Artificial Intelligence in Fintech **20 Nov, 2023**

Project 3: Final

Instructor: Yuan Yao *Due: 23:59 Sunday 3 Dec, 2023*

1 Project Requirement

This project as a warm-up aims to explore basic techniques in machine learning.

1. Pick up ONE (or more if you like) favourite dataset below to work. If you would like to work on a different problem outside the candidates we proposed, please email course instructor about your proposal.

2. Team work: we encourage you to form small team, up to **FOUR** persons per group, to work on the same problem. Each team just submit

   (a) *ONE report, with a clear remark on each person's contribution.* The report can be in the format of either a *poster*, e.g.

   `https://github.com/yuany-pku/2017_math6380/blob/master/project1/DongLoXia_poster.pptx`

   or *technical report within 8 pages*, e.g. NIPS conference style (preferred format)

   `https://nips.cc/Conferences/2019/PaperInformation/StyleFiles`,

   with source codes such as Python (Jupyter) Notebooks with a detailed documentation.

   (b) *ONE short presentation video within 10 mins*, e.g. in Youtube or Bilibili link. You may submit your presentation slides together with the video link to help understanding.

3. For Kaggle contests, please register your team with name in the format of math6010z_lastname, so that we could easily find your results on Kaggle. For example, a team with Shawn Zhu and Kate Wong would be named by math6010z_Zhu_Wong.

4. In the report, show your proposed scientific questions to explore and main results with a careful analysis supporting the results toward answering your problems. If possible, you should include your Kaggle contest score or rating in the report. Remember: scientific analysis and reasoning are more important than merely the performance tables. Separate source codes may be submitted through email as a GitHub link, or a zip file.

5. Submit your report by email or paper version no later than the deadline, through Canvas. To ensure a smooth review process, each student in the group is required to submit a copy.

2 Kaggle: G-Research Crypto Forecasting

Over USD40 billion worth of cryptocurrencies are traded every day. They are among the most popular assets for speculation and investment, yet have proven wildly volatile. Fast-fluctuating prices have made millionaires of a lucky few, and delivered crushing losses to others. Could some of these price movements have been predicted in advance?

In this competition, you'll use your machine learning expertise to forecast short term returns in 14 popular cryptocurrencies. We have amassed a dataset of millions of rows of high-frequency market data dating back to 2018 which you can use to build your model. Once the submission deadline has passed, your final score will be calculated over the following 3 months using live crypto data as it is collected.

`https://www.kaggle.com/c/g-research-crypto-forecasting`

2.1 Background Knowledge

The simultaneous activity of thousands of traders ensures that most signals will be transitory, persistent alpha will be exceptionally difficult to find, and the danger of overfitting will be considerable. In addition, since 2018, interest in the cryptomarket has exploded, so the volatility and correlation structure in our data are likely to be highly non-stationary. The successful contestant will pay careful attention to these considerations, and in the process gain valuable insight into the art and science of financial forecasting.

2.2 Model and Methods

Given a time-series based dataset of cryptocurrency prices, one can use the hierarchical time series model, which is to train a model for all time, models per weekday, model per day, etc. and ensemble them together. Multiple popular models could be applied and you might conduct research about the effectiveness of these models. For example, LSTM is popular among quantitative finance fields, as it can capture the relationship between previous cryptocurrency prices. Since cryptocurrency price is a time-dependent variable, multiple time-series prediction models may also be adopted, like traditional ARIMA, GARCH, and Dynamic linear model. You are welcome to create innovative methods, bringing more insights to the quantitative finance field.

