Learning-Based Financial Price Prediction and Visualization on Cloud Platform

Author: Natali Hovhannisyan
BS in Data Science
AUA

Author: Esfira Babajanyan
BS in Data Science
AUA

Author: Margarita Harutyunyan
BS in Data Science
AUA

Supervisor: Aleksandr Hayrapetyan
YSU
AUA

***Abstract*—Nowadays investments in stock prices are a core pillar of market economy. There are many factors that may affect the industry and; hence, the prices. Those factors may include inflation, market prices, industry trends and most importantly supply and demand. In order to predict stock prices and catch volatility of prices, this project incorporates advanced understanding of Machine Learning(ML) with a user firendly representation of the insights through intergation of Google Cloud Platform(GCP) and the Looker Studio in the project.**

***Index Terms*—GCP, Time Series, Stock Market, RNN, LSTM, AutoEncoder, Looker**

I. Introduction

The stock market is a leading industry for investors, and the dynamic financial market requires price predictions to make accurate data-driven decisions. In recent years, ML algorithms have greatly influenced the field of financial analysis. Financial datasets, such as stock prices, exhibit non-linear patterns;therefore, powerful tools such as deep learning are used to explore the dynamics of price changes in the industry. The aim of this project is to develop machine learning models that will provide critical insights for the users in the stock market and to visualize it in a user-friendly manner. In order to capture volatility of financial data, we have implemented different models such as Long Short-Term Memory(LSTM) with Attention Mechanism, along with Convolutional Neural Networks(CNN), dropout layers and also Autoencoders(AE). In order to have visually appealing representation of all the predictions we have also created visualizations via Looker on GCP.

Cloud computing is a revolutionary technology that enables on-demand provisioning of various computing resources, including servers, storage, databases, software, networking, and many other tools over the internet on a pay-as-you-go basis. It equips individuals and organizations with the necessary tool set to avoid resource overprovisioning and provides higher performance and availability, dynamic scalability, lower expenses, and better security.

GCP is one of the top cloud service providers, offering a wide range of cloud services and a platform to deploy applications and store and analyze data on a scalable and reliable infrastructure. Several of these services aligned perfectly with the needs of our project, such as Google Cloud Functions for serverless computing, Bigquery for data analytics, and Cloud Run for running containerized applications, enabling us to create a customizable and expandable architecture that fits our project requirements. Last but not least, GCP offers generous free trials and credits for new customers, which is perfect for using GCP services without investing money upfront.

Looker Studio is one of the free tools that GCP offers. It is an excellent tool for creating customizable, interactive dashboards and reports that help businesses analyze data more deeply. Its easy connectivity with other Google services and source types expands its usefulness and makes it an adaptable option for data-driven analysis.

II. Literature Review

A. Previous researches in ML field

The fluctuations in the stock market have been a topic of huge interest to many industries, which has led to constant research in the sphere. Various methods have been employed, such as time series models like Arch, Garch (Dana, 2016), reinforcement learning (Lee, 2001), and machine learning models, among others, to understand the volatility of price changes and assist investors in making data-driven decisions. This paper focuses on predicting stock prices using Recurrent Neural Networks(RNN) and AE. Hence, we will delve deeper into a few research studies that have utilized these methods.

In (Shahi et al., 2022) a prediction of chaotic time series was conducted using recurrent neural networks and reservoir computing. In their research, they used LSTM, Gated Recurrent Unit (GRU), Reservoir Computing (RC), Echo State Networks (ESN), and Nonlinear Vector Autoregression (NVAR). For testing the robustness of these models, those were tested on times series from the famous Lorenz 63 and Mackey-Glass which are systems of ordinary differential equations. In addition, other datasets were also used, such as cardiac voltage dataset. To address future computational problems, some of the hyperparameters necessary for the RNNs were predefined but for the most influential ones grid search was used along with the Adam optimizer. For implementing LSTM and GRU

a Depp Learning toolbox in MATLAB was utilized that uses acyclic graph structures. Initially, the network received sequential input, after which LSTM or GRU layers were implemented to capture connections in time series data. In the end, a fully connected layer linked the last gated RNN to the regression output while implementing dropout layers to avoid overfitting. After the training process, multi-step-ahead prediction was utilized through the recursive method. Every output generated from the network served as an input for the next predictions. As an indicator for prediction accuracy, RMSE was used. In conclusion for this paper, the authors derived that although RNNs are good at time series forecasting, they could not perform well with chaotic data. In this specific research, the NVAR and ESN models show better results than LSTM and gated RNN do, yet have lower computational costs.

