Machine Learning Research Project

Daniele Prevedello, Federico Panariello

1 Introduction

Our project tackles a very difficult task of predicting stock price, with unknown masked feature and certain historical data. This is a past Kaggle competition. https://www.kaggle.com/competitions/the-winton-stock-market-challenge

The data set involves the return of stocks, given the history of the past few days. We are provided with 5-day windows of time, days D-2, D-1, D, D+1, and D+2. We are given returns in days D-2, D-1, and part of day D, and we want to predict the returns in the rest of day D, and in days D+1 and D+2.

During day D, there is intraday return data, which are the returns at different points in the day. We are provided with 180 minutes of data, from t=1 to t=180. In the training set we are given the full 180 minutes, in the test set just the first 120 minutes are provided, and we are asked to predict the next 60 minutes, and the daily return on D+1 and D+2.

For each 5-day window, we also have 25 masked features, which may or may not be useful in the prediction. Each row in the dataset is an arbitrary stock at an arbitrary 5 day time window. There are 40000 rows in the training set, and 120000 rows in the test set.

The loss function is the weighted mean absolute error (WMAE). Upon examining that data, the weights assigned seems to be a random number between 10^6 to 3×10^6. The mean is 1.5046×10^6 and the standard deviation is 2.059117×10^5, so we can assume that the randomly assign weight is not so different each row, and to train our model we only need to optimize the ordinary mean absolute error.

2 Data Analysis

We will analyze the features before we start to fit our models.

2.1 Missing values

In the training set, we see that most features have missing values except for features 5 and 7 (See table 1). Especially for feature 1 and feature 10. Out of 40000 entries, 33313 entries are NaN for feature 1, and 19471 entries are NaN for feature 10. The other columns have NaN counts that range from 469 to 9146. We suspect that the features with a significant amount of NaN values could be categorical, and NaN would correspond to a unique category.

Table 1: Missing Values in Feature Columns

Feature	Missing Values	Feature	Missing Values
Feature_1	33313	Feature_14	728
Feature_2	9146	Feature_15	2141
Feature_3	1237	Feature_16	610
Feature_4	7721	Feature_17	646
Feature_5	0	Feature_18	568
Feature_6	1933	Feature_19	1190
Feature_7	0	Feature_20	7826
Feature_8	469	Feature_21	1018
Feature_9	1875	Feature_22	1345
Feature_10	19471	Feature_23	1711
Feature_11	987	Feature_24	726
Feature_12	1096	Feature_25	655
Feature_13	594		

2.2 Identify Categorical variables

Next, we will identify the categorical features and numerical features. To do that, we first identify all the features with less or equal to 50 unique values. These should be highly likely to correspond categories. 8 out of 25 of the features have fewer than 50 unique values. We then plot the histogram of these features.

Interestingly, the histogram of feature 9 shows a strong bell shape, and figure 13 shows a weak bell shape as well (see figure 1 and figure 2). The observation provide evidence that these features could be regarded as numerical or ordinal. So instead of considering features 9 and 13 as ordinary categorical features to do one-hot coding, we move feature 9 to the numerical features and apply an ordinal encoder for feature 13.

As a summery: Categorical features include `Feature_1`, `Feature_5`, `Feature_8`, `Feature_10`, `Feature_13 (Ordinal)`, `Feature_16`, and `Feature_20`.

Numerical features include `Feature_2`, `Feature_3`, `Feature_4`, `Feature_6`, `Feature_7`, `Feature_9`, `Feature_11`, `Feature_12`, `Feature_14`, `Feature_15`, `Feature_17`, `Feature_18`, `Feature_19`, `Feature_21`, `Feature_22`, `Feature_23`, `Feature_24`, and `Feature_25`.

2.3 Numerical variables

Next, we visualize the numerical variables (See figure 4). There are several observation to make.

