Project Scoping Submission - FinSight

Team Members:

1. Aditya Sai Saandeep Prayaga
2. Vishwaben Umeshbhai Bhadiyadara
3. Malav Makadia
4. Gaurav Ashvinbhai Ramoliya

1. Introduction

In today's fast-paced financial markets, leveraging advanced technologies to analyze and predict stock performance is crucial for investors and financial analysts. This project, **FinSight: Adaptive Stock Prediction and Monitoring System**, aims to harness the power of machine learning and deep neural networks to provide accurate and timely stock predictions. By utilizing the last 10 years of stock data and a company's financial data, this system integrates data cleaning and preprocessing with training and testing models to enhance efficiency and model accuracy.

The primary goal of FinSight is to create a robust framework that not only delivers high accuracy and precision in stock predictions but also continuously monitors model performance. Based on key metrics, such as accuracy and precision, the system can trigger alerts and initiate retraining processes to maintain optimal performance. This adaptive approach ensures that the predictive models remain relevant and effective in the ever-evolving financial landscape.

2. Dataset Information

1. Dataset Introduction

The dataset consists of Netflix's stock price data from 2002 to 2022. It includes daily records of the stock's opening, closing, highest, and lowest prices, along with the trading volume. This dataset is crucial for analyzing the historical performance of Netflix's stock and predicting future trends.

2. Data Card

- **Size:** 348.27 KB
- **Format:** CSV
- **Shape:** 5031 rows × 6 columns

Variable Name	Data Type	Description
Date	Date	The date of the stock price record
Open	Float	The opening price of the stock for the day
High	Float	The highest price of the stock during the day
Low	Float	The lowest price of the stock during the day

Close	Float	The closing price of the stock for the day
Volume	Integer	The total trading volume of the stock for the day

Label Classes and Descriptions:

Label Class	Description
Open	The price at which the stock opened trading on a given day
High	The highest price the stock reached during trading hours on a given day
Low	The lowest price the stock fell to during trading hours on a given day
Close	The price at which the stock closed trading on a given day
Volume	The total number of shares traded during the trading hours on a given day

3. Data Sources

The dataset is sourced from Yahoo Finance: Netflix Stock Price Data

4. Data Rights and Privacy

The Netflix stock price dataset is publicly available and sourced from Yahoo Finance. This data is considered public domain as it pertains to publicly traded companies and is freely accessible to anyone interested in financial markets. Here are the key points regarding data rights and privacy:

1. Public Domain Data:
The dataset consists of historical stock prices, which are public information. Such data is regularly published by stock exchanges and financial news websites, including Yahoo Finance. As a result, there are no proprietary restrictions on its use.

2. No Personal Data:
The dataset does not contain any personal or sensitive information about individuals. It solely includes financial data related to Netflix's stock prices, ensuring compliance with privacy regulations.

3. Compliance with GDPR:
The General Data Protection Regulation (GDPR) focuses on the protection of personal data of individuals within the European Union. Since the dataset in question does not contain any personal data, GDPR compliance is not a concern.

However, users of the dataset should still adhere to good data management practices to ensure data integrity and ethical use.

3. Data Planning and Splits

In this project, we will handle the specific challenges presented by our particular time series dataset, which includes historical stock prices and related market indicators. Our data preprocessing steps will be thorough to ensure high data quality for model training.

Handling Outliers:

- Outliers in stock prices can significantly impact model performance. We will use statistical methods such as the IQR (Interquartile Range) method or Z-score analysis to detect and handle outliers. Depending on the analysis, we may cap outliers to a certain threshold or use robust scaling techniques to minimize their impact.

Handling Duplicates:

- Duplicate entries can skew the model's understanding of the data distribution. We will identify and remove any duplicate records based on unique identifiers such as date and stock symbol to ensure the dataset's integrity.

Handling Missing Values:

- Missing data is a common issue in stock price datasets. We will employ various imputation techniques such as forward filling, backward filling, or using statistical methods like mean/median imputation to handle missing values. In cases where imputation is not feasible, we might drop the rows or columns with excessive missing data after careful evaluation.

Data Schema and Validation:

- We will define a clear data schema, including data types for each feature (e.g., date, open price, close price, volume, etc.). Schema validation will be performed to ensure data consistency and correctness before further processing. Any deviations from the schema will be flagged for correction.