In (Jahan, 2018), Israt Jahan examined how the LSTM network will work for the prediction of stock prices. The analysis is done on historical data of five years from Yahoo Finance covering 5 major companies: Amazon Inc., FB Inc., Google Inc., Microsoft Inc., and Netflix Inc. After collecting and scaling data with MiniMaxScler the data was used to train LSTM. To asses the model's performance, the Mean Squared Error(MSE) was calculated .In total, 1259 days of data were collected, with the initial 1008 points used as training data for the first run. The model then predicted subsequent data points by incorporating each newly predicted point back into the training set. For example, after predicting the 1009th point the model moved to predicting the 1010th point while including the 1009th point in the training as well. The prediction process continued until the model had price prediction for all days in 2017 with all five models. The study employed a basic LSTM network with the Rectified Linear Unit (ReLU) activation function. Visualization of weekly, monthly, and quarterly averages was used to display the differences between actual and predicted values for all 5 companies. Apart from the visualizations, the author uses statistical metrics to show the model performance. The metrics include the Coefficient of Determination (R**2), Pearson's Correlation Coefficient, Spearman's Rank Correlation Coefficient, and Explained Variance Score. In addition, important statistical tests were also conducted. The tests include the Chi-Square Goodness of Fit Test, Kernel Density Estimation, and Wilcoxon Signed Rank Test. The statistical solutions showed a strong correlation between the predicted and actual values, hence proving that LSTM was a good model for predicting stock prices.

(Selvin et al., 2017) study focuses on predicting the stock prices of companies in the National Stock Exchange list. It is worth mentioning that the data consists of minute-wise information rather than daily prices covering data for 1721 companies for a year. The research has examined the sliding window approach with data overlap, feeding minute-wise data as an input. After obtaining the data, three companies were selected for further examination. The aim of the research was accurate short-term future predictions; namely, a window size of 100 minutes including 90 minutes overlap was chosen for predicting future 10 minutes. Three models that were used in this study are RNN, LSTM, and CNN. Based on their research CNN showed the lowest error percentage and the main reason for this is that CNN does not consider previous lags as is the case for RNN and LSTM. Instead, CNN uses only changes in that current window, minimizing historical data's impact on the final output.

In this research (Xie and Yu, 2021), an unsupervised method is used for the prediction of stock prices, more specifically, Convolutional AutoEncoders (CAE). The structure of the model is as follows: the input is the data covering daily stock prices, exchange rates, oil prices, etc. That is followed by the encoder layer, which consists of three components: Rectified Linear Unit activation function, batch normalization, and a pooling layer. In the middle part, when the features already form a vector, the Support Vector Machine(SVM) model is applied to determine class probabilities. Afterward, the responsibility of the decoder layer is to learn the feature maps, transform those to their original dimension, and pass those to the convolutional layer using the sigmoid activation function. For analyzing the results of CAE, accuracy for next-day predictions is calculated. In the end, the results of the method are compared to four baseline methods such as SVM, Principal Component Analysis(PCA) and SVM, PCA and Deep Neural Networks(DNN), and last but not least LSTM. The paper is considered to be one of the first studies to use CAE for price prediction, and based on this research, it could improve the predictions by 4-7 percent.

III. Methodology

A. ML Components

In our study, we employ three distinct models for predicting stock price, the core of which are LSTM networks and AE. Before diving deeper into the architecture of the models, it is crucial to understand what they represent and how they handle data. LSTM networks belong to RNN, which can take a sequence as their input, such as temporal data. Although RNNs perform well when working with short-term data, they encounter a problem in backpropagation. After doing the backpropagation multiple times, RNNs have either exploding or vanishing gradient problem; to be more precise, when doing backpropagation, we first find the gradient or the derivative for each parameter, and after that, we insert those derivatives into the gradient descent algorithm to find the point that minimizes our loss function. It is worth mentioning that weights and biases remain the same for every layer of RNN. This causes a problem in the backpropagation phase whenever we have a weight that is bigger or smaller than 1 since, in the case of ¿1, our initial input gets multiplied by a considerably huge number (based on how "long" our data is). Because of this problem, we get the exploding gradient problem, which hinders us from finding the optimal parameter. In contrast, when the weight is a number smaller than 1, our initial input gets multiplied by a very small number which will cause a vanishing gradient problem because of keeping the parameter very close to zero and hindering us from finding the optimal parameter because the steps are too slow. LSTM comes in handy to overcome

the exploding/vanishing gradient issue. LSTM is a special type of RNN that was introduced in 1997 by Hochreiter and Schmidhuber.

