Walrus Co.
Introduction to Data Science 2023
Technical Report

Christian Cardin

Elva Granados Escartin

Julia Pukarinen

Abstract

In this technical report, we introduce the technicalities of our project for the course Introduction to Data Science 2023, and enter the details on few selected highlights.

1 Motivation

Our motivation for the project was to understand the situation of the economic cycle, based on US historical economic data, and give traders an insight of possible opportunities. Our purpose was to predict short-term performance of financial indices based on macroeconomic data and other market sector indices. We also wanted to learn the correlation between market indices and macroeconomic data.

2 Running the Code

We recommand having a version of Python 3.8 or superior, and an updated version of pip. Our project is hosted on GitHub at the address https://github.com/sechlol/intro_ds_2023. First of all, clone the repository. Then, locate the file `requirements.txt` in the project root folder and install the required packages with the command `pip install -r requirements.txt`. To run the code, simply call `python main.py`.

3 Software Architecture

The project is structured into distinct modules, each with a specific role, reflecting the best practices of clear programming. Breaking down the code into these separate units encourages modularity and encapsulation, which, in turn, enhances code clarity and makes it more understandable. This organized approach not only benefits human readability but also simplifies maintenance and troubleshooting, as modifications are confined to their respective modules, reducing unexpected side effects.

At the core of the codebase, the `main.py` file serves as the project's entry point, orchestrating the execution by sequentially calling these individual modules. This method promotes a clean and logical structure, enabling developers to navigate the codebase more easily.

1. Data Collection: Each data source is queried independently, resulting in an array of time-series dataframes. Each dataframe contains different finance data features, which is then combined together in a unique large dataframe. See Section 4 for more information on data collection.

2. Machine Learning: The dataset is passed to the `super_duper_ai.py` script, which contains multiple pipelines for the training and evaluation of machine learning and deep learning models. The module performs some independent manipulations on the dataset, like normalization and augmentation.

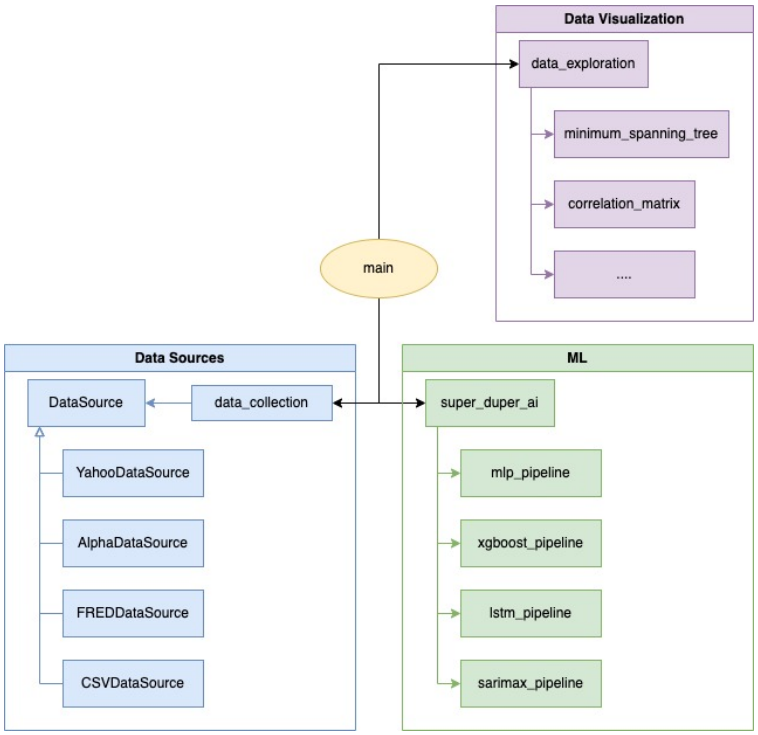

Figure 1: Simple view on project architecture

3. Data Visualization: From the original dataset, this module is responsible for extracting interesting insights from the data and create plots to be then analyzed.

A simple schematic view on the project software architecture is depicted in Fig. 1

4 Data Collection

4.1 DataSources

There is a great variety of services that offer high quality economic data API; however, the almost totality of them ask for a premium price to access their data. We identified a handful of services that, previous registration, allowed users to access their data free of charge, with limitations imposed by usage volume of their APIs. For the daily price action of the major market indices, we chose **Yahoo! Finance**[1], for which a convenient python library (`yfinance`[2]) is available to easily retrieve timeseries for stocks data given their unique identifier (ticker). From Yahoo finance we got the data on S&P 500 and the 11 industry sectors: Information Technology, Health Care, Financials, Consumer Discretionary, Communication Services, Industrials, Consumer Staples, Energy, Utilities, Real Estate, and Materials.

