Executive Summary – Group 04

Project Overview

This project focuses on developing a **Minimum Viable Product (MVP)** for a stock price prediction and financial analysis application. The system integrates **machine learning (ML)** and **real-time financial data** into an interactive **Streamlit web app**, enabling users to access stock insights and trading recommendations.

The application retrieves **historical and real-time stock data from SimFin**, processes it through a structured **ETL pipeline**, and applies **linear regression models** to predict **the current day's closing price** (CLOSE price t+1 from the latest available CLOSE price from SimFin) for the chosen company out of the five we selected as representative tests for our MVP. While the MVP successfully delivers **real-time stock forecasts**, the system is designed to be **iteratively improved** with enhanced features and more sophisticated models.

A key principle of the design was creating a **modular and decoupled architecture**, allowing independent improvements to **data processing, machine learning models, and the web interface** with minimal disruption.

Approach & Methodology

ETL Process & Web API Integration

- **Historical Data Preparation**: An account is created on SimFin to gain access to their free bulk download tab,where two .csv files (companies.csv and shareprices.csv) containing information about 5.556 companies in USA along with their stock prices information (open, high, low, close, adjusted close and volume) since 1st of April of 2019 until 4th of March of 2024 are downloaded. This data is processed, getting rid of the data believed to be irrelevant for the model training, as well as detecting the variables that worked best to predict the next closing price. Essentially, in this first MVP we only focus on past CLOSE prices, while knowing that a more sophisticated model would require additional features.
- **Real-Time Data Retrieval**: For actual live prediction, data is then extracted from the Web API using the Web API Key provided by SimFin. This is done using OOP, where a function is then called, inputting the company's ticker, the starting date and the end date as arguments. These are the only changeable variables on which the data retrieved depend.

- **Real-Time Data Processing:** The output data obtained from the Web API is then processed so that only the relevant data from the .json file is used for the prediction of the model (the last three closing prices for the company selected by the user). The Web API fetches the last three closing prices for a selected company, ensuring that the model receives real-time inputs to generate the prediction for the next financial day. With the real time data input into the correct, trained model, users are able to predict the next CLOSE price on the specific day they are accessing the app. Additionally, data is processed from the financial statements (Total Revenue and Gross Profit) are retrieved based on the selected fiscal year for an in-depth company analysis.

Machine Learning Model

- After doing the data processing of the historical data from the bulk download, **five different models** were trained, one for each of the companies selected by the team (Apple, Amazon, Google, Microsoft and Tesla), using a Linear Regression Model.
- After training the models with the closing prices for the stocks of each company, this **data is then reviewed and tested** with made up input data to ensure the models are working as expected (knowing that the model is not going to be as precise as we would like, since stock market is unpredictable). We plotted against the test data and calculated error terms incl. Mean Absolute Error (MAE) and Mean Absolute Percentage Error (MAPE) to **evaluate model performance.**
- The models were then **stored as pickle files**, and imported into the script, allowing **real-time inference** within the web application, making sure the user gets the best experience when dealing with stock price predictions. Each input data from the user will now **leverage real-time data** and combine with **predictions from the model trained on historical data,** inputting real-time data only into the corresponding model to ensure accuracy.

Trading Recommendation System

- Users can select a company from **Apple, Amazon, Google, Microsoft, or Tesla** to generate a **predicted closing price** for the current day. The key here is that we predict the **closing price** and **recommend an action for the stockholder** (e.g. buy or sell, because the closing price changes up or down with a simple, percentage-based trading logic).
- Historical and real-time stock data are displayed using **Plotly visualizations** for **trend analysis** and show the user the **training data (blue), new real-time data since then (purple) and his predicted closing price (red star).**
- The **"PREDICT" button** triggers the ML model, displaying the forecasted closing price alongside historical data and real-time data.
- **User-Driven Prediction Mode**: The app allows users to manually input three theoretical closing prices, enabling **custom scenario analysis** and market trend exploration. The catch here is that **the user can prolong the forecast with his own**

predictions (either from our tool or his intuition) - this would allow him to **display his own trend, use multiple forecasts and determine a strategy based on this.**

- **Interactive Forecasting**: By leveraging our **linear regression model**, users can test hypothetical stock movements and see **real-time predictive outcomes**, making it a valuable tool for strategy testing and learning.