$$
\begin{aligned}
i_t &= \sigma(W_{ii}x_t + b_{ii} + W_{hi}h_{t-1} + b_{hi}) \\
f_t &= \sigma(W_{if}x_t + b_{if} + W_{hf}h_{t-1} + b_{hf}) \\
g_t &= \tanh(W_{ig}x_t + b_{ig} + W_{hg}h_{t-1} + b_{hg}) \\
o_t &= \sigma(W_{io}x_t + b_{io} + W_{ho}h_{t-1} + b_{ho}) \\
c_t &= f_t \circ c_{t-1} + i_t \circ g_t \\
h_t &= o_t \circ \tanh(c_t)
\end{aligned}
$$

In the case of LSTM, hidden layers of the network are replaced with LSTM layers, which help to differentiate between events that happened long ago (long memories) and ones that happened recently (short-term). LSTM utilizes the sigmoid function, which maps the input to a range between 0 and 1, and the tanh function, which maps the input to a range from -1 to 1 (See Figure 1). Apart from the functions included, the architecture of LSTM can be decomposed as the following:

- Input gate
- Cell state
- Hidden state
- Forget gate
- Output gate

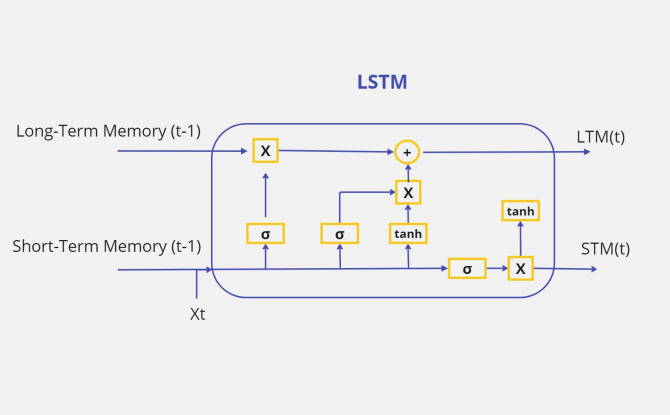

Fig. 1. Long Short-Term Memory

The cell state of LSTM is responsible for long-term memory and does not include any biases and weights in order to avoid the infamous exploding or vanishing gradient problem. The hidden state of LSTM is responsible for short-term memories and, hence, also includes weights that can modify the initial input. LSTM networks also include 3 gates, which are crucial parts of the network's functionality. Firstly, in the forget gate, the sigmoid function decides if the long-term memory will be kept or not, since if it maps the value with 0 and multiplies it with the long-term memory, it will, obviously, vanish. In the input gate, the existing long-term memory is already being updated using the tanh and, afterward, the sigmoid activation functions. Last but not least, we use the output gate to decide the final short-term memory. Since the LSTM network is designed to handle the exploding/vanishing gradient problem when accepting sequence as an input we have decided to incorporate it in two of our models.

AugmentedLSTM: In our initial model, we have implemented LSTM complemented with dropout, ReLu, and linear transformation. After going through the LSTM layers, the data goes through the dropout layer. The functionality of the dropout layer is as follows: it randomly nullifies some of the input values, which helps avoid overfitting the data and optimizes computational efficiency. After passing through the dropout layer and deactivating some of the nodes, the data undergoes the linear transformation layer upon which the ReLu activation function is implemented. In the linear layer, the data is reduced in dimensionality twice, and after going through linear transformation, the data is immediately introduced to the ReLU function.

$$ReLu(x) = max(0, x) \tag{1}$$

The function retains only non-negative values by comparing the input to zero and proceeding with the maximum. From that, we can infer that ReLu excludes negative values and introduces the model to non-linearity, which is a crucial step in detecting more complex patterns. In the final layer, the model undergoes another linear transformation, but this time, the dimensionality is reduced to 1, and the output is already the prediction (See Figure 2).

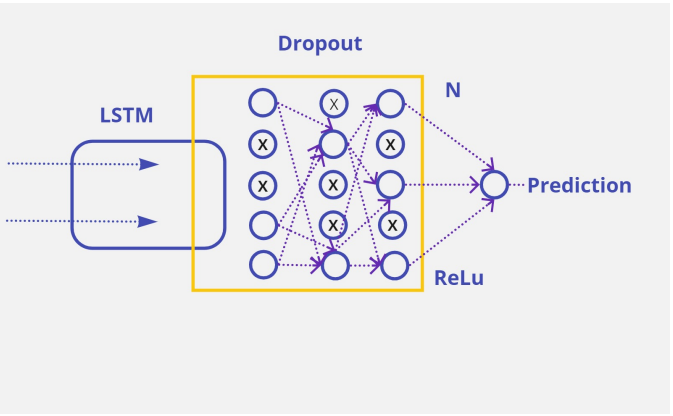

Fig. 2. Augmented LSTM

ComplexModel: Our second model is a hybrid integrating LSTM with CNN (Sayavong et al., 2019) and Attention Mechanism. Although the model also includes ReLu activation and dropout layer, CNN and multi-head attention have a pivotal role in enhancing the accuracy of the predictions. To elaborate, let us decompose the functionalities of each. CNN is usually well known for image processing and recognition due to its ability to find necessary shapes or patterns in an image (Chen and He, 2018). However, stock price prediction

can also be a field for CNN to find patterns. Mainly, CNN finds features with higher importance and increases their weights to amplify their influence on the prediction. CNN processes the stock data as a matrix and applies convolutional filters over all the blocks in the matrix (See Figure 3). The results of the conducted operation are saved in a new matrix in the form of a dot product between the convolutional filter and a block. CNN will particularly accentuate the importance of the ADJ Close feature since its impact on the predicted output is bigger.