2.3 Data Source

Kaggle link:

`https://www.kaggle.com/c/g-research-crypto-forecasting/data`

3 Kaggle: M5 Forecasting

There are two complementary competitions that together comprise the M5 forecasting challenge:

- Accuracy, Estimate the unit sales of Walmart retail goods. Can you estimate, as precisely as possible, the point forecasts of the unit sales of various products sold in the USA by Walmart?
  `https://www.kaggle.com/c/m5-forecasting-accuracy`
- Uncertainty, Estimate the uncertainty distribution of Walmart unit sales. Can you estimate, as precisely as possible, the uncertainty distribution of the unit sales of various products sold in the USA by Walmart?
  `https://www.kaggle.com/c/m5-forecasting-uncertainty`

How much camping gear will one store sell each month in a year? To the uninitiated, calculating sales at this level may seem as difficult as predicting the weather. Both types of forecasting rely on science and historical data. While a wrong weather forecast may result in you carrying around an umbrella on a sunny day, inaccurate business forecasts could result in actual or opportunity losses. In this competition, in addition to traditional forecasting methods, you're also challenged to use machine learning to improve forecast accuracy.

In this competition, you will use hierarchical sales data from Walmart, the world's largest company by revenue, to forecast daily sales for the next 28 days and to make uncertainty estimates for these forecasts. The data, covers stores in three US States (California, Texas, and Wisconsin) and includes item level, department, product categories, and store details. In addition, it has explanatory variables such as price, promotions, day of the week, and special events. Together, this robust dataset can be used to improve forecasting accuracy.

4 Cryptocurrency Trading

4.1 Project Overview

Cryptocurrencies are fast becoming rivals to traditional currency across the world. The digital currencies are available to purchase in many different places, making it accessible to everyone, and with retailers accepting various cryptocurrencies it could be a sign that money as we know it is about to go through a major change.

In addition, the blockchain technology on which many cryptocurrencies are based, with its revolutionary distributed digital backbone, has many other promising applications. Implementations of secure, decentralized systems can aid us in conquering organizational issues of trust and security that have plagued our society throughout the ages.

In this project, we aim to use historical minute-level OHLCV (opening, high, low, close, volume) data of 4 major cryto currencies to simulating trading process. We hope your designed trading policy can generalize to various kinds of crypto currencies and can gain promising return.

4.2 Data Description

You are provided with historical minute-level OHLCV data of 4 major cryto currenciesâ BTC, EOS, ETH, TRX. The raw data is downloaded through a web crawler, which contains information including OHLCV and number of trades. After preprocessing, the minute-bar data in the standard OHLCV format is obtained.

Currently, the training set's time scale ranges from "2021-09-14 00:00:00" to "2023-04-19 00:00:00". The remaining time periods are reserved as the backtesting set.

Preprocessed data download address:

- .h5 format(recommend):
  `https://drive.google.com/drive/folders/1g-uyTcdD1PFP-HufnrgXK48JmjYimFa1?usp=drive_link`
- non-splitted csv format:
  `https://drive.google.com/drive/folders/1yjMuQftRQFBBhKyEdonLZMCjhKtkpMIg?usp=drive_link`

4.3 Task Description

You need to write a minute-level trading strategy function. Given data from one-minute, it can output its desired position next minute, which implies how will you trade (long/short) assests next minute in Python script.

We provide you with backtesting script and several demos for reference, please refer to these instructions. (To ensure backtesting script works, we suggest you follow demo's format to writing your strategy code)

- **Trading Guideline**
  - The initial asset you own is $100,000 (USD)
  - At one minute, your strategy should make decision about longing/shorting difference crypto currencies next minute by giving your desired position next bar
  - During trading, we also consider the transaction cost. The transaction rate set as 0.0005 for each trading action. For example, suppose you strategy will short 5 BTC next minute, and the average price of BTC is $9000 next minute, then your transaction cost will be $9000 \times 5 \times 0.0005 = \$22.5$. Suppose after one hour, the average price increases to $9500 and tiy want to close to your position, then you need to pay another $9000 \times 5 \times 0.0005 = \$27.5$ as transaction cost. (But no worry, the backtesting script has taken care of transaction cost)
- **Performance Evaluation**

We can use Sharpe Ratio as the metrics. The formula for Sharpe Ratio is

$$Sharpe\ Ratio = \frac{R_p - R_f}{\sigma_p},$$

where R_p= return of portfolio, R_f=risk-free rate, σ_p=standard deviation of the portfolio's excess return.

