Methodology

Data Exploration

Based on Data Exploration, the columns were divided into 3 categories:

(i) Return type column: Columns with both -ve and +ve and range bounded, suggesting that the data is already normalized or contains return-type data
(ii) Price type column: Column with only positive and continuous values, like price data
(iii) Integer type columns: Column containing only integer value, probably containing information related to volume

Further, significant number of columns were very similar with correlation close to 1. These types of columns were removed while leaving the column with highest correlation only, leaving 42 columns. The distribution of columns were also considered by looking at Kurtosis and Skewness, serving a measure of their distribution compared to Normal Distribution

Data Preprocessing & Feature Engineering

Missing Values: The missing values in the given dataset were first forward filled to avoid lookahead bias.

Feature Engineering: As the metadata on each column is not available, certain assumptions were related to nature of data by categorizing features into 3 categories. The academic literature focused on using returns as covariates to predict the target variables, the price type and integer type columns were transformed into return types by finding percentage change in value from it's previous 24 hr value, to match the time frame with the target variable.

Feature Generation: Since the model template was restrictive, recursive training and predictions couldn't be performed. But, the to incorporate the serial nature of returns data, lag variables were generated both for return and other features generated following previous steps.

Outlier Treatment: From data exploration, it was clear that outliers were present in each feature. To avoid lookahead bias, a novel method was implemented where the rolling mean and standard deviation of past 20 days were calculated, and values above or below 2 standard deviations were clipped to preserve the information contained in them.

Model Training & Validation

Models: Due to lack of time, only linear models were considered, a better result could be obtained by including non-linear models, for example, neural network models. The models tested were: Linear Regression and Regularized Linear Models: Lasso, Ridge and Elastic Net

Data Design: The candidate dataset was split into train (80%) and test (20%). The test data was kept untouched and was used to test OOS forecasts. The train data is further split into train and test data for training the model in case of Linear Regression model and into train, cross validation and test data for train, tune hyperparameter and find optimal model in case of regularized models as they contain hyperparameters.

Cross Validation: As the model template required the data to be trained on single dataset and output the optimal model, the recursive cross validation couldn't work. Additionally, k fold validation would have failed owing to the potential serial correlation among the dataset leading to lookahead bias. Thus, a simple cross validation routine was considered where the originally built train dataset was split into train, cross validation and test dataset and the hyperparameter tuning was performed over cross validation dataset once they were trained on train dataset.

Optimal Model Selection: To avoid any human bias and errors, the whole process was automated. Following optionality were added in the code so that the most optimal model can be selected with least bias:

1. Feature Engineering: Added optionality to perform feature engineering or not to check whether the assumption comes to fruition on this dataset
2. Outlier Treatment: Added optionality to perform outlier treatment or not
3. Lag X variables: Whether to create lag variables of all the features or just the target variable to be used as covariates

The code optimizes considering the following aspects:

1. Hyperparameter Tuning; Optionality is added to input the range in which the hyperparameters can be tested. In this case, the alpha and l1_ratio was tested in the range of 0 to 1 with gap of 0.1
2. Number of lags: Different lags were tested to find optimal lag length
3. Models: List of models to be tested

Result & Interpretation

Running the models on train and test data curated from candidate.csv, the optimal configuration came out to be following:

Feature Engineering: No

Outlier Treatment: Yes

Model: Lasso

Alpha: 0.1

Lag: 3

Include Lags of All Variables: True

RMSE on Test Data: 0.015549

References

Following resources were used to complete the assignment:

- Kelly, Bryan T. and Xiu, Dacheng, Financial Machine Learning (July 1, 2023).
- Gu, S., B. Kelly, and D. Xiu. (2020b). Empirical asset pricing via machine learning
- Chinco, A., A. D. Clark-Joseph, and M. Ye. (2019). Sparse Signals in the Cross-Section of Returns

- Tsay, Ruey S., (1951). Analysis of Financial Time Series

Feedback

1. Were the instructions in the test clear? If not, kindly highlight improvements you would have liked to see

   The instructions were clear but I believe the metadata of the candidate dataset would have helped.

2. Was the model template easy to work with? Would you suggest any changes to it?

   The template helped a lot in keeping the structure consistent and the instruction to stick to scikit-learn methodology and passing arguments to the functions helped in thinking out the solution. But, I felt a major limitation with the structure which was that you couldn't use the rolling training window method as the model template required you to train once on the training data and predict on behalf of that, thus using all the data which felt like a major hurdle in granular time series data where the relation is dynamically changing as suggested by Chinco (2019). There is also a possibility that I made a mistake in interpreting the model template or the intent of the treat it like a cross-sectional data problem.

3. Was the amount of data supplied enough for the modeling task? Would having more data have helped in terms of granularity or timespan?

   As I was working on linear models, the data was sufficient considering this granularity. The issue would have arised if the model template allowed for recursive modeling or if a separate model is trained for specific time, for example, separate model for predicting returns at 9:15 AM v/s 3:00 PM where the data points might have been low in rich models. Deep Neural Network models might have also faced lack of data issue but shallow networks might have worked well with this amount of data and granularity.