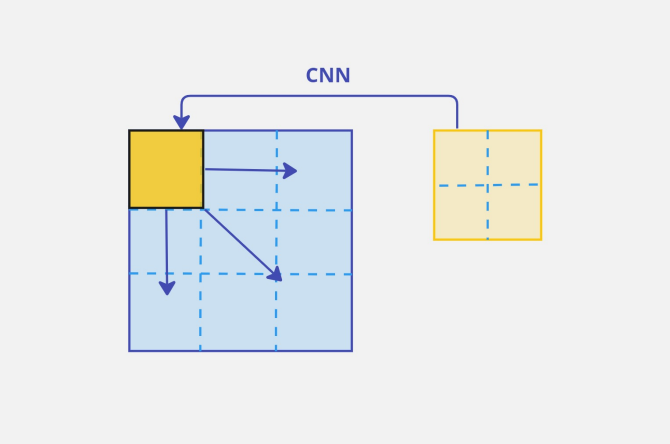

Fig. 3. Convolution Filter

The next important technique in this model is the attention mechanism (Vaswani et al., 2017), in our case specifically the multi-head attention mechanism. The attention mechanism mimics the human brain's neurological system which blurs some background information when a certain part of our brain needs more attention. The attention mechanism complements the availability of CNN by highlighting the important features once again. It selectively focuses on some parts of the input when making the prediction and in our case, it should be the Adj Close since it impacts the close price the most. The attention mechanism is beneficial when working with long data, such as time series, since due to its technique of constantly highlighting useful parts it helps to extract necessary information from extensive data without losing those. The self-attention mechanism works by having three important components for each input vector: query, key, and value. The query and key pass through layers before reaching the matrix multiplication with the value vector. The first query and key undergo matrix multiplication which results in an attention score. The attention score is the dot product of the query vector with a key value and shows how well the values align with each other. Since these initial attention scores can be huge numbers, the next layer is scaling so as to have more consistent values. Lastly, the values are passed to the softmax function which maps their values to a number in the range of 0 and one. The probabilities achieved after softmax determine the weight of each value vector. (See Figure 4)

The multi-head attention does multiple self-attentions in parallel; hence, covering a bigger space of information and being more time-efficient. The difference is that there is not only one vector of query, key, and value but sets of queries, keys, and values that simultaneously focus on different sections of the input data. (See Figure 5)