4.4 Submission

You should submit your report and source material (a zip folder contains: your "strategy.py" and other facility files). For detailed information, please refer to demo files at the following link:
`https://github.com/yao-lab/yao-lab.github.io/tree/master/aifin/2021/project3/demo`

4.5 Extension

If you want to explore the performance of your trading strategy in more cryptocurrencies and under different trading frequencies, we will also provide additional historical OHLCV data (including 'BTCUSD', 'BCHBTC', 'ETHUSD', 'LTCUSD' transaction data in the minute /15-minute/1-hour/1-day bar). Please contact us through email. Current data is still in preparation.

5 Large Language Models + Financial Analysis

5.1 Project Overview

Large Language Models (LLM) use transformer models and are trained using massive datasets. This enables them to recognize, translate, predict, or generate text or other content. LLMs have demonstrated remarkable capabilities in automating and streamlining financial analysis tasks. These models can process vast amounts of data, including news articles, research reports, financial statements, and market data, to extract valuable insights. By leveraging their deep learning capabilities, LLMs can identify patterns, trends, and anomalies in financial data, assisting analysts in making more informed investment decisions.

In this project, we present two cases that demonstrate the use of LLMs for financial analysis:

- Case 1: Utilizing a powerful but closed ChatGPT model and flexible framework LlamaIndex, we create a retrieval-augmented generation system to build a customized analysis assistant.
- Case 2: By leveraging open-sourced domain-specific LLMs trained on financial data, we fulfill various analysis tasks such as sentiment analysis, financial forecasting, and risk assessment.

5.2 Example 1: Financial News Analysis with Llama Index and ChatGPT

In this example, we use OpenAI and LlamaIndex to analyze financial news. By integrating LlamaIndex, which expedites the initiation of LLM applications and facilitates the creation of Retrieval-Augmented Generation (RAG) systems within minutes, with the advanced capabilities of ChatGPT from OpenAI, known for generating safer and more relevant responses, and leveraging Streamlit, a Python-based framework tailored for machine learning engineers to swiftly build and disseminate visually appealing machine learning and data science web applications, we can efficiently construct an assistant dedicated to analyzing financial news. This assistant would encompass essential functionalities such as providing a single stock outlook and conducting competitor analysis.

The process involves sequentially moving through three key steps: firstly, fetching news data; secondly, constructing an index for subsequent retrieval purposes; and finally, establishing a platform for query-response interactions. For more details please refer to our tutorial slides.

The reference series of courses can be found at `https://www.youtube.com/playlist?list=PLvzuUVysUFOv4iwJE6p1TtuH_I4MFrK50`. The courses will provide a detailed explanation of the functionality of each component and how to connect them in building a user interface for a financial news app. And for code example, please refer to the repo on GitHub `https://github.com/hackingthemarkets/financial-news-llama-index`.

5.3 Example 2: Local-deployed FinGPT-Forecaster

FinGPT-Forecaster `https://github.com/AI4Finance-Foundation/FinGPT/tree/master/fingpt/FinGPT_Forecaster` is an open-source AI tool designed for financial analysis. It analyzes market news and optional financial data specifically for a chosen company. It provides a stock price movement forecast for the upcoming week, accompanied by a summarized analysis. The model, Fine-tuned on Llama-2-7b-chat-hf with LoRA, incorporates past year's DOW30 market data in its training process. One of the notable features of FinGPT-Forecaster is its adaptability and generalization capabilities, allowing it to provide accurate forecasts and insights across various stock symbols.

In this example, we hope that you can try to deploy the FinGPT-Forecaster locally. The entire process will include downloading weights from Hugging Face $\rightarrow$ installing the necessary environment $\rightarrow$ acquiring raw financial data $\rightarrow$ generating example data and transforming it into Llama2 Format $\rightarrow$ performing model inference in the demo. Remember to refer to the documentation and resources provided with the model for detailed instructions on each step and to make the necessary adjustments based on your specific requirements and data.