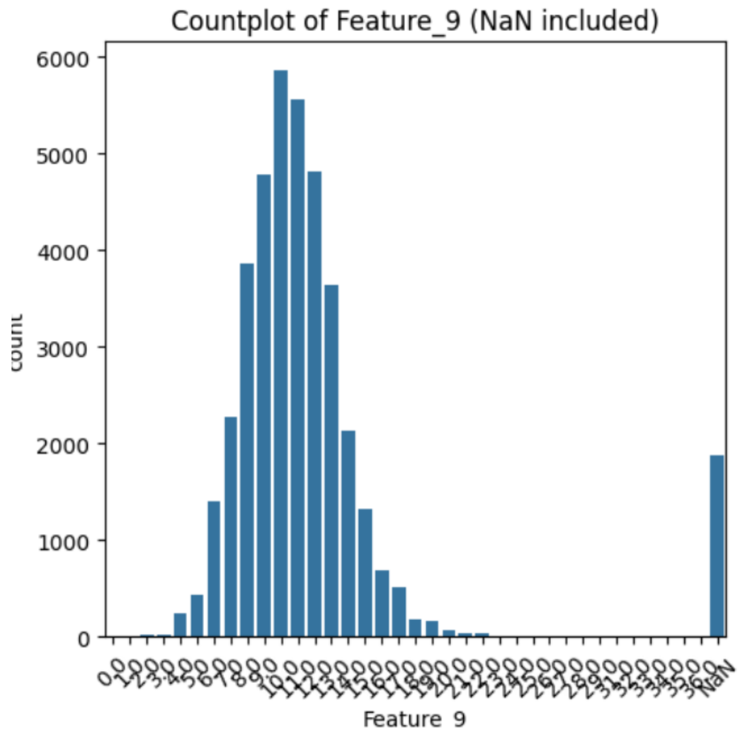

Figure 1: Histogram of feature 9

1. Except for feature 7 and 12, all the other features resemble a bell-shape curve. This is a good indicator that these variables are numerical and resemble a Gaussian distribution.

2. For features 2,17, 18, 21, 25, the fitted curve has a bimodal shape. That indicates that the data could come from sampling in two Gaussian distributions. To deal with this, we apply the k-means algorithm with $k = 2$ to find two clusters of these features. The cluster will then be added to the categorical features for more insight.

3. The bell curves for features 3,6,11,21 seems to be skewed. Taking the logarithm of these features, therefore, is appropriate to recover an underlying Gaussian distribution. In detail, since feature 11 is skewed to the left, we first take the minus of the distribution of feature 11 and then take the logarithm. Whereas the other features (3,6,21) are all right-skewed, so we directly take the logarithm. As a sanity check after the transformation, their histogram resembles a Gaussian distribution (See figure **??**).