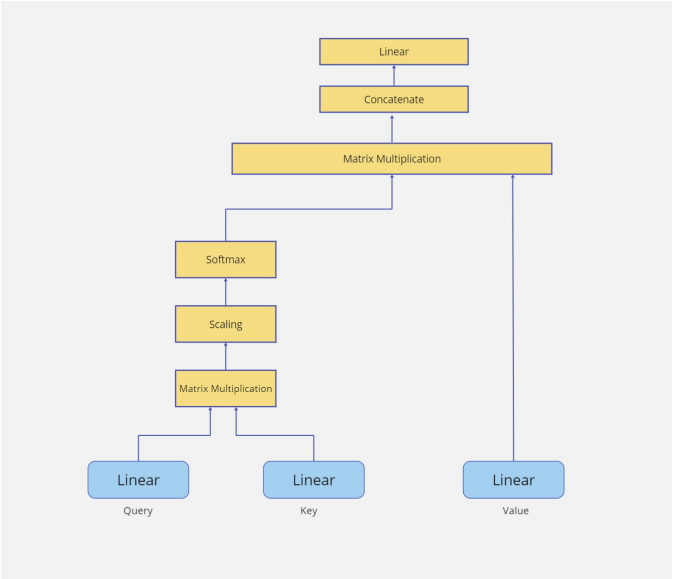

Fig. 4. Self-Attention Mechanism

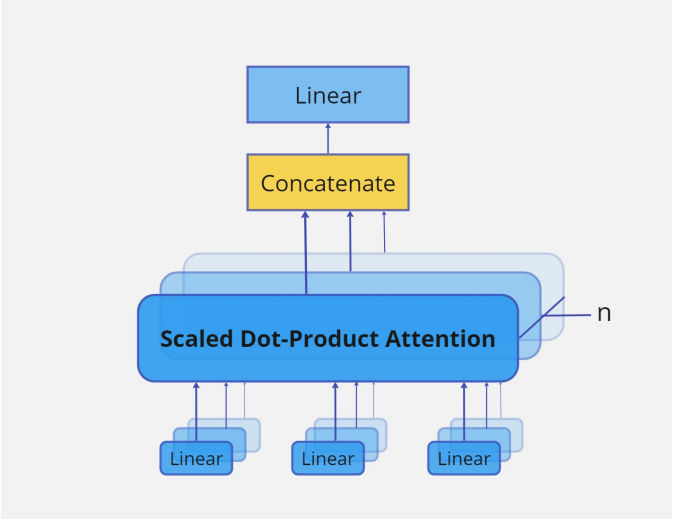

Fig. 5. Multi-Head Attention

AutoEncoder: In the development of our last model, we have integrated AE. AE are used to encode and reconstruct the input data as their output. AE are mainly used for dimensionality reduction or for feature learning (Xie and Yu, 2021). They are composed of four key components: input, encoder, decoder, and output. Upon receiving the input, AE moves it to the encoder chunk which may consist of several layers. In

the encoder section, the input data undergoes dimensionality reduction, which then is given to the decoder part as input. The decoder, in contrast, tries to bring back dimensionality and mimic the input data. The difference between the output and input is then calculated with a loss function; so ideally, we should have identical input and output. In our proposed version of AE for price prediction, we have employed linear transformation and the ReLu activation function several times. First, when the input sequence enters the encoder section it undergoes a linear transformation which maps the input dimension to a hidden layer dimension. Exactly afterward we use the ReLu activation function to integrate non-linearity. At this point, the data is transferred to the decoder section as input. The decoder tries to reconstruct the data by first transferring the data back from the encoder layer to the hidden layer and implementing ReLu above. Then again one more linear transformation already maps data from the hidden layer to its original dimension (See Figure 6). The last timestamp of the prediction is treated as the close price.

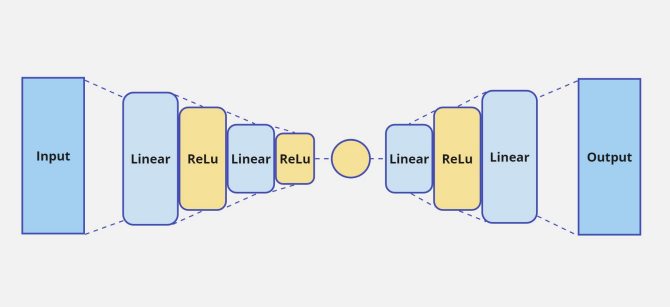

Fig. 6. AutoEncoder

B. Platform Engineering

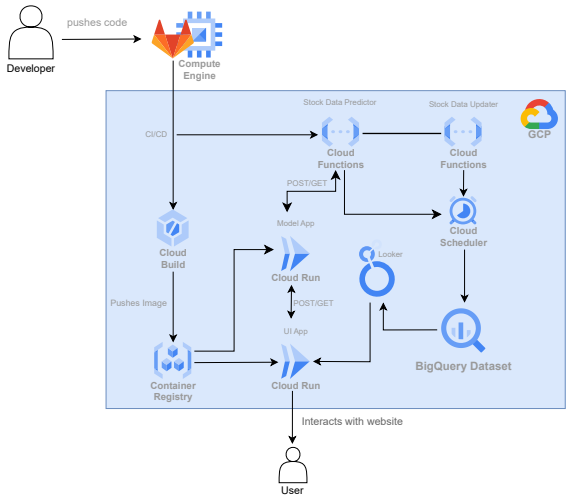

Fig. 7. Cloud Platform Architecture

1) Cloud Architecture and Deployment:

Google Compute Engine: Google Compute Engine is an Infrastructure-as-a-Service (IaaS) product provisioned by Google Cloud Platform for hosting self-managed virtual machine (VM) instances, which provide computing and hosting capabilities (Cloud, 2024). We have utilized this service to deploy a self-hosted GitLab instance on a custom Machine type with x86/64 Architecture and Ubuntu 23.10 Operating System (OS). The GitLab instance provides the central repository and automation hub for our project's source code versioning, deployment scripts, and pipelines.

Continuous Integration and Deployment (CI/CD): The project utilizes a GitLab CI/CD pipeline for automated testing and resource deployment to follow Software Development best practices to avoid bugs and failures, decrease complexity, and increase efficiency. The pipeline automates the testing, building, and deployments to GCP. GitLab runners, lightweight agents responsible for running CI/CD pipeline jobs, are also hosted on the same Compute Engine instance for reduced network latency, simplified management, and cost efficiency. The runners use a Shell executor, the simplest executor provided by GitLab. It allows it to run the jobs directly on the host machine without dealing with the overhead of running virtualized workloads and managing containerization.

