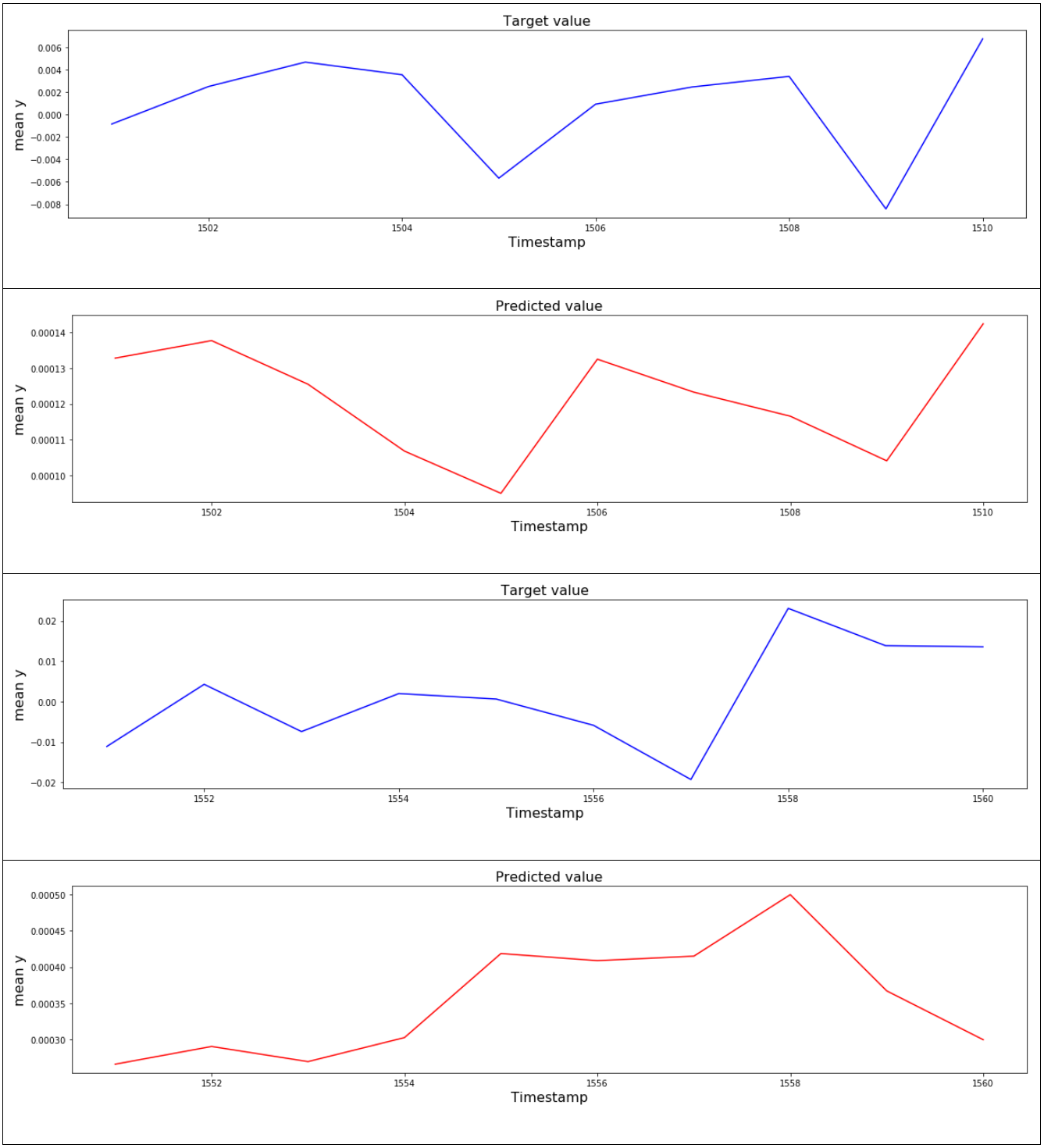


Figure 8: Trend of target ‘y’ values and predicted ‘y’ values

Reflection

End to end process followed in this project is listed below.

- Data downloaded and preprocessed
    - Data downloaded from Kaggle 2sigma competition web page.
    - Split the data in to test and train sets
    - missing values replaced using Imputer
    - Handling large feature values
        - Large outliers removed or clipped
            - Plot histograms for all the feature columns.
            - Manually identify the suitable cutoff range
            - Remove the data outside the identified cut off range.
        - Apply Non-linear transformation
            - QuantileTransformer: uniform distribution
            - QuantileTransformer: Normal distribution
- Feature selection
    - Univariate selection using sklearn SelectKBest
    - Tree based methods: sklearn ExtraTreeRegressor feature_importances_
    - Manually optimize the features by cross validation and linear regression.
- Implementation
    - Fit multiple linear regression models.
    - Fit multiple ridge regression model with grid search cross validation for alpha parameter.
    - Fit random forest model with cross validation and grid search for maximum depth and number of estimators.
    - Fit Ada boost and Extra trees models with cross validation
    - Fit SVR model with cross validation and grid search.
- Select three best models, evaluate its performance for test data and compare with benchmark.

The most interesting aspect of the project for me is the surprisingly good performance from linear model compared to other more complex ones, given the high noise to signal ratio in the data. I found it

difficult or rather complex to figure out the optimum features. Having 110 features, some with very high range and skewed distributions made it difficult to determine the best features. The final model, even with its ability to predict the trend, still requires improvements before applying in the real world scenarios.

Improvement

Current model can be improved further by utilizing more complex models such as deep neural nets (hardware power) and optimized features. More Information on the features such as their meaning and how they were obtained will certainly help optimize the model performance by giving much needed insight in to better utilizing the features. Financial market domain knowledge combined with the information on how the data was collected and features were created and scaled will be a good starting point to further improve the model.

Reference

[1] D. E. D. Silva and J. C. Duarte., "Prediction of assets behavior in financial series using machine learning algorithms.," *International Journal of Advanced Research in Artificial Intelligence*, vol. 2, no. 11, pp. 46-52, 2013.

[2] T. Fletcher, "Machine learning for financial market prediction.," University College London, 2012.

[3] X. Qian and S. Gao, "Financial Series Prediction: Comparison Between Precision of Time Series Models and Machine Learning Methods.," 2017.

[4] E. Z. R. M. S. E. Zankova, "High Frequency Financial Time Series Prediction: Machine Learning Approach.," 2016.

[5] "kaggle," [Online]. Available: https://www.kaggle.com/c/two-sigma-financial-modeling/data.

[6] P. P. Gilles Louppe, "stackoverflow," [Online]. Available: https://stackoverflow.com/questions/15810339/how-are-feature-importances-in-randomforestclassifier-determined.

[7] F. Breiman, Classification and regression trees, 1984.