Reimplementation: Autoencoder Asset Pricing Models

Shuai SONG, Mingxi CHUAN
song-s20@mails.tsinghua.edu.cn, chuanmx@qq.com

Abstract

In this project, we reproduced most of the content of the paper from Gu, Kelly, and Xiu (GKX, 2019). Our work is strictly in accordance with the processing method of the original paper, and confirmed some details having not been mentioned. Although the final results of experiments are slightly different from GKX's, mainly due to some hyperparameters which have not been given, our work provides a extensible framework that will be helpful for further research in the future. Especially, we use class template inheritance to implement all models, which not only accords with the theory of factor model, but also makes the implementation more elegant and efficient. The code is available on *https://github.com/RichardS0268/Autoencoder-Asset-Pricing-Models*.

1 Introduction

Starting from the most basic CAPM model, researchers try to explain the return rate of assets through the factor model, which can be uniformly expressed by the following formula

$$\mathbf{R}_{N\times 1} = \beta_{N\times K}\mathbf{F}_{K\times 1} + \mathbf{U}_{N\times 1} \ \ (*)$$

where R: Excess Return, β Risk Exposure, F: Risk Premium.

In order to improve the explanatory power of the model, researchers have made many extensions and improvements to the model. From CAPM models to Fama-French three-factor and five-factor models, researchers have tried to improve the explanatory power of models by increasing the number of observable factors. However, this kind of model still cannot explain the return rate of assets well. Kelly, Pruitt, and Su (KPS, 2019) proposed through empirical experiments that factors (latent variables) used to explain asset returns are actually proxy variables of unobservable variables baesd on assets' characteristics. Moreover, an asset's exposure to these latent variables should vary over time. In other words, characteristics appear to predict returns because they help pinpoint compensated aggregate risk exposures. KPS propose a method to use asset attributes as instrumental variables to calculate factors and asset exposure. The new method is called instrumental PCA (IPCA) and has following formation

$$\mathbf{R}^t_{N\times 1} = \beta^{t-1}_{N\times K}\mathbf{F}^t_{K\times 1} + \mathbf{U}_{N\times 1},\ \beta^{t-1}_{N\times K} = \mathbf{Z}^{t-1}_{N\times P}\Gamma_{P\times K} + \mathbf{v}_{N\times K}$$

In this form, KPS splits β into a time variant part Z and a time invariant part Γ.

Although IPCA uses characteristics of assets as instrumental variables, Γ, which does not change over time, is essentially a linear function and does not combine these features and their covariate well. Therefore, GKX proposed the autoencoder model in the paper. Their main motivation is to use neural networks to fit beta and factor in factor pricing model respectively. They introduce nonlinear relations for feature Z and its covariates through activation functions (relu) in neural networks. In addition, GKX gives the corresponding derivation in the paper, proving that mathematically direct PCA decomposition of the asset return matrix and IPCA using Z as an instrumental variables are both special forms of their autoencoder. We do not provide a detailed proof of equivalence in our report, but the optimization and calculation of various models are derived in the Approach Experiment section.

In terms of data, we used 60 years of historical data of US equity to conduct experiments. Due to computational resource constraints, we did not analyze the performance of the pricing model on a single stock, but instead constructed 94 portfolios based on 94 characteristics. By artificially constructing 94 portfolios, on the one hand, we reduce the amount of computation and make the input of the model more regular. On the other hand, some data noise is indirectly shielded by the combination of long and short, which makes the results more valuable for research. More details of the data process are shown in the Approach section.

2 Related Work

We mainly refer to articles from GKX and KPS. The evaluation techniques of various machine learning methods comes from another article in GKX [citation needed]. In that article, GKX focused on the effectiveness of various machine learning methods for yield prediction. The paper we reproduce, on the other hand, focuses on the effect of using machine learning methods to explain realized returns. In addition, another predecessor related to this paper is Kozak et al. (2018), who propose an approach to factor analysis for asset pricing using principal components from "anaomaly"-sorted portfolios. Their approach is similar to how we build 94 portfolios with 94 characteristics.

The rest of our report is organized as follows. In section 3, we explain the models and methods mentioned in GKX's paper in detail. In section 4, we We explain the design and results of the replicated experiment. Finally, in section 5, we make a summary of this project and propose some directions for further research in the future.