For macroeconomic data, we resorted to **FRED**[3] (Federal Reserve Economic Data), which contains a treasure of data from the United States dating back as early as the beginning of the 20^{th} century.

For special technical indicators, we identified **AlphaVantage**[4] as a valid source. Unfortunately, we encountered some troubles with the latter: their free tier strongly limits the frequency of queries that a user can make, so we had to keep the number of data features retrieved from this source quite low.

[1] https://finance.yahoo.com/
[2] https://pypi.org/project/yfinance/
[3] https://fred.stlouisfed.org/
[4] https://www.alphavantage.co/documentation/

5 Preprocessing

The data of the indices are presented as a function of time. The sampling frequency varies between indices. In particular, stock data come with a Business daily frequency (5 times per week), and macroeconomic data is usually released at monthly or quarterly cadence. We had to decide whether to collapse the daily data to monthly or expand the monthly data to daily in order to match the data collected at the two different sampling frequencies. We decided to expand the monthly data to daily by giving each day of the month the same value (forward filling). To keep a consistent dataset, we decided to drop each row with missing data, meaning that we trimmed the dataset to the earliest time for which we had complete data, until today. Unfortunately for us, the timespan of each economic feature varied greatly. Some features like GDP (Gross Domestic Product) were available starting from the early 1900, but some indices like Real Estate (XLRE) and Communication services (XLC) were only available starting from 2018, which would limit our historical timespan too much. We then decided to drop the last two indices, allowing us to collect data starting from 1998 onward.

6 Learning Tasks

To answer one of our research question "Why aren't ML experts all billionaires, already?" we tried to predict whether the stock market would be profitable in the near future. This resuted in a binary prediction problem: predict a 1 if the ticker of interest after X periods will be greater than today's price, else predict a 0 if the price will be lower. Both ticker and number of periods are variables that can be tuned at will. We focused mainly on the S&P 500 index (**SPY**), and tried a multitude of periods, ranging from 1 day to 2 months.

We implemented three algorithms, here we only discuss the hyperparameter tuning, while more general details on the algorithm and the results are highlighted in the blog.

XGBoost: using the dedicated python library.[5] In our specific implementation, we set several hyperparameters for XGBoost to optimize its performance. We use "binary:logistic" as the prediction objective, which means the model will output the probability of a positive return, ranging from 0 to 1. To evaluate the quality of predictions, we employ metrics such as "error" for binary classification error, "auc" for the Area under the curve, and "logloss" for the negative log likelihood. For training and model evaluation, we use a 5-fold cross-validation approach. The trees in XGBoost use the "hist" algorithm, which is a faster histogram-optimized approximate greedy algorithm, with specific settings such as "max_depth=3" and "min_child_weight=4." Early stopping is also implemented if there are no improvements in 20 rounds.

Multilayer Perceptron: from Sci-Kit package.[6] We employ a neural network architecture with 100 hidden layers. This deep architecture allows the model to learn intricate and hierarchical patterns from the data. The optimizer used is Stochastic Gradient Descent (SGD) with a constant learning rate. The batch size for training is set to 64, which determines how many data points are used in each iteration to update the model. Early stopping is a mechanism implemented to prevent overfitting and save computational resources. If there are no improvements in the model's performance for 20 consecutive iterations, the training process is terminated.

LSTM / GRU: "Long-Short term memory" and "Gated Recurrent Unit", from Keras.[7] In our experimentation, we explored various model architectures based on both LSTM and GRU networks. These models consisted of 50 layers with a hyperbolic tangent (tanh) activation function. The tanh activation function introduces non-linearity into the network, allowing it to capture complex patterns. The models included a unit normalization layer, which can enhance the training stability and convergence of the network. Each architecture culminated with a 3-layer dense classifier, providing the final prediction output. The optimizer utilized in this setup is Stochastic Gradient Descent (SGD)

[5] https://github.com/dmlc/xgboost
[6] https://scikit-learn.org/stable/index.html
[7] https://keras.io/

with a learning rate of 0.01. For loss computation, Binary Cross-Entropy is chosen as the loss function. This is a common choice for binary classification tasks, where the model evaluates how well its predictions match the actual outcomes. Binary Accuracy is used as the evaluation metric, assessing the model's ability to predict binary classifications accurately.