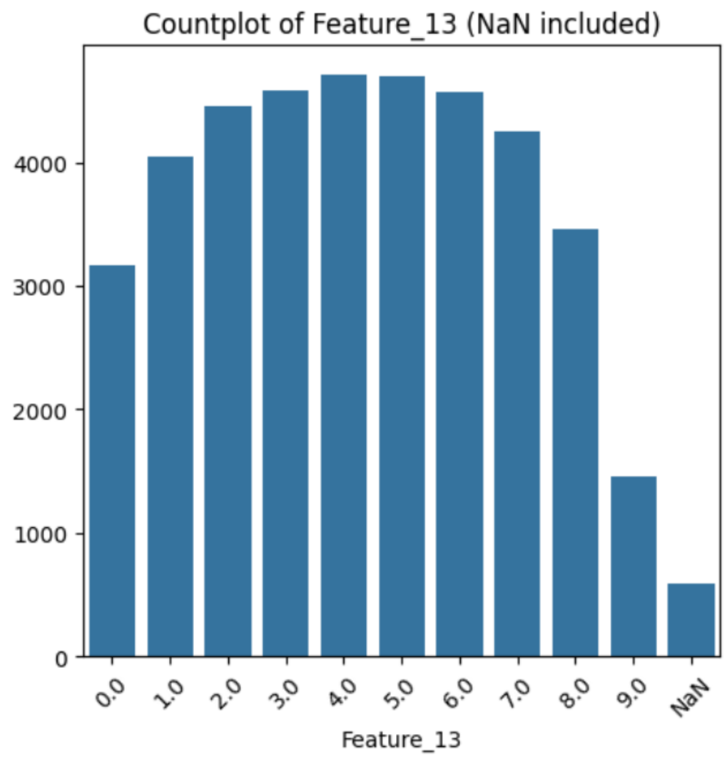

Figure 2: Histogram of feature 13

2.4 Normalization and encoder

Since most of the numerical features are approximately Gaussian, we apply the standard scaler instead of the maxminscaler.

As we explained above in the Categorical features, we apply one-hot encoder for all the categorical features (including the cluster features added by k-Means), except for feature 13, for which we apply an ordinal encoder. Combined with the cluster feature provided by k-Means, there are 92 categorical features in total.

3 Linear regression

We first try linear regression as a benchmark for all machine learning models. Finance data is known to be notoriously noisy, and indeed in a lot of cases a linear model with good feature engineering will outperform advanced nonlinear models.

After feature engineering, we have three different kinds of features on the training and test set. Among the 25 masked features, we identify 7 of them as

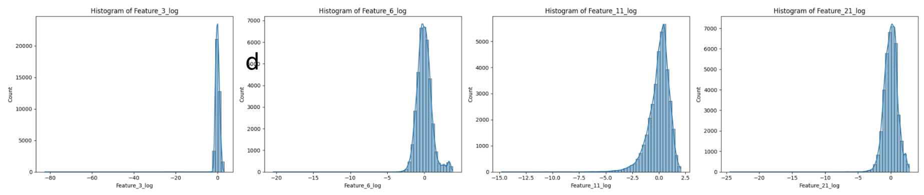

Figure 3: Histograms of log transformed features

categorical and 18 of them as numerical, which satisfy some kind of distribution. After one hot encoding, ordinal encoding and k-means preprocessing, we have 92 features. We will explain below different attempts we have made regarding treat these features during training.

3.1 Random walk benchmark

Before training any model, we consider the assumption that the price movement is similar to a random walk, or Brownian motion to be more precise. Therefore our benchmark should be to fill zero as return for all our prediction. The all zero submission has score 1728.62346 on the private leaderboard. This score actually ranks the 371st among all 830 teams, which means that more than half of the teams have prediction worse than simply putting zero everywhere. This is another indicator that the data is extremely noisy, and people will very easily tends to overfit their model. As we will show, it is actually very difficult for a model to beat this naive martingale benchmark.

3.2 Ordinary linear regression

We first use all the available features(categorical, numerical and time series) to predict the target variables, including the 60 intra-day target variable and the two inter-day target variable. Our resulting submission has exactly the same score as the all-zero benchmark. Upon examining the data, we find that all the prediction are very small relatively, on the order of 10^{-16}. After some trial and error, we identify the problem to be that with such a large data set, after one hot encoding, there are too many features, and the gradient descent algorithm for OLS does not converge within the max iteration limit, since the OLS program is based on gradient descent instead of theoretically find the inverse of a very large matrix.

In order to solve this problem, we dropped all the categorical columns and only run regression on the numerical and time series ones. This time we obtain meaningful output at a reasonable order. However the score= 1821 is even worse. We will see that one difficulty of this project is that, a large prediction will likely cause a very big error, thus it becomes very hard to beat the all-zero filling, where the errors is controlled by the magnitude of solution themselves. Furthermore, this shows that the data set is extremely noisy.

Additionally we consider the following strategy: predicting the stock price every minute is an extremely noisy and difficult task. Instead we should just focus on predicting the daily return for the next two days, and submit zero for the intra-day prediction. We will employ this strategy for all the following models.