3 Approach

3.1 Base Model

As $(*)$ equation, all models fall into the form of it. So when we implement it, we use class inheritance, the modelBase class represents the model in $(*)$, and all the remaining models are its subclasses. The main approach of modelBase is shown in the following figure

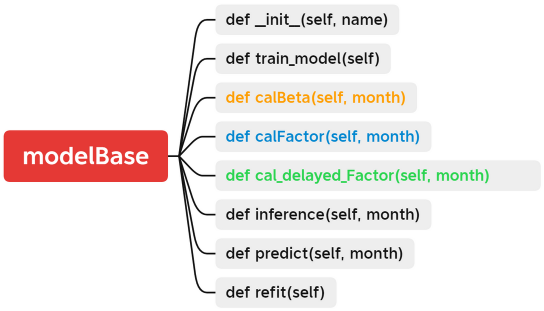

Figure 1: modelBase functions

Different subclasses override methods for calculating beta, factor, and so on in modelBase. This way of implementation is very consistent with the factor pricing model, and the code implementation is more concise and efficient.

3.2 FF model

The first model we reproduce is one with an observable factors. The advantage of this model is that the factors are interpretable, and all the factors are defined by clear meaning and calculation formulas. The disadvantage is that the explanatory power of the model is not high, the characteristic of the asset itself is not used, and the calculation of factor exposure is also realized by simple linear regression.

The model can be expressed as follows

$$y_{i,t} = \sum_{j=1}^{K} \beta_{i,j} f_{j,t} + u_{i,t}$$

Where K takes 1, 2, 3, 4, 5, 6, and the observable factors added to the model in turn are market, SMB, HML, CMA, RMW, and UMD respectively. It is worth noting that the model is actually CAPM when only the market factor is used. When market, SMB, HML are included, the model is fama-french three-factor model. When market, SMB, HML, CMA, RMW are included, the model is fama-french five-factor model. The last UMD is the momentum factor proposed by fama-french. The data source for the implementation model is the french website , which has available data for all six factors since July 1963.

The estimation of FF model adopts OLS multiple regression, and the intercept term is not added in the regression, which is for the purpose of analyzing the explanatory power of the model later.

3.3 PCA model

Unlike FF model, which has observable factors, PCA model and the following models assume that factors are implicit variables. The PCA model is as follows:

$$y_t = \beta_{t-1} f_t + \epsilon_t$$

It is called the PCA model because it assumes that beta does not change over time and tries to solve the following optimization problems,

$$argmin_{\beta} ||y - \beta f||$$

This problem can be solved using PCA method. Using the least square method, for the latent factor model, the estimator of the feature is

$$\hat{f}_t = (\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y$$

residual error is

$$y_t - \beta_{t-1} \hat{f}_t = y_t - \beta_{t-1} (\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t$$

the square of the residual is

$$[y_t - \beta_{t-1}(\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t]^T [y_t - \beta_{t-1}(\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t] = y_t^T y_t - y_t^T \beta_{t-1} (\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t$$

thus minimizing the square of the residual is maximizing $y_t^T \beta_{t-1} (\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t$. Note that

$$y_t^T \beta_{t-1} (\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t = tr\big((\beta_{t-1}^T \beta_{t-1})^{-1} \beta_{t-1}^T y_t y_t^T \beta_{t-1}\big)$$

Assume $\beta_{t-1} = \beta$, which is time invariant, then the maximization problem is equivalent to

$$\max_{\beta} (NT)^{-1} tr\Big((\beta^T \beta)^{-1} \beta^T \sum_{t=1}^{T} y_t y_t^T \beta\Big)$$

Using Rayleigh theorem, the optimal β are the eigenvectors corresponding to the first K eigenvalues of $\frac{1}{NT} \sum_{t=1}^{T} y_t y_t^T$.

The PCA model implemented in our code is step-by-step following above procedure.

3.4 IPCA model

For IPCA model, the form is the same as PCA model but we do not assume β is time invariant. Define $x = Z_{t-1}^T y_t / N$, then

$$(NT)^{-1} \sum_{t=1}^{T} tr\Big((\Gamma^T Z_{t-1}^T Z_{t-1} \Gamma)^{-1} \Gamma^T Z_{t-1}^T y_t y_t^T Z_{t-1} \Gamma\Big) = \sum_{t=1}^{T} tr\Big((\Gamma^T [Z_{t-1}^T Z_{t-1}/N] \Gamma)^{-1} \Gamma^T (x_t^T x_t) \Gamma\Big)/T$$

when $Z_{t-1}^T Z_{t-1}/N$ is I, the estimation of Γ is the eigenvectors corresponding to the first K eigenvalues of $\sum_{t=1}^{T} x_t^T x_t / T$.