More information on these algorithms are found in the blog.

6.1 Challenges

Developing good ML algorithms is hard. It's easy to blindly train a model, and be fooled by the good performance in the training data. The biggest challenge for us was to find a good set of hyperparamters for our models to improve its inference accuracy on unseen test data. Ultimately, we were not able to reach any meaningful prediction that was better than random guess. This made us think that we should have spent more time experimenting with the features we included in our training dataset, rather then focusing on hyperparmeter tuning.

7 Exploratory Data Analysis

Correlation is a statistical measure that describes how two assets are related. It can give perspective on the overall nature of the market. We use rolling Pearson correlation with pandas. The window used for the rolling correlation was one month or one year. It depended on whether we were dealing with an index for which daily or monthly data was available. A monthly correlation could be calculated for the indices for which daily data was obtained. Although the monthly data was extended to daily, no monthly correlation could be calculated for those indices, because the value remained the same throughout the month. Thus, one year had to be selected as a window whenever there was an index with monthly data.

The scale varies between -1 and +1. Correlation of -1 indicates a perfectly linear negative correlation and +1 indicates a perfectly linear positive correlation. Correlation of 0 indicates that there is no linear dependency between the two assets. So the higher the positive correlation, the more strongly the two assets vary together, and the higher the negative correlation, the more strongly they vary together but in opposite directions.

8 Visualizations

We wanted to visualize indicators and market indices to look at their performances and correlations as well as the performance of the machine learning algorithms used.

We plotted basic plots of the performance of indices and markets in time, picking different combinations, depending on what features were being analyzed. On these basic plots the times of financial crises were highlighted in grey to add to the readability.

To show correlations we used correlation matrices and rolling correlations. We used Plotly's interactive Slider feature to help us show the rolling correlation as a function of time between two indices. We also used Plotly's interactive minimum spanning tree to show how all indices behave in relation to each other within a given time interval. Using the minimum spanning tree, we obtained information about which indices show the strongest and weakest correlations with each other, annualised returns and annualised volatility.

To assess the performance of the ML algorithms we plotted accuracy metrics and confusion matrices.

9 Communication of Results

The results obtained from the analysis and predictions are reported in a blog post, available at the GitHub-hosted address https://sechlol.github.io/intro_ds_2023/. The HTML pages are generated with Quarto[8], an open source scientific publishing system that automatically generates HTML pages from a markdown file, or Jupyter notebook.

[8] https://quarto.org/

In this text we communicate the results from the machine learning algorithms as well as some more general economic analysis of our data. For the sake of knowledge and speculation.

In the part where the results of the machine learning algorithms are considered, we explain the algorithms used and go through the results, comparing them together using accuracy metrics.

10 Final Considerations

Unfortunately, the outcome was not as desired. Our predictions were as good as a random guess, and we won't be getting our Lambos and Range Rovers any time soon.

So our end result cannot be said to be actionable, despite our best efforts. This was expected, because it is not so easy to predict the stock market, even taking into consideration a wide array of macroeconomic indicators. Indeed, if it was so easy everyone would do it, and in the blink of an eye, the efficiency of the markets would take care of terminating the party before things get out of hand. Nevertheless we wanted to try, because it seemed like an interesting learning opportunity, plus you never know...

Nevertheless, we managed to build a good framework for data collection, processing and analysis that can be used for further development, for example in the case of getting access to more data, or adding new ML models. For us all, it was an exemplary learning experience encompassing all steps in the life cycle of a data science project, from finding the data to presenting the end results.

It is worth mentioning that one component of our group was dealing with many of these topics for the first time, and gained an important amount of insights, from the details of using Git for collaborative coding to the core ideas in data science and even getting an excuse to study a little economic theory. So it was a great opportunity with a splendid learning outcome.

Another component of our group didn't know much about the subject before the project. They would definitely have gotten more out of the work and been able to make a wider contribution if they had understood the subject more. Git as an environment was completely new and learning its basics was very useful. Using classes with Python was new and interesting. It was also nice to recall some methods of data collection. Learning how to make interactive graphs was pleasant and it certainly will give great opportunities for presenting some results in the future.

Future opportunities may lie in focusing the analysis in more specific, and less watched areas of the markets, for finding lurking market inefficiencies, i. e. potential profits.