5.4 Submission

You are required to submit your report, which should include a demonstration of your *demo application* and the necessary *analysis*. Additionally, please provide the source materials in a zip folder. The zip folder should contain the main logic codes and other essential files, excluding large data files such as checkpoint weights.

5.5 References

- Financial Analyst App - GPT-4, Streamlit, and Llama-Index Tutorial `https://youtu.be/c_LAIkkhKts?si=zbyANC91SymDJuV3`
- LlamaIndex stater tutorial `https://gpt-index.readthedocs.io/en/latest/getting_started/starter_example.html`
- HKUST Azure OpenAI API service `https://itsc.hkust.edu.hk/services/it-infrastructure/azure-openai-api-service`
- FinGPT-Forecaster Huggingface Space Demo `https://huggingface.co/spaces/FinGPT/FinGPT-Forecaster`
- Huggingface Generation with LLMs tutorial `https://huggingface.co/docs/transformers/llm_tutorial`

6 Old Project2: Paper Replications

6.1 *(Re-)Imag(in)ing Price Trends*

6.1.1 Background

We are targeting to replicate the following paper by Jingwen Jiang, Bryan Kelly and Dacheng Xiu: `https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3756587`.

This paper explores convolutional neural networks that flexibly learn price patterns as images that are most predictive of future returns. The raw predictor data are images – stock-level price charts, from which authors model the predictive association between images and future returns using a convolutional neural network (CNN). They claims that by using CNN they can automatically identify context-independent predictive patterns which can gave more accurate return predictions, translate into more profitable investment strategies and are robust to variations.

In the empirical designs, they first embeds 1D time series data in a higher dimensional space, representing it as a 2D image depicting price and volumes. Then they feed each training sample into CNN to estimate the probability of a positive subsequent return over short (5-day), medium (20-day) and long (60-day) horizons. Afterwards, they use CNN-based out-of-sample predictions as signals in a number of asset pricing analyses. Finally, they attempt to interpret the predictive patterns identified by the CNN.

6.1.2 Replication studies

In this reproduction process, we mainly focus on understanding the data preparation (how to transfer 1D time series data to 2D images representing historical market data), model design (CNN architecture design and mechanism behind it), workflow design (from training to model tuning and finally to prediction), performance evaluation and finally the interpretation part.

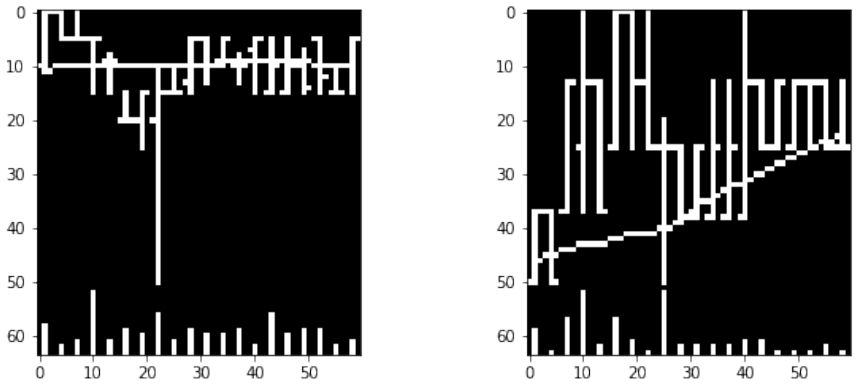

Figure 1: Examples of 20-day Image with volume bar and moving average line

1. **Data**

   The sample runs from 1993-2019 based on the fact that daily opening, high, low prices. In the original paper, authors construct datasets consisting three scale of horizons (5-day, 20-day, 60-day), Here we just collect the 20-day version. The total size of data is 8.6G in a zipped file (802.9MB). The download link of data is:

   `https://dachxiu.chicagobooth.edu/download/img_data.zip`

   or a fast access

   `https://www.dropbox.com/s/njehqednn8mycze/img_data.zip?dl=0`

   with iPython image processing demo in

   `https://dachxiu.chicagobooth.edu/download/img_demo.html`.