When $Z_{t-1}^T Z_{t-1}/N$ is not I, which is often the case, we can following the procedure proposed by KPS in their paper. In their paper, they implement the IPCA by an iterated function to update Γ. Assume we have Γ_{old}, then

$$\hat{f}_t = (\Gamma_{old}^T Z_{t-1}^T Z_{t-1} \Gamma_{old})^{-1} (\Gamma_{old}^T Z_{t-1}^T y_t) = (\Gamma_{old}^T W_{t-1} \Gamma_{old})^{-1} (\Gamma_{old} x_t^T)$$

where $W_{t-1} = Z_{t-1}^T Z_{t-1}/N$. Then we can derive Γ_{new} based on Z_{t-1} and $\hat{f}$. Consider

$$\min_{\Gamma} \sum_{t=1}^{T} (y_t - Z_{t-1}\Gamma f_t)^T (y_t - Z_{t-1}\Gamma f_t)$$

calculate

$$(y_t - Z_{t-1}\Gamma f_t)^T (y_t - Z_{t-1}\Gamma f_t) = y_t^T y_t - 2 y_t^T Z_{t-1} \Gamma f_t + f_t^T \Gamma^T Z_{t-1}^T Z_{t-1} \Gamma f_t$$

The first order condition of Γ is

$$\sum_{t=1}^{T} Z_{t-1}^T Z_{t-1} \Gamma f_t f_t^T = \sum_{t=1}^{T} Z_{t-1}^T y_t f_t^T$$

Apply the vec operator to both sides of the equation,

$$\sum_{t=1}^{T} \left[(f_t f_t^T) \otimes Z_{t-1}^T Z_{t-1} \right] vec(\Gamma) = \sum_{t=1}^{T} (f_t \otimes Z_{t-1}^T) y_t$$

and then the estimation of Γ_{new} is

$$vec(\Gamma_{new}) = \left(\sum_{t=1}^{T} \left[(f_t f_t^T) \otimes (Z_{t-1}^T Z_{t-1}) \right] \right)^{-1} \sum_{t=1}^{T} (f_t \otimes Z_{t-1}^T) y_t = \left(\sum_{t=1}^{T} \left[f_t \otimes Z_{t-1}^T \right] \left[f_t^T \otimes Z_{t-1} \right] \right)^{-1} \sum_{t=1}^{T} (f_t \otimes Z_{t-1}^T) y_t$$

The IPCA model implemented in our code is step-by-step following above procedure.

3.5 Conditional Autoencoder

From PCA model to IPCA mode, we introduce 94 characteristics as instrument. However, we only use the linear combination of these instrument variables and their covariates. In the Conditional Autoencoder (CA) model, we try to fit a nonlinear combination between them by using activation function in the networks. The overall architecture of the CA model is as following. There are two networks in the CA model, representing beta and factor in the factor pricing model respectively. Inputs of the beta network are characterics (instrument variables) and the inputs of factor network are either stock-level returns or portfolio returns. In our implementation, we use 94 portfolios rather than individual stocks. That is, the beat network inputs have dimension of 94*94 ($P * N$) and the factor work inputs have dimension of 94*1 ($P * 1$).