Data Rescaling:

- To ensure that all features contribute equally to the model, we will apply MinMax scaling to rescale the data. This method scales each feature to a given range, typically [0, 1], which is particularly important for algorithms like LSTM that are sensitive to the scale of input data.

Data Summary and Exploratory Analysis:

- A comprehensive summary of the dataset will be generated, including descriptive statistics (mean, median, standard deviation, etc.) and visualizations (histograms, box plots, time series plots). This step will help us understand the data distribution, identify patterns, and spot any anomalies.

Data Splitting:

- The dataset will be split into three distinct sets:
  - **Training Set (70%)**: Used to train the LSTM model.
  - **Validation Set (10%)**: Used to tune hyperparameters and prevent overfitting during model training.
  - **Testing Set (20%)**: Used to evaluate the model's performance on unseen data. This split ensures that the model's evaluation reflects its performance in real-world scenarios.

4. GitHub Repository

➔ Here is a GitHub repository and below is the folder structure.

Using this as Cookiecutter to generate project outline.

```
├── LICENSE
├── README.md          <- The top-level README for developers
using this project
├── data
│   ├── external       <- Data from third party sources.
│   ├── interim        <- Intermediate data that has been tra
nsformed.
│   ├── processed      <- The final, canonical data sets for
modeling.
```

```
│   └── raw            <- The original, immutable data dump.
│
├── docs               <- A default Sphinx project; see sphin
x-doc.org for detai  ls
│
├── models             <- Trained and serialized models, mode
l predictions, or model summaries
│
├── notebooks          <- Jupyter notebooks. Naming conventio
n is a number (for
│                         ordering),the creator's initials, a
nd a short `-` deli
│                         mited description, e.g.`1.0-jqp-ini
tial-data-explorati
|                         on`.
│
├── references         <- Data dictionaries, manuals, and all
other explanatory |                        materials.
│
├── reports            <- Generated analysis as HTML, PDF, La
TeX, etc.
│   └── figures        <- Generated graphics and figures to b
e used in reporting
│
├── requirements.txt   <- The requirements file for reproduci
ng the analysis env|                        ironment, e.g. g
enerated with `pip freeze > requiremen|
ts.txt`
│
├── setup.py           <- Make this project pip installable w
ith pip install -e
├── src                <- Source code for use in this projec
t.
│   ├── __init__.py    <- Makes src a Python module
│   │
│   ├── data           <- Scripts to download or generate dat
```

```
a
│   │   └── make_dataset.py
│   │
│   ├── features       <- Scripts to turn raw data into featu
res for modeling
│   │   └── build_features.py
│   │
│   ├── models         <- Scripts to train models and then us
e trained models to |   |   |                 make prediction
s
│   │   │
│   │   ├── predict_model.py
│   │   └── train_model.py
│   │
│   └── visualization  <- Scripts to create exploratory and r
esults oriented vis|       |                 ualizations
│       └── visualize.py
│
└── tox.ini            <- tox file with settings for running
tox; see tox.readth  |                          edocs.io
```

● README: README file includes essential project information, installation instructions, and usage guidelines.

5. Project Scope

Defining the project scope for a age old prediction very hard problem is a must, as the level of difficulty we are trying to solve is baseline and focus more on building a robust pipeline. Below is the problem statement and the current and proposed approaches:

1. Problems

Stock Price Prediction is a really hard problem labelled according to the module. The ground truth changes according with time(day, hour, minute) and the choice to buy and sell changes. This problem till date is being tackled by

Mathematicians, Scientists from all Spectrums, but the problem holds ground for its dynamic nature. This is because stock market in line with buy, sell which are gauge for supply and demand has also effect of company financials, logistics etc. For our case we are planning the prediction with a day duration data and concentrate only on open and close ranges not on entire company's financials.

2. Current Solutions

For our Netflix stock price prediction project, **current solutions** utilize a **simpler model**, such as Linear Regression, to forecast stock prices. While this model offers the advantage of being straightforward and easy to interpret, it **lacks the precision** and sophistication of more complex algorithms. As a result, its predictions are less accurate, often failing to capture the nuances and volatility of the stock market. Additionally, current solutions **does not include a robust pipeline** for retraining, monitoring and alerting systems for the model. Without these critical components, the model's performance may degrade over time as market conditions change, and there is no systematic way to update it with new data. This simplicity, while beneficial for initial implementation and ease of understanding, ultimately limits our ability to provide reliable and timely predictions.