3.3 Ridge, LASSO and elastic net

To prevent overfitting and in order to find the truly relevant features, we tried different penalty function. We use ridge regression with L^2 penalty, LASSO with L^1 penalty, and elastic net with a mixture of both. We normalize the features which is the necessary step for these regression. We also use 5-fold cross validation to identify the best hyperparameter alpha. Interestingly, the best alpha for cross validation tends to be small (0.01 over 0.1 or 1). However on the test set, alpha with 0.1 or above actually outperform the former. The test score is 1728.43412. Upon examining the data we find out the reason: with large alpha, actually all features are forced to be zero. Therefore we are only submitting the intercept, which is the best constant on the training set. This will outperform all-zero prediction, which suggests that the market is not really a simple random walk, and mean-filling is better than zero-filling, which is a reasonable assumption.

We also made several attempts to adjust our feature engineering, since the problem of convergence on a large data set is apparent. We try to only feed the models with numerical features, only feed them with numerical features plus time series, or refine the time series parameters: take the return of D-2 and D-1, take the sum of all 120 minutes and treat this as the third features, then run regression of these three together with 18 masked numerical features. For all these attempts the resulting score is not better than filling with intercept, and a number of them is worse than zero-filling. This again validate that the data is quite noisy, and real pattern from historical returns can hardly be seen.

As for these regression models themselves, LASSO and elastic net outperform Ridge, which is expected. Ridge only shrink the coefficient while LASSO makes irrelevant features zero, which reduce overfitting and autocorrelation significantly.

Additionally, we try support vector machine model, which performs worse on test set comparing with simple regression techniques. With noisy data usually simple model will be better than more complex ones, from point of view of convergence and overfitting.

4 CART models

We tested the following four tree-based model: Random Forest regressor, lightgbm regressor, xgboost regressor, catboost regressor. All of them are standardized with the number of estimators equal to 100. The models are first trained for with all the features (numerical features, categorical features and the return for the first 120 minutes) and then for comparison they are trained with only numerical features. Based on our results, we can make the following observation:

1. Out of all the models, CatBoost with numerical features only has the highest score(1728.33846). Interestingly enough, it outperforms the CatBoost with full features.
2. Lightgbm with numerical features and without numerical features seems to collapse. It predicts the same value $(-7.0457E-05)$ for almost all entries for D+1 and the same value(-0.000105) for almost all entries for D+2.
3. XGBoost doesn't collapse, but it has the same prediction with numerical features or with all the features.

We tried to figure out the reason. It seems that the full features, including those from one-hot coding is very sparse. As the tree built by CatBoost is symmetric, it may lead to overfitting. So maybe running CatBoost on engineered features may not be the best idea, the model probably performs better without preliminary feature engineering. The reason is that LightGBM predicts a constant whenever no split passes its gain threshold. The model probably stopped learning.

5 Neural network

Although the dataset is large and noisy, we made some attempt of using nonlinear models such as neural networks. For time-series like problem like this, naturally we start with LSTM for sequential data prediction. We also tried transformer models. Due to our computing power and the fact that the data set is extremely noisy and large, these nonlinear models cannot outperform simple linear models or even zero-filling. Neural networks cannot even converge within a reasonable time, thus resulting a very large error since the prediction value is very large. For 15 epoch, we have the loss around 300 for LSTM, and for transformer. Due to slow convergence, the predicted return per minutes keeps at the magnitude of $10^{-2}-10^{-1}$, while the actual order should be around 10^{-6}.

6 Discussion and improvement

After trying all these models (regression based, tree based, SVM and neural networks) and tweaking, we find out indeed it is a challenging and difficult project. The main difficulty lies in the huge noise in the data set, which not only makes a nonlinear model impossible, but also makes linear models hard to improve. Imagine if we have a uniform distribution on a unit disk for the sample data, running ordinary linear regression will give us very small coefficients and predictions, while the error size roughly depends on the target value themselves. Large data size also makes rate of convergence an issue in this project.