4. How much time did you spend on the test in total?

   I spent close to 30 hours in the test in total. A large portion of it didn't reach fruitation as I was initially working with recursive training models which didn't work with model template

5. How much was spent in –
   a. Exploring the dataset

      ~20% time

   b. Feature generation

      ~20 % time

   c. Model Fitting and Validation

      ~60% time

6. Did you experience any challenges while attempting this test?

   I faced following challenges:

a. Working with data where no information was given about covariates
b. Analyzing highly correlated variables
c. Handling outliers
d. Avoiding Lookahead bias in each operation
e. Streamlining the models with the model template
f. Lack of focused time slots and delays due to family emergencies

7. Did you learn something new while working on the test?

I learn following new things:

a. Working with non-metadata covariates
b. Time Series properties of granular financial data (Not shown code for sake of brevity)
c. Various cross validation routines in both time series and cross sectional dataset
d. Outlier Treatment and Feature generation while avoiding lookahead bias
e. Handling high correlated covariates
f. Incorporating temporal nature in cross-sectional setup using optimal lags
g. Automated model selection in OOP based code base

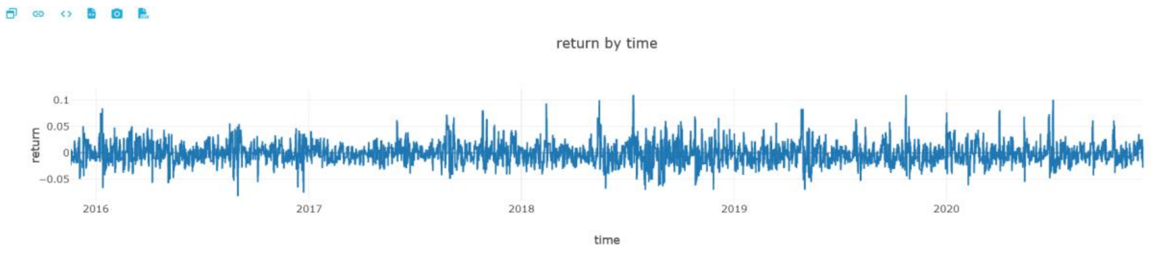
8. What sort of approaches did you try? What worked and what didn't work?

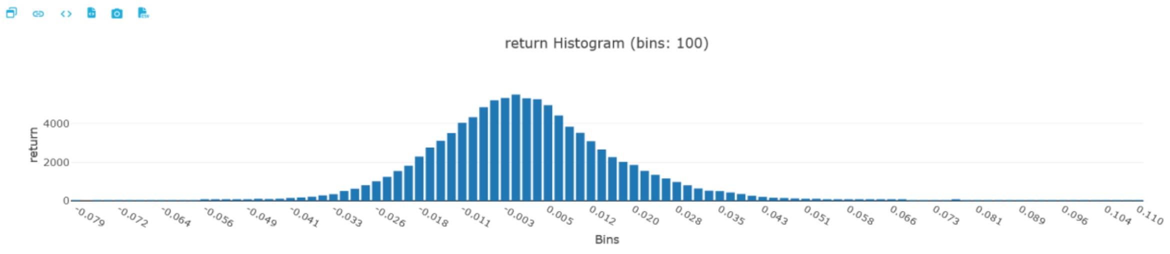
Due to lack of time, I only tried linear models. Regularized linear regression models worked as the features were correlated even after dropping highly correlated variables. Additionally, the number of features increased exponentially when increasing lags, making regularized models more robust and stable compared to Linear Regression model thus OLS models seem to fail in this case.
In terms of data approach, cross sectional approach seems to work, with temporal effect incorporated through lags.

Appendix

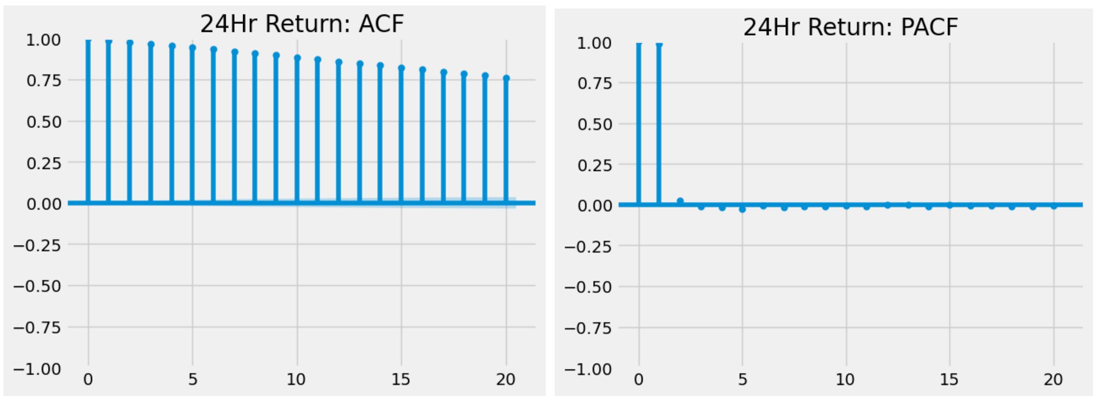
Understanding Data & Analysis

<u>Returns Distribution</u>





Return Properties



	lb_stat	lb_pvalue
10	8.794268e+05	0.0
20	1.547151e+06	0.0
30	2.029342e+06	0.0
40	2.356826e+06	0.0

- Ljung Box Test Statistic rejects null hypothesis that return is white noise => scope of prediction
- From ACF and PACF plot, looks like non-stationary as ACF decaying slowly or AR(1) with high persistence, check it explicitly below

	Test Statistic	pvalue	usedlag	num_obs	1pct_sig	5pct_sig	10pct_sig	best_ic
0	-40.153396	0.0	55	99865	-3.430415	-2.861569	-2.566785	-907048.05724

- Augmented Dicky Fuller test is rejected stating that the return series is non-stationary
-

Thus, no serial correlation and hence the traditional ARMA models won’t help in predicting the series