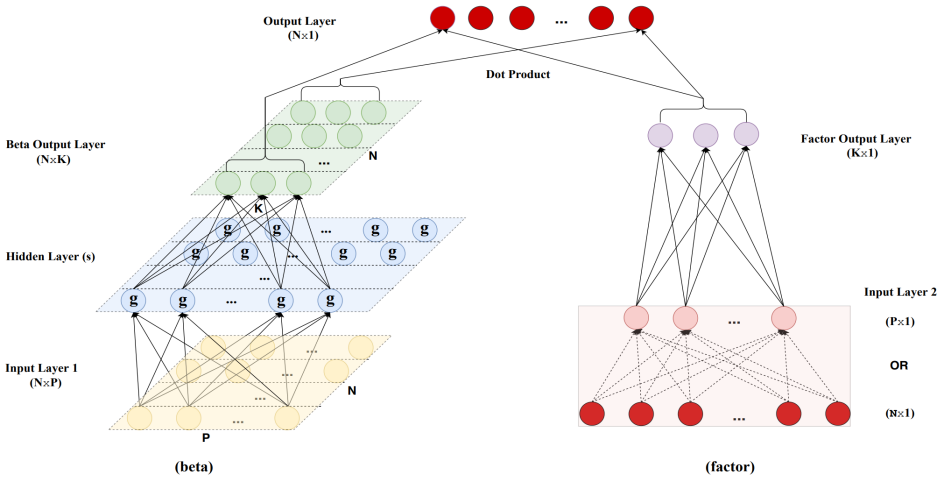

Figure 2: Architecture of Conditional Autoencoder

We construct 4 types of CA models following original paper, i.e. CA0, CA1, CA2 and CA3. They are different in the structure of beta network. For CA0, the beat network of it is just a linear layer, making it similar (but not identical) to IPCA. For CA1, there is a hidden layer with 32 neurons in the beta network. CA2 adds a second hidder layer with 16 neurons and CA3 adds a third hidden layer with 8 neurons.

Note that CA0 through CA3 all maintain a one-layer linear specification on the factor network. In these cases, the only variation in factor network is in the number of neurons, which is range from 1 to 6, corresponding to the number of factors in the model. The intuition of the one-layer factor network is to maintain the interpretability of the factor to some extent, that is, the linear combination of the long-short portfolio returns of the stock-level characteristics.

4 Experiment

4.1 Data

The data we have is downloaded from Xiu's website. The dataset contains monthly individual stock returns from the Center for Research in Securities Prices (CRSP) for all firms listed in the three major exchanges: NYSE, AMEX, and NASDAQ. We use the Treasury bill rate to proxy for the risk-free rate from which we calculate individual excess returns. The sample begins in March 1957 and ends in December 2016, totaling 60 years. For the convenience of realization and data transmission, we modify some format of the original data provided and we save some well-preprocessed data so that it can be used directly when the code is reused. All data process related operation is in the data_prepare.py.

As GKX pointed in the paper, the stock-level characteristics (94 in total) are released to the public with a delay. To avoid a forward-looking bias, they match realized returns at month t with the most recent monthly characteristics at the end of month $t-1$, the most recent quarterly data as of $t-4$, and most recent annual data at of $t-6$. This process has already been done in the data downloaded from Xiu's website.

Distribution of some characteristics are highly skewed and leptokurtic. We follow the method of original paper and rank-normalize all characteristics into the interval (-1, 1) for each month. We then form 94 managed portfolios based on 94 characteristics. For example, to construct the 'mvel1' portfolio, for each month, we rank stocks according to 'mvel1'. Then we write down the permnos of the top 0.1 stocks and the bottom 0.1 stocks. Calculate averages of the 94 characteristics of the top 0.1 stocks and the bottom 0.1 respectively, denoted by $\bar{C}_{top}, \bar{C}_{bottom}$ (94*1 vectors). Then the characteristics of the 'mvel1' portfolio is $0.5 * (\bar{C}_{top} - \bar{C}_{bottom})$.

4.2 Training, Validation, and Testing

We implement a uniform way of training and testing. We divide data into three parts, i.e. train set, valid set, and test set. The initial separation is train set: 18 years (1957-1974), valid set: 12 yeas (1975-1986), and the remaining 30 years (1987-2016) are out-of-sample test set. We use a rolling training shceme to refit models once a year. Each time we refit, we rolling the train set and valid set by one year, maintaining their size. Note that here we are slightly different from the original paper, in which train set is increased by one year each time refit. We compare these two methods and find that there is no significant difference in results but our method has less computation cost.