One potential direction to improve the performance would be better feature engineering. If we can find a key feature to group the rows together, for example, if the sign of the return is highly correlated to a certain feature, we can group the rows based on it and run linear models on each group, which we can expect to outperform our current approach.

One can also consider ensemble methods, to combine different models together hopefully for a better performance. Howvever with such noisy data, one need to find better models for benchmark before any ensemble attempt.

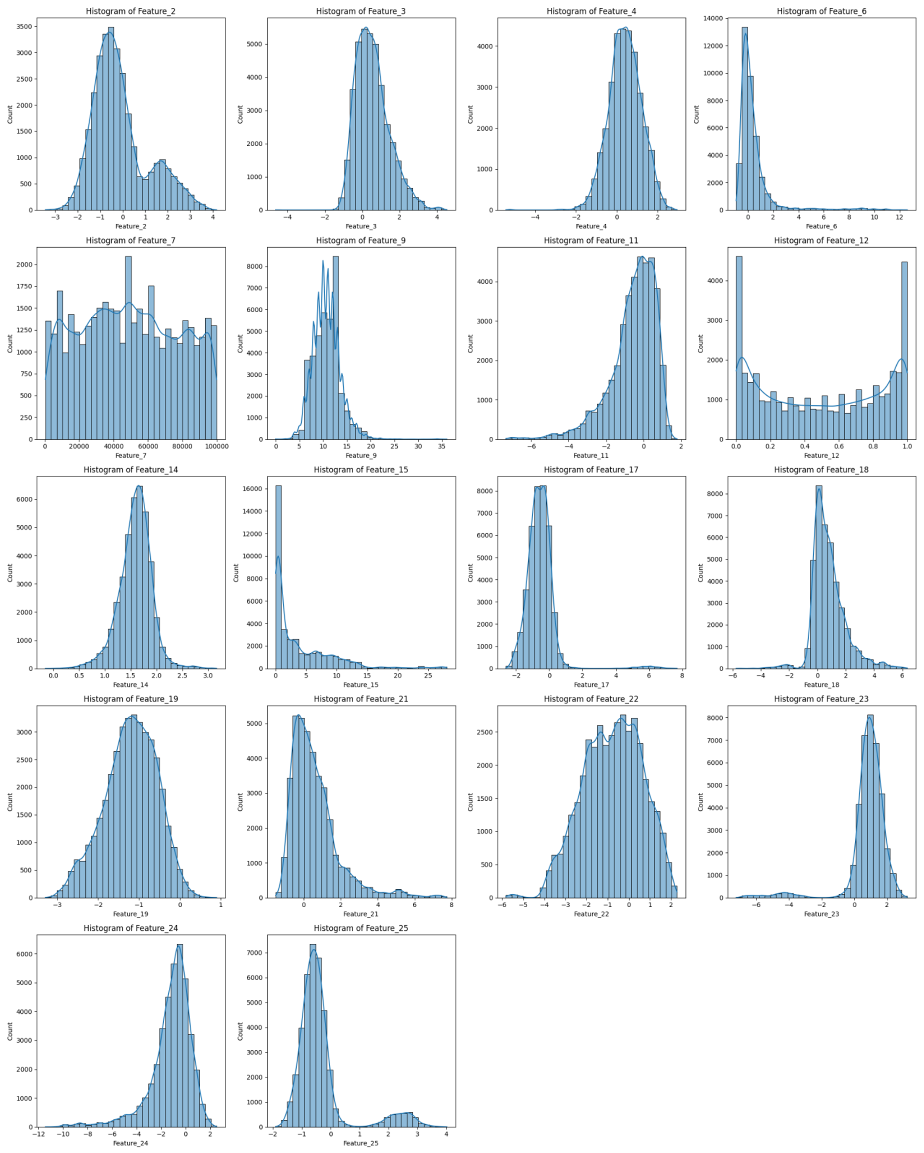

Figure 4: Histograms of numerical features