

Milestone 2: Billionaire Secretary

Team Billionaires: Huandong Chang, Mingyuan Ma, Lance Lu, April Zhang

Virtual Environment Setup

Our virtual environment setup is included in the Dockerfile of each component. We used Pipenv with `Pipfile` and `Pipfile.lock` to manage dependencies.

Data Pipeline

```
(base) aprilzxd@DESKTOP-JKSGKNH:/mnt/c/Study@Harvard/AC215/AC215-project/src/datapipeline$ sh docker-shell.sh
[+] Building 0.5s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 275B                             0.0s
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim 0.3s
=> [1/6] FROM docker.io/library/python:3.11-slim@sha256:a9476aa83bc40fce9b019b8e5b6ff3f855ae5bdd2460f60173ae405c 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 306B                                  0.0s
=> CACHED [2/6] WORKDIR /app                                    0.0s
=> CACHED [3/6] COPY Pipfile Pipfile.lock /app/                 0.0s
=> CACHED [4/6] RUN pip install pipenv                          0.0s
=> CACHED [5/6] RUN pipenv lock && pipenv sync                  0.0s
=> CACHED [6/6] COPY . /app                                     0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:d0005ed38ccad7333638a9443f072c1b74400d1b95f1c871ce8881249b344ebb 0.0s
=> => naming to docker.io/library/datapipeline                  0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
Launching subshell in virtual environment...
root@9a4dfed44173:/app# source /root/.local/share/virtualenvs/app-4PLAip0Q/bin/activate
(app) root@9a4dfed44173:/app# |
```

Gemini Finetuner

```
(base) aprilzxd@DESKTOP-JKSGKNH:/mnt/c/Study@Harvard/AC215/AC215-project/src/gemini-finetuner$ bash docker-shell.sh
[+] Building 1.0s (12/12) FINISHED                                docker:default
=> [internal] load .dockerignore                               0.0s
=> => transferring context: 2B                                    0.0s
=> [internal] load build definition from Dockerfile              0.0s
=> => transferring dockerfile: 1.56kB                             0.0s
=> [internal] load metadata for docker.io/library/python:3.9-slim-buster 0.8s
=> [auth] library/python:pull token for registry-1.docker.io    0.0s
=> [1/6] FROM docker.io/library/python:3.9-slim-buster@sha256:320a7a4250aba4249f458872adecf92eea88dc6abd2d76dc5c 0.0s
=> [internal] load build context                                0.0s
=> => transferring context: 220B                                  0.0s
=> CACHED [2/6] RUN set -ex; for i in $(seq 1 8); do mkdir -p "/usr/share/man/man${i}"; done && apt-get 0.0s
=> CACHED [3/6] WORKDIR /app                                    0.0s
=> CACHED [4/6] ADD --chown=app:app Pipfile Pipfile.lock /app/ 0.0s
=> CACHED [5/6] RUN pipenv sync                                  0.0s
=> CACHED [6/6] ADD --chown=app:app . /app                      0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:f4c5b287295fe40ce4a576d8567ce984a5d197406a7e04e01b99aa6042d115c0 0.0s
=> => naming to docker.io/library/gemini-finetuner              0.0s

What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
Container is running!!!

Launching subshell in virtual environment...
app@cee7e818a9a0:/app$ source /home/app/.local/share/virtualenvs/app-4PLAip0Q/bin/activate
(app) app@cee7e818a9a0:/app$ |
```

Agents

```
(base) aprilzxd@DESKTOP-JKSGKNH: /mnt/c/Study@Harvard/AC215/AC215-project/src/agents$ sh docker-shell.sh
[+] Building 1.0s (11/11) FINISHED                                docker:default
=> [internal] load build definition from Dockerfile                0.1s
=> => transferring dockerfile: 321B                               0.0s
=> [internal] load .dockerignore                                  0.1s
=> => transferring context: 2B                                      0.0s
=> [internal] load metadata for docker.io/library/python:3.11-slim-buster 0.6s
=> [1/6] FROM docker.io/library/python:3.11-slim-buster@sha256:c46b0ae5728c2247b99903098ade3176a58e274d9c7d2efea 0.0s
=> [internal] load build context                                0.1s
=> => transferring context: 526B                                    0.1s
=> CACHED [2/6] RUN set -ex; pip install --no-cache-dir --upgrade pip && pip install pipenv 0.0s
=> CACHED [3/6] WORKDIR /app                                     0.0s
=> CACHED [4/6] COPY Pipfile Pipfile.lock /app/                 0.0s
=> CACHED [5/6] RUN pipenv install --deploy --ignore-pipfile    0.0s
=> CACHED [6/6] COPY . /app                                     0.0s
=> exporting to image                                           0.0s
=> => exporting layers                                           0.0s
=> => writing image sha256:3418854bb9fe35albc289fe4e326cb4c937cde4ddddd5dee94a430ac6677ba063 0.0s
=> => naming to docker.io/library/agents                        0.0s



What's Next?
View a summary of image vulnerabilities and recommendations → docker scout quickview
root@2537fc6e3c23:/app#
```

Containerized Components

As shown in the screenshots above, the data pipeline, Gemini fine-tuner, and agent components each run in their own container. The documentation and setup instructions for these containers are provided in the README file in our repository.

Versioned Data Strategy

For our project, we use a single set of text data under 1GB for fine-tuning, so we determined that detailed data versioning is unnecessary. However, as a precaution, we have enabled object versioning in Google Cloud Storage and will use GitHub to track changes to our prompt versions.