Company Information Page

- Users can **retrieve key financial insights** (market data, currency, and employee count) for a selected company.
- The system **fetches revenue and gross profit** using the **SimFin Web API**, ensuring **real-time** financial analysis.
- A **candlestick chart** visualizes stock price movements, while a **comparison tool** enables multi-stock performance evaluation.

Team Page

- Users can **check the trustworthiness of the development team** and base their decision on that as well whether they want to trust predictions
- Our team **consists of multiple, interdisciplinary roles** which ensure that a user does not only trust our technical but also our **business knowledge**

Summary Streamlit Web Application

- **Intuitive Multi-Page Navigation**: The web's structure ensures easy access to a **general overview** with the explanation of the origin and the usage of the web itself, intricate and **real-time company details**, **trading recommendations** for the next day, as well as an **interactive user-friendly tool** allowing the user to try with fictional data to test the models to its limits, and little peek into the team that worked together to make this happen.
- **Real-Time & Historical Data Display**: Data from **CSV files** and **Web API calls** are merged to provide **comprehensive market insights**.
- **Interactive Visuals**: Plotly-powered charts enhance **user experience** and **decision-making clarity**.
- **Deployed on the streamlit Cloud** allowing accessibility for all users globally

Challenges Encountered

1. Making the Trading App a Meaningful MVP

Our current **forecast model** predicts **only today's closing price** based on the last three closing values. While this is a **good starting point**, it does not account for **external market influences, trend shifts, or macroeconomic factors**.

Future enhancements should:

- Use the **predicted close price** as input for **multi-day forecasts**.
- Integrate **additional financial and non-financial indicators** (e.g., **volume, volatility, moving averages, news**) to refine predictions.
- Implement **non-linear models** such as **XGBoost, GARCH** or **LSTMs** for more robust forecasting.
- Expand the **trading strategy to more complexity** e.g. with customizable return thresholds or a portfolio-composition component

2. Ensuring Data Consistency Between CSV and API Data

Historical data (CSV files) had **inconsistent column names** compared to **real-time SimFin API responses**. We implemented **dynamic column mapping** to standardize data structures, ensuring **seamless integration** between bulk and live data.

3. API Rate Limits & Data Retrieval Optimization

SimFin's API has **rate limits**, requiring an **efficient query strategy** to avoid excessive calls. We optimized data retrieval by:

- **Caching API responses** to minimize redundant requests.
- **Validating input dates** to prevent erroneous requests and API failures.

4. User Interface & Experience

- Ensuring **Streamlit's navigation was intuitive** while keeping the UI clean and informative.
- Balancing **detailed financial insights** with a **streamlined user experience**.
- Extended **loading time for the page** resulted in errors infrequently which led us to **(a) delay the display** of the predict button and **(b) add an error message** onto the web interface for users to be aware
- Finding the right **range for the prices in the personal strategy page** to allow extreme movements while **maintaining a meaningful scale logic** on the page

Conclusions & Future Improvements

- The **modular MVP architecture** ensures the project can be **iteratively enhanced** without major system overhauls.
- **Real-time data integration** through the SimFin API enables **dynamic insights and predictions**.
- **Prioritized next steps**:
  - Significantly enhance modelling (with more features, better algorithms like XGBoost and longer time horizon for prediction output)
  - Enhance user experience by switching to a more flexible web development environment (e.g. Flask)

This project provided **hands-on experience** in **financial data processing, machine learning-based stock prediction, API integration, and interactive web development**. The MVP lays the foundation for a more **comprehensive, data-driven trading system** in future iterations.