**Week 1**

This week let's mainly get hands-on and familiarize ourselves with the applications of data analysis in quantitative investing areas, along with sharing your more detailed personal learning and career goals with me. Also, you will be assigned a task working on data retrieval of financial data from a variety of different platforms, which is the basics of data analysis in the financial & quantitative investing/trading field. Following are the tasks for you to start whenever you have time:

Task 1

Firstly, please develop a deep understanding of what data analysis is and what responsibilities data analysts/scientists will need to cover and write a brief report including your own understanding combined with your expected future career goal. Please deliver a report covering the previous points and make sure references are listed at the end of the summary report. The aim of this project is to cultivate a profound and holistic understanding of the related field. This encompasses the roles, responsibilities, necessary skills, trends, and personal development pathways in this domain so that we can assign you more suitable projects in the future days. It also gives you an opportunity to ask yourself important career questions that you will need to get prepared for not only for joining this internship but also for your future career path. The report will need to cover the following points:

1. What is data analysis?

2. What responsibilities do data analysts/scientists cover?

3. What skillsets do data analysts/scientists need?

4. What is the trend of development in the data analysis field?

5. How can you improve to break into /succeed in this industry?

6. Future individual career goal in detail? The report will also need to cover:

- How does data analysis or what specific data analysis skillsets are used in the quantitative investing/trading field?

- General types of quant trading/investing strategies in the financial markets

- The history and development of quant trading/investing, and the future of quant trading/investing

- What are the advantages and disadvantages of quant investing/trading compared to discretionary trading/investing?

- What are the top 20 quantitative hedge funds in the world by 2023（if you can’t find data for 2023, please find the most recent years as you can) ranked by asset under management (AUM) and also please list their official websites’ links.

- What are some most important steps to develop a quantitative strategy?

- What is backtesting in the quantitative investing/trading areas? What are the advantages and disadvantages of backtesting? What are some common traps of backtesting? What are the difference between event-driven backtesting and vectorized backtesting?

- How can machine learning help with quantitative investing/trading? Please take some specific examples.

Task 2

In the meantime, through this week, please also start the following data retrieval task:

Please write Python code to retrieve data from different API sources, including:

1. Yahoo Finance (please retrieve any 10 stocks’ data)

2. Tushare (please retrieve 5 stocks’ data)

3. FRED (Federal Reserve Economic Data) (please retrieve 5 macroeconomic indicators’ data)

For Tushare, you can directly use the following token without needing a Chinese phone number to register an account. The token is as follows:

756a1f4aba6dd90ced168a81497f46697c8499ccfa76b317daf6e874.

Please send the task to me in the form of an ipython notebook showing all your code and their stock price plots/ data results on it. Please make sure all plots are conducted using Plotly (a powerful Python plotting package).

**Week 2**

This time, please write a research report related to NLP-driven sentiment analysis on the financial market including answering the following questions:

1. What are some of the most referred website links for sentiment analysis on the financial market? What assets do they cover? What general calculation methods do they expose to the public for getting the sentiment results/scores, if there are any?

2. How accurately do these sentiment analyses measure/predict the assets? Are there any proofs from these web links/sources you find that provide any backtested results or any statistical proofs validating their sentiment analysis's accuracy?

3. If there is any, please find Python code that uses either one source or multiple sources' sentiment or sentiment-related data to generate sentiment scores/results on financial assets, like stocks, ETFs, futures, etc. If there are any Python packages/modules serving similar functions, please also find them.

After collecting/writing down all sentiment web sources and Python code along with possible sentiment analysis results mentioned above, please write a brief summary at the end of the report elaborating your own insights and analysis indicating which web sources or Python code/packages/modules/results indicate the most or at least relatively solid sentiment analysis results on the financial market, preferably on the stock market. The report format is similar to those that you have done before.

Please send me two files this time: the report with your word elaborations and an ipython notebook including all Python code related to sentiment analysis or packages/modules and their application examples.

**Week 3**

Project Background

In the constantly evolving financial markets, understanding the intricate relationships between different stocks and major indices provides valuable insights into market dynamics and potential risks. Historically, the Dow Jones Industrial Average (DJIA) and the Nasdaq 100 Index have been two of the primary benchmarks that investors and analysts utilize to gauge market performance and sentiments. Each index comprises a different set of companies with its unique industry representation and risk factors. Correlation analysis has emerged as a powerful tool in portfolio management, risk mitigation, and investment strategy formulation. By examining how the daily returns of individual stocks correlate with these benchmark indices, we can glean insights into systemic risks, company-specific volatility, and overall market sensitivity.

Project Mission

Our project aims to employ Python to visually and quantitatively explore the correlation relationships between individual companies within the DJIA and Nasdaq 100 indices against their respective benchmark indices. You can use data sources like Yahoo Finance and their Python API to retrieve their historical daily price data.