3. Proposed Solutions (a better model with robust pipeline)

As an improvement over above solution, for our Netflix stock price prediction project, we propose transitioning to a **more complex model**, such as **Long Short-Term Memory (LSTM)**, which is better suited **to capture the intricate patterns** and volatility of stock prices. To enhance the model's effectiveness, we will **develop a robust pipeline** that includes capabilities for continuous retraining and real-time monitoring. This pipeline will leverage MLOps tools like Cloud Composer for orchestrating workflows using Apache Airflow in Google Cloud Platform (GCP), MLflow for tracking model metrics and performance, and Airflow triggers for automated alerting systems. These enhancements will ensure that our model remains accurate and responsive to market changes, providing more reliable predictions and allowing us to quickly address any issues that arise.

6. Current Approach

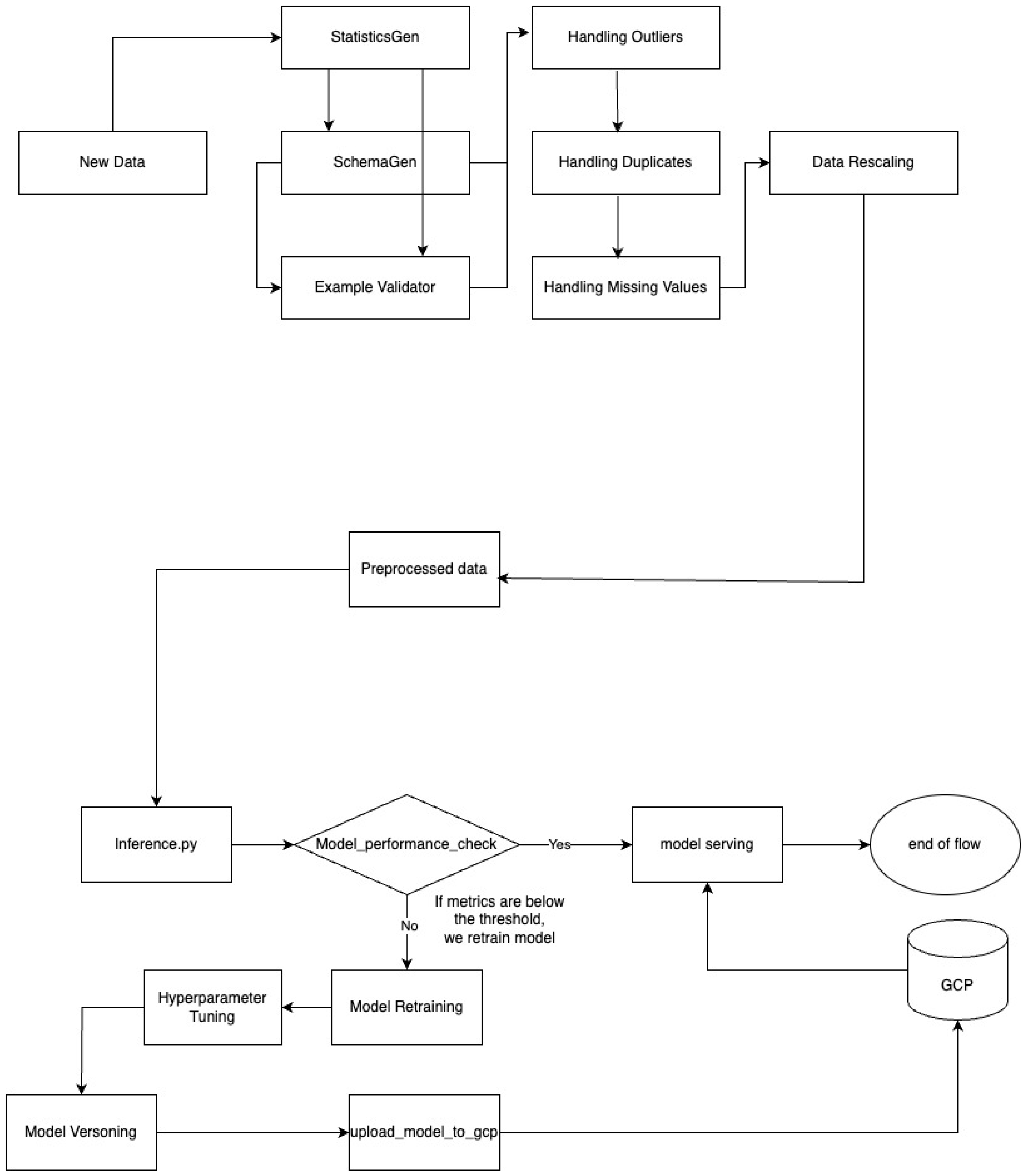
1. Data Ingestion

Data ingestion involves collecting data from various sources, such as historical stock prices, market indicators, and possibly social media sentiment. This data is stored in Google Cloud Storage (GCS), which provides scalable and durable storage for large datasets.

- Handling large volumes of data and high-speed data streams might overwhelm the system.
- Potential approach is to use **Parallel Data Loading or Batch Processing.**

2. Data Preprocessing

Data preprocessing includes cleaning the data, handling missing values, normalizing or standardizing features, and feature engineering. Apache Spark, running on Google Dataproc, provides a robust framework for distributed data processing.



3. Model Training

Model training involves training an LSTM model on the preprocessed data with help of packages like Keras/pytorch/tensorflow . Google Compute Engine provides the computational resources needed, especially GPUs, to train complex models efficiently.

- LSTM models are computationally intensive and may take a long time to train, especially with large datasets.
- If the dataset is large, **distribute the training process** across multiple machines.

4. Model Evaluation

Model evaluation includes calculating performance metrics like accuracy, precision, recall, and F1 score. This step ensures the model is performing well before deployment.

- Setting up trigger for retraining with decreased performance with respective to any metric is a tricky bottleneck step in the pipeline

5. Model Deployment