Object version ↓	Generation	MD5 hash	CRC32C hash	Storage class	Size	
 train.jsonl (Live object)	1729283713762454	5a85fab9f8f15ffd77a38072630199ed	2779461978	Standard	1.1 MB	
 Oct 18, 2024, 2:59:21 PM	1729277961780941	5a85fab9f8f15ffd77a38072630199ed	2779461978	Standard	1.1 MB	Restore ⋮

Model Finetuning

We used the [reddit_finance_43_250k](#) dataset, with a more detailed description available in the README file of our repository. For the current implementation, we fine-tuned the gemini-1.5-flash-002 model on a small subset of the data for 1 epoch as a proof of concept. Moving forward, we plan to use a larger dataset and train the model for additional epochs.


Here's a list of hyperparameters we used in our current implementation:

- **max_output_tokens:** 3000 (defines the maximum number of tokens that the model can generate in a single response, controlling the length of the output)
- **temperature:** 0.75 (controls the randomness of the model's responses)
- **top_p:** 0.95 (enables nucleus sampling, where the model samples tokens from the top 95% of the cumulative probability distribution, encouraging more diverse yet relevant outputs)
- **epochs:** 1 (will increase this moving forward)
- **adapter_size:** 4 (controls the size of the adapter layers used during fine-tuning, influencing the model's capacity to adapt to new data)
- **learning_rate_multiplier:** 1 (scales the base learning rate for model training)

Agent Setup

Instead of using RAG, our team opted to leverage an agent-based approach. Our chatbot integrates with tools like YFinance for market data, Portfolio Volatility for risk analysis, and StockPlotter for visualizing stock trends. Additionally, it offers features such as sending financial reports to multiple recipients via email and tracking earnings calendars. This agent-based solution has enhanced the chatbot's ability to deliver seamless financial research and decision-making through a user-friendly chat interface.

Mock-up of the Application

CorrelationMatrix between NVIDIA, Capital One, Amazon, TikTok, Apple, Microsoft

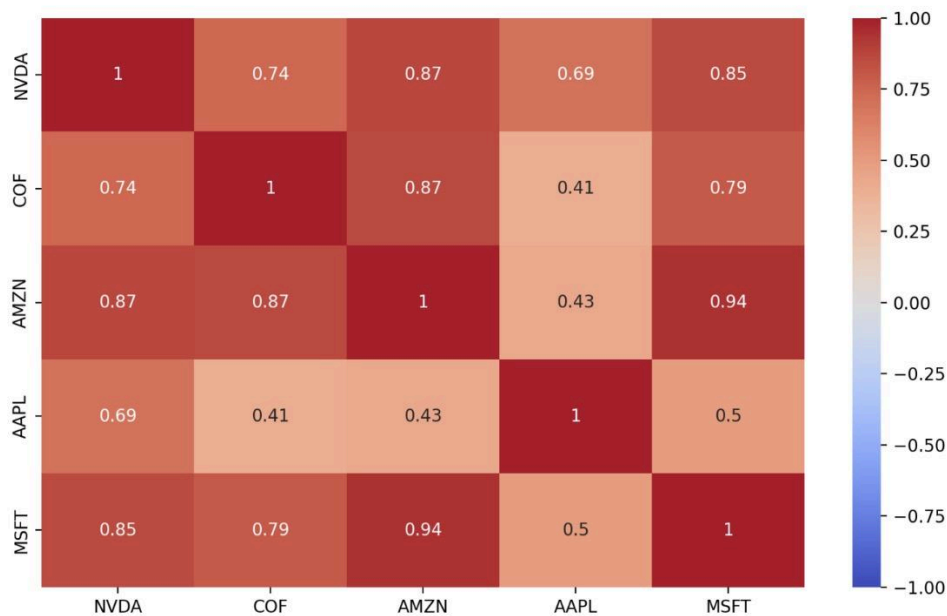
Summary of the Task

- The correlation matrix was successfully plotted for the following companies: **NVIDIA (NVDA)**, **Capital One (COF)**, **Amazon (AMZN)**, **Apple (AAPL)**, and **Microsoft (MSFT)**.
- **TikTok** was excluded from the analysis as it is not a publicly traded company.

The correlation matrix has been displayed successfully, indicating how the stock prices of these companies are related over the specified date range.

Task Status

Task Complete





please tell me the earnings dates for NVIDIA and Capital One and Amazon



The earnings dates for NVIDIA (NVDA), Capital One (COF), and Amazon (AMZN) have been successfully fetched. If you need further details or have any other questions, feel free to ask!

Earnings Date	EPS Estimate	Reported EPS	Surprise(%)
2025-08-26 06:00:00-04:00	None	None	None
2025-05-20 16:00:00-04:00	None	None	None
2025-02-19 16:00:00-05:00	None	None	None
2024-11-19 16:00:00-05:00	0.74	None	None
2024-08-28 16:00:00-04:00	0.64	0.68	0.0603
2024-05-22 16:00:00-04:00	0.56	0.61	0.0948
2024-02-21 16:00:00-05:00	4.64	5.16	0.1132
2023-11-21 16:00:00-05:00	3.37	4.02	0.1921
2023-08-23 16:00:00-04:00	2.09	2.7	0.2941
2023-05-24 16:00:00-04:00	0.92	1.09	0.1857

Earnings Date	EPS Estimate	Reported EPS	Surprise(%)
2025-07-21 06:00:00-04:00	None	None	None
2025-04-23 16:00:00-04:00	None	None	None
2025-01-23 16:00:00-05:00	None	None	None
2024-10-24 16:00:00-04:00	3.76	None	None
2024-07-22 16:00:00-04:00	2.20	2.14	0.0451



I want to see stock price volatility for NVIDIA within in a month



Summary of the Task

- The stock price volatility for **NVIDIA (NVDA)** has been successfully plotted for the past month (from 2024-09-17 to 2024-10-17).

Task Status

Task Complete

Rolling Volatility (7-day kernel, 1-day stride) from 2024-09-17 to 2024-10-17