Main Goal

Filtering and Analysis: Please develop a computational methodology to filter and identify companies that exhibit a high correlation (greater than 0.7) with their respective 2 indices, i.e. DJIA for Dow-listed companies and Nasdaq 100 for Nasdaq-listed companies, over the past three years. Do not cross-compare, meaning you can't correlate DJIA's stocks with the Nasdaq Index while filtering, and vice versa. In this process, you will also need to find live-updating sources that can provide the most up-to-date information about these 2 indices' stock tickers to get their price data. This will aid in pinpointing stocks that closely follow market movements and are potentially more exposed to systemic market risks.

Upon completion, our study will offer a clearer understanding of the interrelationships between stocks and their benchmark indices. This understanding is pivotal for investors looking to align their portfolios with market movements, hedge against systemic risks, or exploit stock-specific anomalies. The deliverables will be presented in a .py file for comprehensive examination and accessibility.

**Week 4**

Project Background

This new project revolves around the financial market, specifically Exchange-Traded Funds (ETFs). ETFs are investment funds and exchange-traded products that are traded on a stock exchange, like any other company stock. Each ETF has a ticker, a unique set of letters representing the fund listed on an exchange. The ticker serves as a shorthand way to reference the fund. In this case, we are interested in all available ETF tickers and their respective historical market data.

Project Mission

The main purpose of this project is to identify ETFs with specific characteristics that could potentially be valuable for investment strategies or financial modeling. We aim to develop a Python program that fetches all the available ETF tickers and their historical market data using an API provided by Tushare, a financial data platform.

We then intend to use this data to perform two distinct types of time-series analysis:

We aim to analyze the autocorrelation of the closing prices of these ETFs. Autocorrelation, also known as serial correlation, is the correlation of a signal with a delayed copy of itself as a function of the delay. In finance, it may reveal systematic trends in the price movement of an ETF over time.

We also plan to assess the stationarity of the ETFs' closing prices. A time series is said to be stationary if its statistical properties, such as mean and variance, remain constant over time. Non-stationary series can be challenging to model, so identifying such series could be valuable.

The project will also include developing functions to filter ETFs based on autocorrelation and stationarity criteria, which will help identify potential ETFs of interest.

By successfully completing this project, you will not only gain practical experience working with financial data and time-series analysis but also contribute to developing a tool that could significantly aid our investment decision-making process.

Project Requirement Details

Please go to Tushare (https://tushare.pro/) and find if there's a Python API through which we can write code to get all ETF's historical market data, which means their "DateTime," "open," "high," "low," "close," "volume.”

Please note that, first, you will need to find a source where we can get all available ETF tickers at one time (tickers are a series of number codes representing each asset in the financial market; for ETFs, one example would be 510020. For more definitions, please search from online sources, e.g. Wikipedia or Investopedia). The reason why we need a standard way of getting all ETF tickers all at once is that as time goes on, the ETFs listed or delisted on the market exchanges can change, thus we don't want to manually add or delete each time the pool changes, so there would be an API for us to get all most updated tickers once and for all.

Next, please write code that has the following functions:

1. Function 1: Test all ETFs' autocorrelation on their "close" price data by creating a filter to select those that have strong autocorrelation. The input of the function would be a manually set autocorrelation threshold value, and the output would be a data frame with columns of selected ETFs' tickers, autocorrelation values, start time, and end time. Start and end times represent the period of data that you use to do the calculations.

2. Function 2: Test all ETF's stationarity, filter, and select all those that are not stationary for their "close" price data. The output would be a data frame with columns of selected ETFs' tickers, start time, and end time. Start and end times represent the period of data that you use to do the calculations. Also, please use ADF as the stationarity test tool.

3. Function 3: Take the intersection of selected ETF tickers of both Function 1 and Function 2 and return their commonly selected ETF tickers.

Work Delivery: A Python .py file.

**Week 5**

Project Background

In the evolving landscape of financial markets, Exchange-Traded Funds (ETFs) have emerged as a popular investment vehicle, offering diversified exposure across various sectors and asset classes. As our company ventures into building an advanced ETF price movement monitor and analytical pipeline, we recognize the need for historical and real-time data acquisition and analysis. This project will involve leveraging Python to dynamically retrieve and process a comprehensive list of US ETFs and their historical daily price data for the most recent 1 year.

Project Mission

The objective of this project is twofold: firstly, to develop a Python script capable of retrieving a live-updating list of all US ETF tickers from reliable online sources. Secondly, to create a separate script for downloading the daily price data of these ETFs. This will enable us to monitor market trends, analyze ETF performances, and make informed investment decisions. Your work will contribute significantly to the development of our ETF analytical pipeline, enhancing our ability to offer timely and data-driven insights to our clients.

Project Requirements

1. Ticker Retrieval Script

- Develop a Python script to extract a live list of US ETF tickers from online sources like Finviz or Morningstar, or other credible online sources.