We apply techniques mentioned in the original papar, including Adam optimizer, batch normaliztion etc. As for the penalization, we do not implement the LASSO penalization used by GKX because it's not supported by pytorch. We try to use L2 penalization instead. Besides, we apply dropout layer to avoid overfitting and use early stop mechanism. We use the valid set for hyperparameters tuning (learning rate, regularization coefficients, drop out rate, etc.). A training process example is shown as below

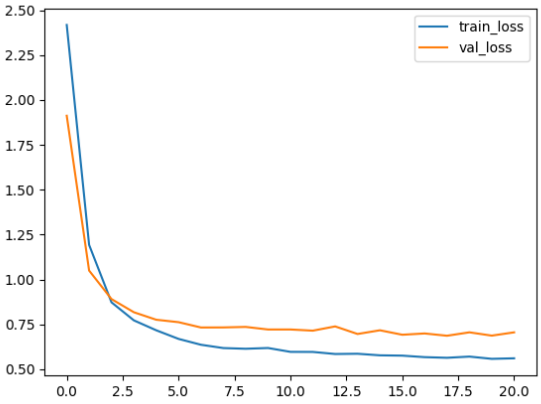

Figure 3: Training Log: CA0, K=3, test_year=1995

Since GKX have not provided the hyperparameters they used in the paper, we cannot exactly coincide our results with theirs.

4.3 Statistical Performance Evaluation

We evaluate out-of-sample model performance using the total and predictive R^2s. The total R^2 quantifies the explanatory power of contemporaneous factor realizations, and thus assesses the model's description of individual stock riskiness:

$$R^2_{total} = 1 - \frac{\sum_{(i,t)\in OOS}(r_{i,t} - \hat{\beta}'_{i,t-1}\hat{f}_t)^2}{\sum_{(i,t)\in OOS} r^2_{i,t}}$$

Model	K=1	K=2	K=3	K=4	K=5	K=6
FF	8.54	15.76	19.86	20.32	31.40	36.16
PCA	40.61	53.00	59.13	62.46	64.68	67.20
IPCA	66.96	77.11	81.98	85.68	86.99	88.67
CA0	56.78	66.77	71.25	74.36	67.27	64.89
CA1	54.70	67.77	71.10	66.60	66.52	74.97
CA2	56.88	64.94	67.17	66.67	69.44	69.94
CA3	47.82	56.23	51.65	51.65	53.95	57.10

Figure 4: R^2_{total} of different models

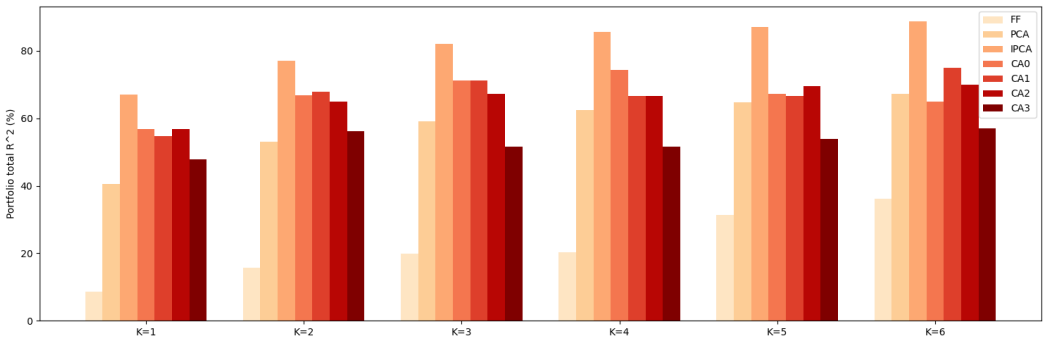

Figure 5: R^2_{total} of different models

The predictive R^2 assesses the accuracy of model-based predictions of future individual excess stock returns, which quantifies a model's ability to explain panel variation in risk compensation:

$$R^2_{pred} = 1 - \frac{\sum_{(i,t)\in OOS}(r_{i,t} - \hat{\beta}'_{i,t-1}\hat{\lambda}_{t-1})^2}{\sum_{(i,t)\in OOS} r^2_{i,t}}$$

where $\hat{\lambda}_{t-1}$ is the prevailing sample average of $\hat{f}$ up to month $t-1$. Note that all factor pricing models are estimated by minizing the realized residuals, that is, trying to maximizing the R^2_{total}. There is a time lag when we use our explanatory models to make predictions, and there is no contraints about R^2_{pred} during the estimation. So we cannot expect R^2_{pred} to be high, and even negative values are possible.