   We already transformed the OHLC charts into images following the same procedures introduced in the paper (Section 2). Current images have the same resolution (64 * 60) and added with moving average lines(MA) and volume bars(VB). Some example figures is shown in Figure 1.

   Images labels take value **1** for positive returns and **0** for non-positive returns. In addition, we use **2** to mark the NaN value. In the simplest terms, you need to complete a two-class classification problem, and use the CNN model to predict whether the trend is 'down' or 'up' for the input image. For detail of data and label file, please refer to appendix.

2. **Architecture Design**

   Why use CNN? Since CNN impose cross-parameter restrictions that dramatically reduce parameterization and embed a number of tools that make the model resilient to deformation and re-positioning of important objects in the image. A core building block consists of three operations: convolution, activation and pooling. In the paper, for 20-day image, they build a baseline CNN architectures with 3 conv blocks and connected with a fully connected layer as a classifier head. You should refer to the design of the conv block in the original paper (including the selection of the size of the convolution kernel, the selection of the convolution method, the design of the pooling layer and the selection of the activation function, etc) Figure 2 shows a diagram of 20-day CNN model proposed in the paper, just for your reference.

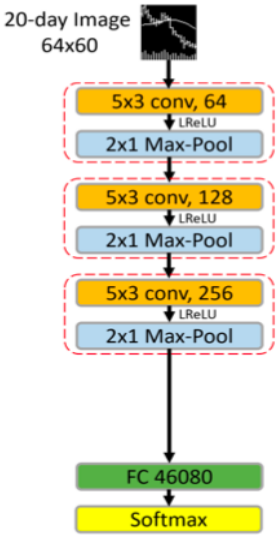

Figure 2: Diagram of CNN model

3. **Working Flow**

Data split First, consider dividing the entire sample into training, validation, and testing samples. In the original paper, they use the first seven-year sample (1993-1999) to train and validate the model, in which 70% of the sample is randomly selected for training and the remaining 30% for validation. The remaining twenty years of data comprise the out-of-sample test dataset. You should consider following the same format in case better compared with the original paper.

Loss and evaluation You can simply treat the prediction analysis as a classification problem. In particular, the label for an image is defined as y = 1 if the subsequent return is positive and y = 0 otherwise. The training step minimizes the cross-entropy loss, which is the standard objective function for classification problem, which define as:

$$L_{CE}(y, \hat{y}) = -y \log(\hat{y}) - (1 - y) log(1 - \hat{y})$$

where $\hat{y}$ is the prediction and y is the ground truth.

To measure the classification accuracy, a true positive (TP) (true negative (TN)) occurs when a predicted probability of greater than 50% coincides with a positive realized return (a probability less than 50% coincides with a negative return, respectively). False positives and negatives (FP and FN) are the complementary outcomes. We calculate classification accuracy as:

$$Accuracy = (TP + TN)/(TP + TN + FP + FN)$$

For more evaluation metrics or methods, like Sharpe Ratio, please refer to the original paper.

Training process The author adopts several ways to combat the over-fitting issue and aid efficient computation. For example, they applied the Xavier initialization for weights in each layer, which guarantees faster convergence by scaling the initial weights. Other techniques like dropout, batch normalization, and early stopping may also improve performance. We recommend referring to the training details mentioned in paper 3.3 when training the baseline model.