- The source should be dynamic, providing real-time updates rather than static files.

- Implement error handling and efficient data retrieval methods.

2. Price Data Download Script

- Create a Python script to download daily price data (e.g., open, high, low, close, volume) for the ETFs identified, with the period of the most recent 1-year daily data for each ETF.

- Utilize sources like Yahoo Finance for data download.

- Incorporate a retry mechanism to handle potential download failures or API instability, ensuring complete data retrieval.

- The script should manage the frequency of requests to avoid issues with the data source’s rate limits.

- Please make sure your Python code can download all the data, either saving it into a SQLite database or exporting it to csv files

3. Data Quality Validation

Considering the large number of ETF tickers and the extensive volume of datasets involved in your project, it's essential to implement a thorough preliminary check on the data's quality. Here's an additional suggestion focusing on the use of statistical descriptions and visualizations:

Implement Comprehensive Statistical Descriptions and Visualizations:

- Descriptive Statistics: Start by generating descriptive statistics for each ETF dataset. This includes measures like mean, median, mode, standard deviation, minimum and maximum values for each attribute (e.g., open, high, low, and close prices). Descriptive statistics will provide a quick overview of the data's distribution and identify any anomalies like extremely high or low values that may indicate data errors.

- Histograms and Boxplots: For each ETF, create histograms and boxplots of price data (open, high, low, close). Histograms will help in understanding the distribution and spotting any skewness or unusual patterns. Boxplots are useful for quickly visualizing the range of data and identifying potential outliers.

- Time Series Plots: Build a function to allow user to plot the time series data for any requested ETF that has been downloaded. This will help in visually inspecting the data for any inconsistencies, gaps, or unusual spikes that may not be obvious in the numerical summaries.

- Missing Data Analysis: Conduct an analysis to check for missing data.

- Automated Alerts for Data Anomalies: Implement an automated system that flags data points that are statistical outliers or fall outside predefined thresholds. For example, this could be based on z-scores or other statistical measures.

By incorporating these statistical descriptions and visualizations above into your preliminary data checks, you can significantly enhance the quality assurance process. It will enable you to identify and address potential data quality issues effectively before they impact any further analysis or decision-making processes.

4. Documentation and Code Efficiency

- Both scripts should be well-documented, with clear comments and readable code.

- Include error handling and data validation to ensure the reliability of the scripts.

Suggested Project Steps

1. Research and identify potential online sources for live ETF ticker data.

2. Develop the Python script for ETF ticker retrieval, ensuring dynamic updates and efficient data processing.

3. Identify a reliable source for daily ETF price data and develop the downloading script with appropriate retry and rate-limit handling mechanisms.

4. Test the scripts thoroughly to ensure accuracy and reliability.

5. Document the code and provide a brief user guide or comments within the script for ease of use and understanding.

Additional Information for your Reference for your Better Understanding:

1. Information to Extract for ETF Tickers

- The primary goal of the first script is to retrieve a list of US ETF tickers. A ticker, in this context, is the unique symbol used to identify an ETF on the stock exchange. For example, "SPY" is the ticker for the SPDR S&P 500 ETF Trust.

- Apart from the ticker symbols, it would be beneficial if you could also extract additional basic information about each ETF, if available, from your data source. This might include:

a. ETF Name: The full name of the ETF.

b. ETF Category or Sector: The market segment or sector the ETF focuses on, such as technology or healthcare.

c. Asset Manager: The firm managing the ETF.

d. However, the focus should primarily be on retrieving a comprehensive and up-to-date list of ticker symbols. Additional information is useful, but the tickers are the priority.

2. Dynamics of ETF Tickers:

- ETF tickers themselves are not dynamic; they are fixed symbols assigned to each fund. However, the list of ETFs is dynamic in the sense that new ETFs are frequently launched, and some are delisted. Therefore, we need a dynamic method to retrieve this list, ensuring it reflects the most current ETFs available in the market.

- Manually creating a list of ticker names is not ideal for this project. We aim to develop a script that can automatically fetch and update this list from online sources. This way, our dataset remains current without manual intervention.

- The script should be designed to periodically query an online source (like Finviz, Morningstar, or other sources) to obtain an updated list of all US ETFs. These sources typically have pages or APIs that list all ETFs they track, including newly listed ones.

It is essential to create a Python script that can periodically connect to these sources, extract the list of ETF tickers (and additional information if possible), and ensure this list is up to date. This dynamic approach is crucial as it allows our system to adapt to changes in the ETF market automatically.

I hope this helps clear up your questions and potential concerns. If you have any more questions or need further assistance, please don't hesitate to reach out. We are confident in your abilities and look forward to your progress on this project!

Deliverables:

1. One Python script (.py file): Both for retrieving ETF tickers and for downloading their daily price data.

2. A brief documentation or in-code comments explaining the functionality and usage of the scripts.