Model	K=1	K=2	K=3	K=4	K=5	K=6
FF	-0.59	-1.10	-0.72	-0.46	-0.38	-0.45
PCA	0.30	0.60	0.69	0.75	0.80	0.82
IPCA	0.15	0.53	-0.34	-0.17	-0.36	-0.73
CA0	0.18	-3.96	-3.46	-1.51	-2.98	-1.87
CA1	-1.61	-5.36	-6.41	-2.05	-3.86	-1.93
CA2	-1.31	-4.73	-1.58	-1.77	-2.47	-0.22
CA3	-0.87	-2.39	-2.00	-1.37	-0.99	-0.63

Figure 6: R^2_{pred} of different models

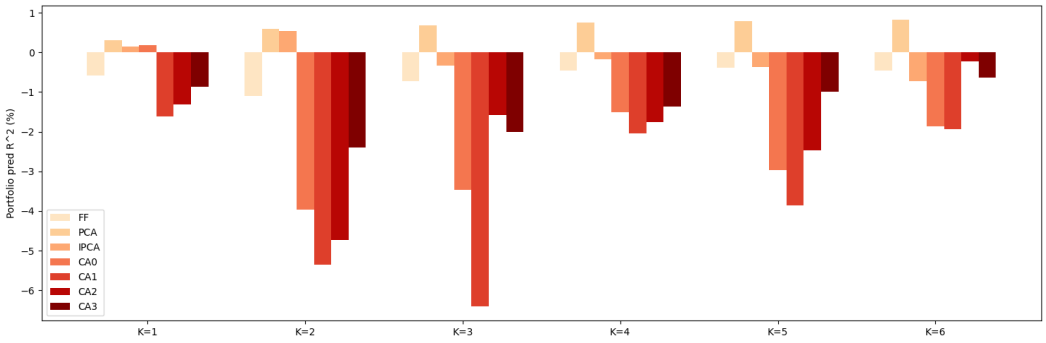

Figure 7: R^2_{pred} of different models

Furthermore, we follow the method to evaluate characteristics' importance mentioned in original paper. By assigning certain characteristic to zero during the model inference (do not change training

process), we calculate the reduction in R^2_{total}, which imply characteristics' importance. The heatmap of reduced total R square is shown as below