Deploy the trained model to Google Kubernetes Engine (GKE) or Google Cloud Run for serving real-time predictions. Monitor model performance and system health in real-time using MLFlow for tracking experiments, Kibana for visualizing logs, and Google Cloud Logging (Stackdriver) for log management. Establish a robust retraining path in the pipeline with well-thought triggers for various scenarios, such as significant drops in accuracy or precision. Use Apache Airflow (Cloud Composer) for scheduling periodic retraining and triggering retraining based on specific conditions.

7. Metrics, Objectives, and Business Goals

- **Metrics:** There are many aspects of Model and Pipeline metrics that can be tracked, but to the time being and current understanding below are few main metrics to be considered.
    - **Model Performance Metrics:**
        - **Accuracy:** Accuracy measures the proportion of correct predictions out of the total predictions made. It is defined as:

$$Accuracy = \frac{Number\ of\ Correct\ Predictions}{Total\ Number\ of\ Predictions\ Accuracy}$$

In the context of stock price prediction, accuracy indicates how often the model's predictions (e.g., the predicted direction of stock price movement) are correct compared to actual outcomes.

- **Precision:** Precision measures the proportion of true positive predictions out of all positive predictions made. It is defined as:

$$Precision = \frac{True\ Positives}{True\ Positives\ +\ False\ Positives}$$

For stock price prediction, precision indicates how often the model correctly predicts a specific event (e.g., a rise in stock price) out of all instances where it predicted that event.

- **Pipeline Performance Metrics:**
    - **Pipeline Efficiency and Responsiveness (PER) Metric:** The Pipeline Efficiency and Responsiveness (PER) metric evaluates the overall effectiveness and timeliness of the model retraining and deployment process, alongside the comprehensiveness of monitoring and alerting systems.
    - **Latency:** The time taken for each stage or component of the pipeline to complete. For example, the time taken for data preprocessing, model training, or making a prediction. Lower latency indicates a more efficient pipeline.
    - **Throughput:** Number of jobs processed by the pipeline per unit time. High throughput indicates the pipeline's ability to handle a large volume of jobs efficiently.
    - **Resource Utilization:** The CPU, memory, and disk usage during the pipeline execution. Monitoring resource utilization helps in optimizing the pipeline for cost-effectiveness and scalability.

- **Business Goals:** Create a robust & extendable pipeline with LSTM model for Netflix Stock Prediction Use case.