4. **Extensions**

   - For ablation studies and testing robustness, we suggest you follow what original paper mentioned in Appendix B. For example, you can perform the same sensitivity analysis of the CNN prediction model to alternate choices in model architecture (e.g. varying the number of filters in each layer or varying the number of layers, like the paper shows in Table 18)
   - Another direction that can be used as an extension is exploring of the interpretability of the CNN model in Chapter 6 of the original paper. Though interpreting a CNN model is quite difficult due to its stacks of non-linear structures, you can imitate what the author did in Part 6.3, using a visualization method (Grad-CAM) to understand how different image examples activate the regions of the CNN to trigger 'up' or 'down' return predictions.
   - What's more, we encourage you not being limited to simple binary classification task, since the label files we provided consist more meaningful attributes, containing both categorical and numerical values. For example, you can use the same 20-day horizon images to train your model to predict the return trend of different subsequent **y**-days even the detailed return values. (**y** can be 5, 20 even larger). In this way, you can prove more firmly that using CNN can automatically identity robust and transferable predictive features.

6.2 Empirical Asset Pricing via Machine Learning

The fundamental goal of asset pricing is to understand the behavior of risk premiums. However, risk premium is difficult to measure: market efficiency forces return variation to be dominated by unforecastable news that obscures risk premiums. This paper predicts the expected return and identifies informative predictor variables via machine learning methods, which facilitates more reliable investigation into economic mechanisms of asset pricing. Now you are required to replicate some results of this paper based your understanding of it, and write a report about your work.

The requirements of this paper replication project are as follow:

- The machine learning methods used in this paper include linear regression (OLS, elastic net), dimension reduction (PLS, PCR), generalized linear models, trees (gradient boosting trees, random forest) and neural networks. Please try to replicate **at least 6 methods** of them (e.g., OLS, elastic net, PLS, PCR, random forest, neural networks, etc. Please note that if you choose OLS, OLS-3 should also be included; and if you choose neural network, NN1 to

NN5 are included. Besides, robust loss function should also be considered. See details in the paper), and analyze your results specifically. Hints on parameter choice are presented in the paper.

- Include the variable importance (section 2.3 of the paper) in your analysis. You do not need to replicate all the figures in section 2.3, but you are encouraged to investigate it carefully.
- Note that this paper uses a 'recursive performance evaluation scheme'. You are also required to evaluate your result by this method. For more details of this method, please refer to the paper and its supplementary material. The PPT presented in class about this project may also be helpful to you.
- As you can know from the paper (section 2.1), predictive characteristics include firm characteristics, sic code and macroeconomic predictors. Firm characteristics and sic code are provided in the original dataset of this paper, and the 8 macroeconomic predictors are constructed following Welch and Goyal (2008), which are not directly provided in the original dataset of this paper. Hence, you may construct the predictors by yourself according to the description in Welch and Goyal (2008), for instance, see `https://christophj.github.io/replicating/r/replicating-goyal-welch-2008/`.
- The portfolio forecast part of the paper (section 2.4) is not compulsory for you to replicate.

You may access the paper and the supplementary material via:
`https://dachxiu.chicagobooth.edu/download/ML.pdf`
or
`https://academic.oup.com/rfs/article/33/5/2223/5758276`.
Meanings of characteristics of the data are provided in the supplementary material.

The original dataset (4.05GB) can be obtained at
`https://dachxiu.chicagobooth.edu/download/datashare.zip`.
The zip file is about 1.64GB. Please be patient since it may take you about 6 hours to download the data. Another fast access can be via
`https://www.dropbox.com/s/zzgjdubvv23xkfp/datashare.zip?dl=0`

7 Old Kaggle Contest in Project 1: Home Credit Default Risk

Many people struggle to get loans due to insufficient or non-existent credit histories. And, unfortunately, this population is often taken advantage of by untrustworthy lenders.

Home Credit strives to broaden financial inclusion for the unbanked population by providing a positive and safe borrowing experience. In order to make sure this underserved population has a positive loan experience, Home Credit makes use of a variety of alternative data–including telco and transactional information–to predict their clients' repayment abilities.

While Home Credit is currently using various statistical and machine learning methods to make these predictions, they're challenging Kagglers to help them unlock the full potential of their data.

Doing so will ensure that clients capable of repayment are not rejected and that loans are given with a principal, maturity, and repayment calendar that will empower their clients to be successful.

Visit the following website to join the competition.

`https://www.kaggle.com/c/home-credit-default-risk/`