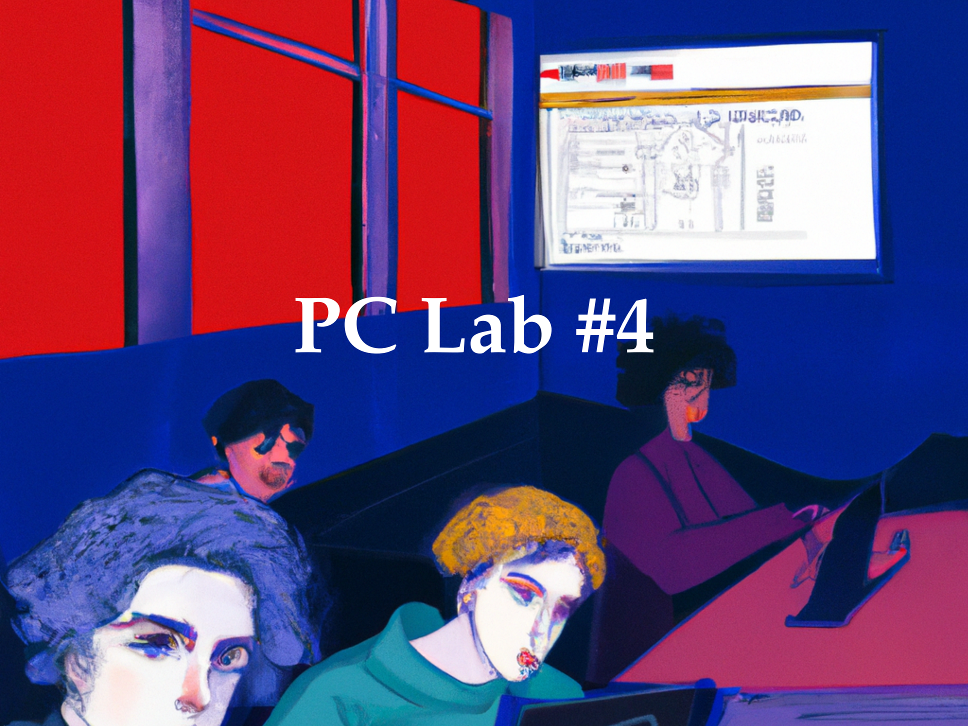
# 20598 – Finance with Big Data

PC Lab #4: Predicting Stock Returns with ML (Week 5)

Clément Mazet-Sonilhac
clement.mazetsonilhac@unibocconi.it

Department of Finance, Bocconi University

# PC Lab #4



# PC Labs Grading

- PC Labs solutions are submitted as Jupyter Notebooks, via email
  - Email title : PCLab#4 - Group X - Name1 Name2 Name3
  - Your Jupyter Notebook starts in the same way (same .ipynb name)
  - Tell me (again) how long did it take

- PC Labs grade will depend on :
  - Your ability to submit it before the deadline
  - The quality of your code (comments, readability, use of functions, etc.)
  - The structure of the Jupyter Notebook : well organized, explain what you are doing and why
  - Your ability to complete the tasks and innovate
  - You should maybe produce less, but more *useful* outputs

# PC Labs Grading

- You're doing a fantastic job!

- PC Labs are great opportunities to learn + create a portfolio of projects
  $\rightarrow$ super useful on CVs and during job interviews

# PC Labs Grading

- You're doing a fantastic job !
- PC Labs are great opportunities to learn + create a portfolio of projects
  $\rightarrow$ super useful on CVs and during job interviews
- But also time-consuming. PC Labs shouldn't have negative spillovers !
- Remember : only 6 over 9 PC Labs will count towards your final grade
  $\rightarrow$ You can take it easy !

The Review of Financial Studies

# Empirical Asset Pricing via Machine Learning*

**Shihao Gu**
Booth School of Business, University of Chicago

**Bryan Kelly**
Yale University, AQR Capital Management, and NBER

**Dacheng Xiu**
Booth School of Business, University of Chicago

We perform a comparative analysis of machine learning methods for the canonical problem of empirical asset pricing: measuring asset risk premiums. We demonstrate large economic gains to investors using machine learning forecasts, in some cases doubling the performance of leading regression-based strategies from the literature. We identify the best-performing methods (trees and neural networks) and trace their predictive gains to allowing nonlinear predictor interactions missed by other methods. All methods agree on the same set of dominant predictive signals, a set that includes variations on momentum, liquidity, and volatility. (*JEL* C52, C55, C58, G0, G1, G17)

# Goals

- Manipulate and visualize financial data (prices and volumes)
- Train a model to predict stock returns
- Use the predictions to build a AI-driven portfolio

# Big picture context

- You're the new intern at Renaissance Technologies LLC !
- Jim Simons has seen the Gu et al. (2020) and the Jiang et al. (2023) papers

- He asks you to try some of the predictive algorithms, but at higher frequency (daily) and with less signals (only the one used in Jiang et al. (2023) : prices and volume).