2) Data Management and Processing:

BigQuery Datasets: BigQuery is the central data warehouse for storing historical stock data and prediction values in two separate Datasets, which are used for the subsequent analysis and visualization. BigQuery is Google Cloud's fully managed, petabyte-scale data warehouse chosen for its cost-effectiveness (Cloud, 2024).

Cloud Functions: The cloud function is a serverless computing solution that enables the developer to run lightweight, single-purpose functions that respond to cloud events without managing the runtime environment or a server. The project cloud functions are used for automatic data retrieval and processing and act as a scheduled task or "CronJob." The stock data updater function retrieves data using 'yfinance' Python library, which in turn sources the data from Yahoo Finance. The second function processes the same data to generate predictions by sending a request to the application that runs our model predictions and forwards the prediction to a specified application endpoint (See Figure 7). These functions are scheduled via Cloud Scheduler, which indirectly triggers the execution by publishing a message to the corresponding function's Pub/Sub trigger topic. After data retrieval, the functions format the data to the required format and upload it to the respective BigQuery Datasets (Figure 7).

3) Application Layer:

Cloud Run Applications and Artifact Registry: There are two applications, a UI application and a Model application running on Cloud Run, a managed computing platform that enables containerized applications to be run that can respond to web requests or events. The CI/CD pipeline triggers a build on the cloud, which creates the Docker images for both applications and stores them in the Google Artifact Registry.

Cloud Run pulls these images from the Registry to run the applications while ensuring both run the latest versions with consistent environments. UI Application serves as the user interface and is the immediate gateway to our platform for our stakeholders. It incorporates embedded Looker Studio reports, which utilize the BigQuery mentioned above datasets as their data source. It also provides the functionality for making prediction requests with appropriate inputs for Open, High, Low, Adj Close, and Volume values to the Model application. The Model application manages the computation of stock price predictions. It receives input from the UI, makes the prediction based on the trained and saved machine learning models, and sends the computed results back to the UI for display (Figure 8). This application also receives requests from the platform's Stock Data Predictor Cloud Function.

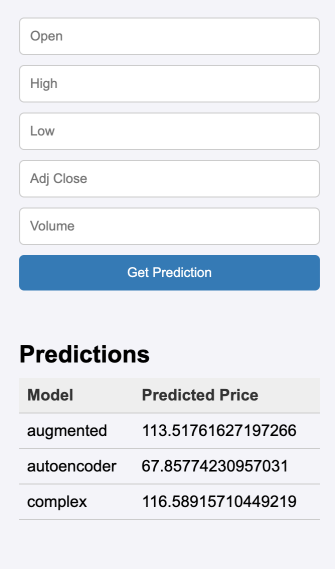

Fig. 8. Prediction results on the UI

4) Integration and Workflow: The system's structure is planned to guarantee a smooth flow of data and connectivity among different parts. Information retrieved and handled by Cloud Functions is saved in BigQuery, utilized by Looker Studio for visualization, and employed by the model app for prediction generation. The UI application shows these forecasts, giving stakeholders useful, live analytics.

5) Security and Access Management:

Service Account Configuration: To maintain a strong level of security and follow the least privilege principle, separate service accounts have been created for different infrastructure components. Every service account is customized to have only the essential permissions and roles needed for its responsibilities. This reduces possible security risks by limiting each account's access to only necessary resources for its tasks.

Key Management and Pipeline Security: To ensure protection from unauthorized access, access keys for service accounts were securely controlled. Before running the CI/CD pipeline, these keys are encrypted and saved as environmental variables in the CI/CD system. Keys are deciphered on the fly only when necessary during the pipeline execution, guaranteeing that they are not revealed in logs or hard coded in repositories.

Identity and Access Management (IAM): The project enforces strict IAM policies consistently reviewed and adjusted to adhere to top security standards. These measures guarantee that only approved individuals, and automated systems can carry out particular tasks in the cloud setting. Role-based access control (RBAC) is strictly implemented to control resource access. Assigned specific roles to users and service accounts to determine access level and allowed operations, ensuring a secure operational environment.