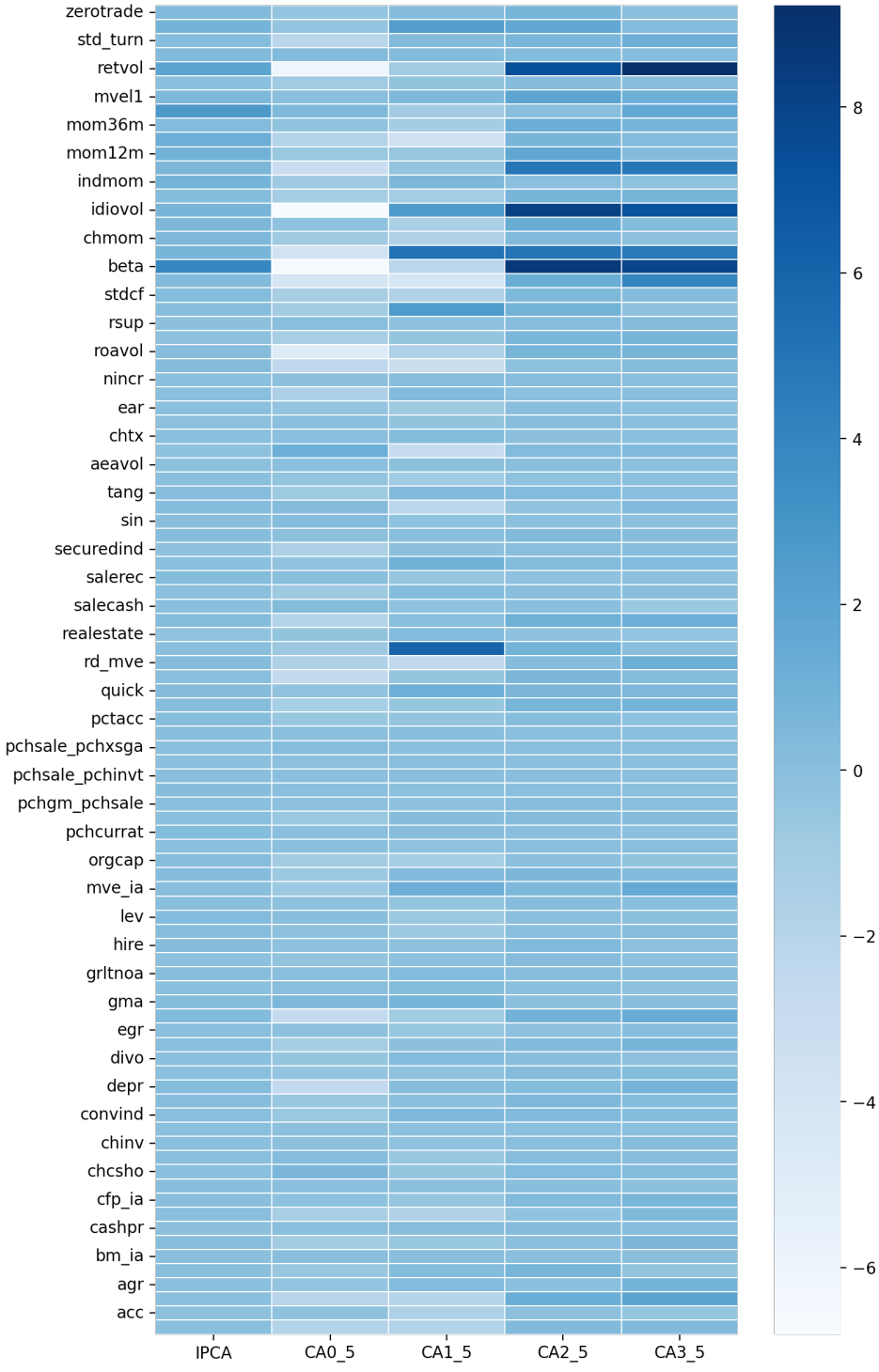

Figure 8: Characteristics importance in reduced R^2_{total} (K=5, value in %)

4.4 Risk Premia v.s. Mispricing

We also follow GKX to carry out the mispricing analysis. Notice that all models we implemented above have no intercepts. Therefore, we can directly test whether the zero-intercept no-arbitrage restriction is satisfied in the data. If it is, the time series average of model residuals for each portfolios, that is, the pricing errors—should be statistically indistinguishable from zero (two side test). The uncoditional pricing errors are defined as

$$\alpha_i \equiv E(u_{i,t}) = E(r_{i,t}) - E(\beta'_{i,t-1} f_t)$$

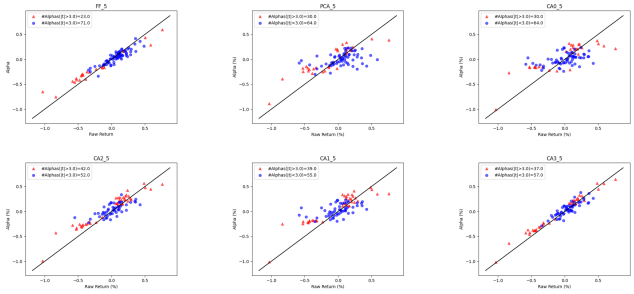

Figure 9: alpha of some models

Take K=5 for example, the x axis represents portoflios' return ($E[r_{i,t}]$) while the y axis represents portfolios' alpha (pricing errors). Blue dots represent insignificant alphas while red dots represent significant alphas. The flatter the distribution of points, the better the interpretation of the model. It can be observed that, compared with the FF_5 model, PCA and CAs can explain the portfolios' returns better.

5 Conclusion

In this project, we reproduced most of the content of the GKX paper. Due to the limitation of computing resources, we only carried out the part of managed portfolios. For the implementation details not mentioned in the original paper, we determined them through online search and experimental attempts. Since GKX did not give some training hyperparameters in the paper, and we did not search for optimal parameters due to limited computational resources, thus our final results are slightly different from those in the original paper. For code implementation, we used classes and inheritance. This not only fits the form of factor pricing model well, but also makes the implementation process elegant and efficient, and provides a reference implementation framework for further research on stock dimension data.

References

[1] Gu, Shihao, Kelly, Bryan T. & Xiu, Dacheng, (2019), Autoencoder Asset Pricing Models, *Yale ICF Working Paper No. 2019-04, Chicago Booth Research Paper No. 19-24* .

[2] Kelly, Bryan, Seth Pruitt & Yinan Su (2019), Characteristics are covariances: A unified model of risk and return, *Journal of Financial Economics, forthcoming* .

[3] Gu, Shihao, Bryan Kelly, & Dacheng Xiu (2019), Empirical asset pricing via machine learning, *Review of Financial Studies, forthcoming* .