    - **Enhanced Predictive Accuracy**
        - Utilize LSTM model to capture complex patterns in stock price movements.
        - Ensure the model adapts to new data and market conditions through scheduled retraining.
    - **Automated and Efficient Pipeline**
        - Implement automated data ingestion, preprocessing, model training, evaluation, and deployment.
        - Reduce manual intervention and streamline the entire workflow.
    - **Comprehensive Monitoring and Alerting**
        - Ensure that deployment processes are reliable and repeatable.
        - Use tools like Cloud Composer for Airflow in GCP and MLflow for monitoring model metrics.
        - Set up Airflow triggers for real-time alerting to quickly respond to anomalies or deviations.

8. Failure Analysis

Below are few points we think are critical risk potential points which are also accompanied with the mitigations steps:

1. Model Underperformance:

- Stock price prediction is inherently challenging due to market volatility and unpredictability. Even complex models like LSTM may underperform.
- Establishing a robust retraining pipeline with clearly defined triggers that activate retraining processes when performance metrics (like accuracy or precision) fall below a certain threshold, ensuring the model adapts to new market conditions.

2. Load Failures During Training:

- Training complex models like LSTM can be resource-intensive, potentially leading to load failures.

- Implementing proper resource allocation strategies and use scalable infrastructure. Utilize autoscaling features of Google Compute Engine (GCE) and GPU instances for efficient training to manage resource demands.

3. Data Ingestion Failures:

- Issues with data ingestion, such as data source changes or interruptions, can impact the entire pipeline.
- Implementing data validation checks and redundancy in data sources to ensure continuous data availability and integrity. Regularly validate input data and handle corrupted data gracefully.

4. Pipeline Orchestration Failures:

- Failures in pipeline orchestration (e.g., Airflow or Cloud Composer) can halt the entire workflow.
- Setting up redundancy and failover mechanisms for the orchestrators, conduct regular health checks, and establish automated alerting systems to detect and address issues promptly.

5. Monitoring and Alerting Gaps:

- Inadequate monitoring and alerting can lead to undetected failures and performance degradation.
- Ensuring comprehensive monitoring using Stackdriver and Kibana, and set up automated alerts for key metrics such as accuracy, precision, resource utilization, and latency. This helps in detecting anomalies and addressing issues in real-time.

9. Deployment Infrastructure

- **Development Phase:** During the initial development phase, we plan to use local systems for deploying and testing the model. This allows for rapid prototyping and troubleshooting.
    - In local systems we try to use all the softwares the were taught in class like airflow, MLFlow, Kibana, etc.,
    - Ensuring that local systems have reasonable compute power to train, evaluate, test, and serve the model artifacts effectively.

- **Production Deployment:** Once we have GCP credentials, the focus will shift to production deployment, utilizing the robust infrastructure provided by GCP.
    - **Cloud Composer:** Use Cloud Composer to manage and schedule workflows. This will help automate and orchestrate the entire ML pipeline, from data ingestion to model deployment.
    - **Google Cloud Storage (GCS):** Store datasets, model artifacts, and other necessary files in GCS for secure and scalable storage solutions.
    - **Google Kubernetes Engine (GKE):** Deploy the model on GKE to ensure scalable and resilient production deployment. GKE allows us to leverage container orchestration, making it easier to manage and scale applications.
    - **Google Compute Engine (GCE):** Once we move to production, GCE will be our primary compute platform. GCE provides scalable virtual machines (VMs) that can be configured with the necessary compute power, memory, and GPU capabilities to handle the intensive training and inference tasks of our LSTM model.
        - **GPU Utilization:** Explore the possibility of utilizing GPUs on GCP for training the LSTM model. This can significantly speed up the training process and improve overall model performance
    - **Stackdriver Logging and Monitoring:** Integrated with GCE, Stackdriver will provide comprehensive logging and monitoring. We will set up alerts for key metrics to detect and respond to issues in real time.
    - **Kibana and Elastic Stack:** For advanced log management, we will use Kibana to visualize logs and set up dashboards that help in monitoring the health of the system.

10. Monitoring Plan

Monitoring is a critical aspect of our project, ensuring the reliability and performance of our stock price prediction system. We employ a comprehensive monitoring plan utilizing Airflow, MLflow dashboards, Github Actions and GCP services to monitor various components of the pipeline.