VISUALIZATIONS WITH LOOKER

In the scope of this capstone, the primary tool used for Visualizing the data and the KPIs was Looker Studio, offered by Google. Looker Studio is a self-service business intelligence that was the most suitable for our project, and it is hosted on the cloud platform. It has a lot of functionality and allows the user to analyze data and make visualizations. One of its most significant advantages is its user-friendly interface, which is intuitive for newcomers. Looker Studio has various connectivity options, including BigQuery, which we used in the capstone project. This connection allows us always to have up-to-date data that will appear on our dashboard as soon as the BIgQuery tables are filled. Moreover, Looker Studio has a wide range of visuals, such as line charts, bar plots, and heat maps. It also allows the creation of metrics that will be used to calculate some KPIs (Key Performance Indicators) to analyze the data more deeply. We made six separate dashboards for our project, five representing the stock data analysis for five companies. These dashboards have the same appearance but analyze data for different companies. The final dashboard contains combined data for all five companies and allows users to compare the stock data of those companies. The individual dashboards contain time series data for the specific company's close price (Figure 9).

Fig. 9. Close Price over time

Our dashboard also includes a date filter that allows us to observe data in a particular date range to avoid complex and unreadable plots. The filter applies to the whole page, which

helps to examine other measures and KPIs we define within that range. For instance, our data contains a calculated column ROI (Return on investment), which shows the profitability of the investment and is used to calculate and depict the number of negative and positive ROIs on the dashboard, which changes when the date range is altered (Figure 10).

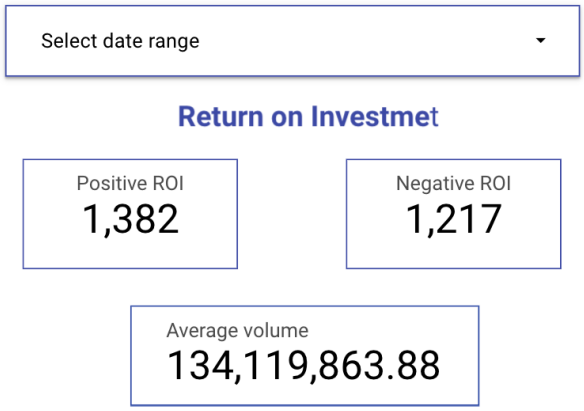

Fig. 10. Date Filter Effect on ROI count

The Main dashboard containing combined data also includes a dropdown menu for Company selection, allowing users to observe data analysis for the companies they need (Figure 11). As in the case of the date filter, the dropdown menu also applies to the whole page and allows one to observe the plotted graphs with the specific companies. For instance, the main dashboard contains the ROI value over time for the five companies, but we can select only Apple and Meta (Figure 11).

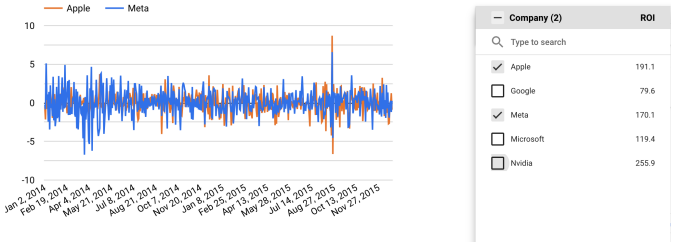

Fig. 11. Drop down menu application

IV. Results

The analysis of the training processes of our three models has given us clear understanding of the performance of each model. Based on training results the best model from our project is the AE model. However; the Augmented LSTM also gives very close results. The model with the highest fluctuations remains the hybrid model which includes LSTM along with CNN and Attention Mechanism. Below are the visualizations corresponding to the training process on Google stock data.