# Task #1 : Basic manipulation and descriptive statistics

- Import the `Data_PCLab4_Stock_Price.csv` data and the `Data_PCLab4_Stock_Volume.csv` or use Yahoo Finance data
- Describe the sample (optional) :
  - What is the average trading volume for Apple stock ?
  - What is the maximum trading volume for S&P500 ?
  - Which security is traded the most ? Comment on your answer
- Plot the time series of volumes for all stocks (raw and normalized)
- Is there a correlation between change in prices (returns) and change in volumes ?

# Task #2 : Train and Test samples + Ridge regression

- Concatenate the date, stock price, and volume in one dataframe
- Tip : scale the data

```
from sklearn.preprocessing import MinMaxScaler
sc = MinMaxScaler(feature_range = (0, 1))
training_set_scaled = sc.fit_transform(training_data)
```

# Task #2 : Train and Test samples + Ridge regression

- You want to predict **stock returns** (alternatively, the price or the probability of a positive return) over short (5-day), medium (20-day), and long (60-day) horizons using stock prices (returns) and volumes at over the past 5, 20, and 60 days :

$$P_{i,t+x} = f(P_{i,t}, V_{i,t}) \text{ or } r_{i,t+x} = f(r_{i,t}, V_{i,t})$$

- Split the sample : 75% training, 25% testing

- Create and train i) an OLS model and ii) Ridge linear regression model (and play with the penalty parameter)

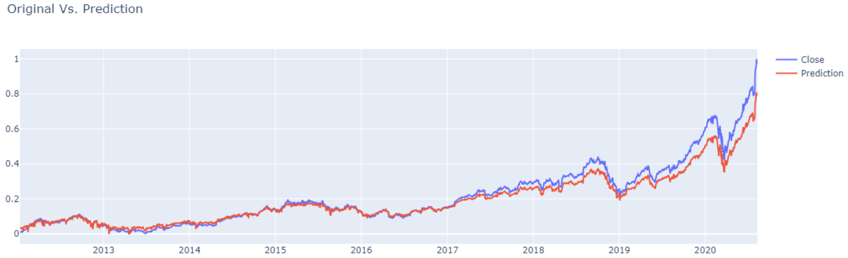
- What is the $R_{OOS}$ of the 2 methods ? Use the formula from Gu et al. (2020) (eq. 19, p. 2246)

$$R_{i,OOS} = 1 - \frac{\sum_{t \in T}(r_{i,t} - \hat{r}_{i,t})^2}{\sum_{t \in T}(r_{i,t})^2}$$

- Alternative measures: number of time the predicted return and the realized return have the same sign (or MSE, R-square)

# Task #2 : Train and Test samples + Ridge regression

- For Apple, you may be able to plot similar result :



# Task #2 bis : Improving the model ?

- What about including the market (and being closer to the theory) ?

$$P_{i,t+x} = f(P_{i,t}, V_{i,t}, P_{M,t})$$

- What about including more signals ?

$$P_{i,t+x} = f(P_{i,t}, V_{i,t}, P_{M,t}, \Delta P_{i,t-h})$$

# Task #3 : Same but with NN or Trees

- Pick one (or several) machine learning model: e.g., NN with many layers
- Same steps as Task #2
- Comment 1: which method performs the best ? Is it in line with Gu et al., (2020). Can we really compare ?
- Comment 2: What about Jiang et al. (2023) paper?

# Task #4 : Performance of the AI-driven portfolio

- Create a long portfolio by selecting every day the 4 assets with the highest predicted return/price at t+x (i.e., you re-balance every x day). You initially invest 100$, how much do you have at the end of the testing period ?

- Compare your result to 1000 portfolios with random weights (you generate the weights at the beginning of the testing period and you never re-balance your portfolio).

- Imagine that you now pay trading fees : 3 bps (basic point) of the amount invested is charged for every transaction, what is the new performance of your AI-driven portfolio ?

  - You want to sell 50$ of stock A to buy the same amount of stock B, you will be charged 0.015$, so you'll end up with only 49.985$ invested in stock B.

# Packages you may need

- Among others : `wordcloud`, `nltk.stem`, `nltk.corpus`, `nltk.tokenize`, `gensim`, `tensorflow`, `string.punctuation`, `sklearn`, `etc.`