- **Airflow:** Utilize Airflow for workflow orchestration and scheduling, enabling the monitoring of pipeline execution and task statuses. Implement alerting mechanisms via email notifications and Slack channels to notify stakeholders or team members of any anomalies or performance degradation.

- **MLflow:** Leverage MLflow for model tracking and visualization, providing insights into model performance metrics and experiment results.
- **GitHub Actions:** Implement GitHub Actions for continuous integration and deployment (CI/CD), ensuring robustness and automation in the development process.
- **GCP Services :** Utilize GCP monitoring services, such as Stackdriver Monitoring and Logging, to monitor infrastructure health, resource usage, and application performance.
- **Kibana:** For advanced log management and visualization, we will use Kibana. Kibana, integrated with the Elastic Stack, will allow us to visualize logs and metrics, making it easier to detect anomalies and investigate issues. This will include:
    - **Log Management:** Using Elasticsearch, we will store logs generated by various components of the pipeline. Kibana will then be used to create dashboards and set up alerts based on these logs. This will help us monitor the health of the infrastructure and applications, and quickly identify and troubleshoot issues.
    - **Visualization:** Kibana's powerful visualization capabilities will enable us to create detailed dashboards to monitor key performance indicators (KPIs) and other critical metrics in real-time.

11. Success and Acceptance Criteria

Stock price prediction is a complex problem to tackle with present-day methodologies. Our agenda for Success & Acceptance criteria is to find the right balance between achieving decent accuracy and maintaining a robust pipeline. The following points outline the criteria for a successful and acceptable solution:

1. **Achieving Decent Accuracy**
    - **Model Performance Metrics:**

  - **Accuracy:** The model should achieve a minimum accuracy(~60-70%) threshold that outperforms baseline models.
  - **Precision:** The model should have high precision(~70-80%) in predicting stock price movements, minimizing false positives.

2. **Robust and Extendable Pipeline**
   - **Handling Large Training Data:**
     - The pipeline should efficiently manage and process large volumes of training data without overburdening the system.
     - Implement data sharding or batch processing techniques to handle extensive datasets.
   - **Automated Retraining:**
     - Define clear thresholds(~60%) for retraining based on model performance metrics, such as a significant drop in accuracy or changes in input data distribution.
     - Set up automated retraining schedules (e.g., weekly, monthly) to ensure the model adapts to recent market conditions.
   - **Monitoring and Alerting:**
     - **Comprehensive Monitoring:** Use tools like MLflow for continuous monitoring of model metrics and pipeline performance.
     - **Alerting Systems:** Implement automated alerts via Airflow triggers to notify stakeholders of any anomalies, such as data quality issues or significant deviations in model predictions.
3. **Scalability and Extendability:**
   - Design the pipeline to be easily extendable for future enhancements, such as incorporating additional features or switching to more advanced models.
   - Ensure the system can scale to accommodate increasing data volumes and complexity.
4. **Email and Notification Triggers:**

    - Set up thresholds for critical metrics (e.g., accuracy drop by 60%, data pipeline failures) that trigger email notifications to relevant team members.
    - Ensure timely communication of issues to facilitate quick resolution and maintain pipeline integrity.

12. Timeline Planning

Week 1 (May 15 - May 24): Project Scoping

- Finalizing Team members
- Narrowing down project idea & surrounding discussions
- Review idea and with TAs
- Create Document and divide sections among teammates

Week 2 (May 24 - May 31): Data and Model

- Data Cleaning & Preprocessing functions
- LSTM Model building, training and testing

Week 3 (May 31 - June 10): Evaluation and Monitoring

- Model evaluation by Metrics like Precision, Accuracy and other metrics
- Create alerting and monitoring systems

Week 4 (June 10 - June 20): Build Pipelines

- Use airflow to replicate above all modular functions and create a Robust Pipeline
- Use User Profile management to verify all functions
- Deploy the local airflow server in GCP

Week 5 (June 20 - June 29): Review and Presentation

- Document each function of above Pipeline
- Create a presentation for final Review and Present to class, TAs & Professor

13. Additional Information

We plan to create a structure where we can extend the project with less modifications for below points according to time limitations or add as future todo activities

- Additional Financial Data used used to analyze stock prize fluctuations
- Netflix is one company but in Future we can add other Companies
- LSTM is one model but to draw comparison in performance we'll add other simpler model.