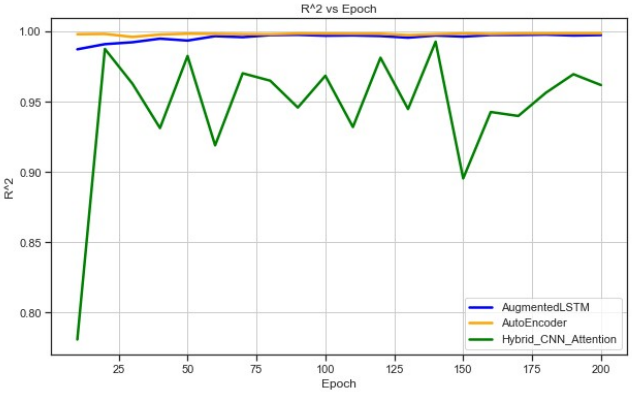

Fig. 12. R2 for Google

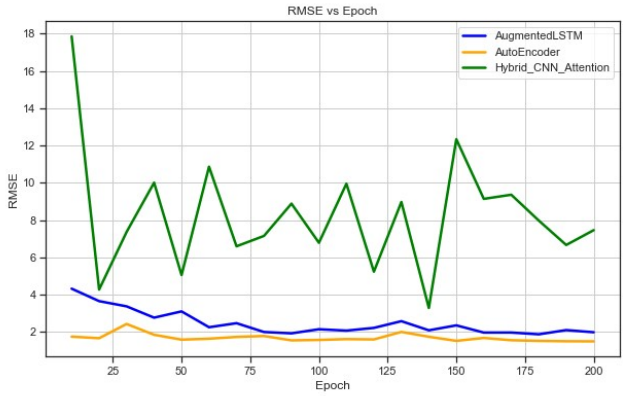

Fig. 13. RMSE for Google

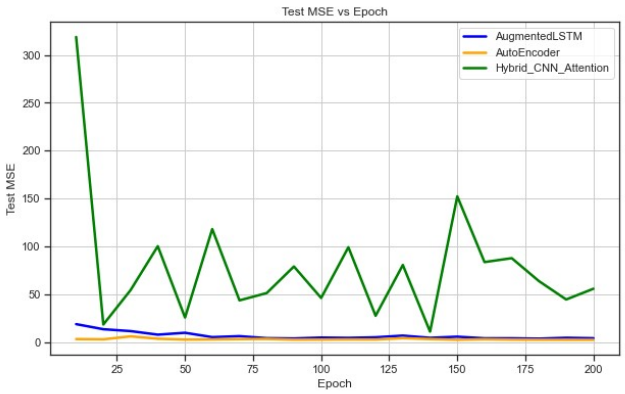

Fig. 14. MSE for Google

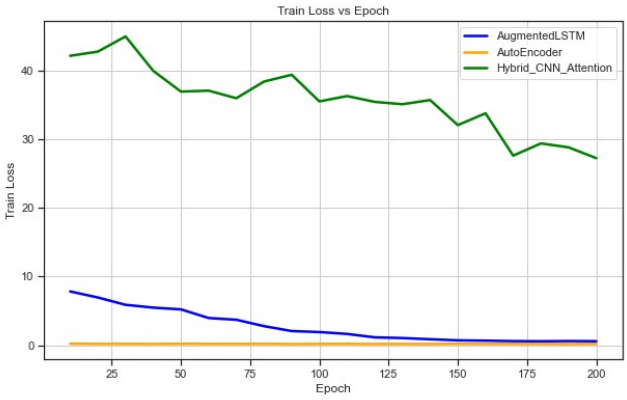

Fig. 15. TrainLoss for Google

V. Conclusion

To conclude, our project has effectively utilized the GCP to implement a multi-layered, scalable, and safe system for stock data management and prediction. Employing Google Compute Engine and other GCP services like BigQuery, Cloud Functions, and Cloud Run, we have created a system architecture that executes all data processing automation and sustains continuous integration and deployment. Moreover, we have successfully incorporated advanced machine learning models, including RNN such as LSTM networks, CNN and AE, to emphasizing important features and make predictions about the closing prices of stocks.This architecture satisfies the project's technical and operational requirements and shows the benefits of using cloud technologies, which ensure better performance, reduce costs, and make data processing and analysis more efficient. This platform now constitutes a strong basis for future innovation; it can serve as an upgradable structure that new project specifications or technological advancements can flexibly modify.

Bibliography

Sheng Chen and Hongxiang He. Stock prediction using convolutional neural network. In *IOP Conference series: materials science and engineering*, volume 435, page 012026. IOP Publishing, 2018.

Google Cloud. Google cloud platform documentation, 2024. URL https://cloud.google.com/docs.

AN Dana. Modelling and estimation of volatility using arch/garch models in jordan's stock market. *Asian Journal of Finance & Accounting*, 8(1):152–167, 2016.

Israt Jahan. Stock price prediction using recurrent neural networks. 2018.

Jae Won Lee. Stock price prediction using reinforcement learning. In *ISIE 2001. 2001 IEEE International Symposium on Industrial Electronics Proceedings (Cat. No. 01TH8570)*, volume 1, pages 690–695. IEEE, 2001.

Lounnapha Sayavong, Zhongdong Wu, and Sookasame Chalita. Research on stock price prediction method based on convolutional neural network. In *2019 international conference on virtual reality and intelligent systems (ICVRIS)*, pages 173–176. IEEE, 2019.

Sreelekshmy Selvin, R Vinayakumar, EA Gopalakrishnan, Vijay Krishna Menon, and KP Soman. Stock price prediction using lstm, rnn and cnn-sliding window model. In *2017 international conference on advances in computing, communications and informatics (icacci)*, pages 1643–1647. IEEE, 2017.

Shahrokh Shahi, Flavio H Fenton, and Elizabeth M Cherry. Prediction of chaotic time series using recurrent neural networks and reservoir computing techniques: A comparative study. *Machine learning with applications*, 8:100300, 2022.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Li Xie and Sheng Yu. Unsupervised feature extraction with convolutional autoencoder with application to daily stock market prediction. *Concurrency and Computation: Practice and Experience*, 33(16):e6282, 2021.