Final Project Report (Group No: 10)

Sanjana Vasireddy, Shyam Krishnan Ondanat Veetil, Shayan Javid

Introduction

In this project our goal is to design a comprehensive tool for stock data analysis and prediction. In our project, we have implemented 5 major components. We have the user interface (UI), data analysis/machine learning, data storage, earnings call transcript analysis, data streaming with Kafka, and machine learning with Spark. For our UI we are using streamlit to implement a simple user interface that is very easy to follow for a non-technical user. In our interface we have given the option of uploading a stock trading csv file for a company or the user can choose one of the pre-loaded companies and view their analysis without having to upload anything. Our data, user's data, and the result of machine learning models are stored in the firestore. For the machine learning aspect of our project we are utilizing Facebook's Prophet procedure which enables us to predict the closing, low, and high price of stocks in the future. In addition, we are utilizing spark and Spark Mlib for three different regression types, which gives additional power to the user on stock price prediction. We also support live data streaming for the top S&P 500 stocks, and earnings call transcript analysis.

Main Components and Their Implementation

Data Uploading, Data Visualization, and Machine Learning

The main page of our platform allows users to either upload their own stock data, or to choose from some of the pre-loaded stock data provided by us. Users can upload stock data in csv format for further analysis. Our platform requires that the user's data has the six main columns (date, open, high, low, volume, and close) for a comprehensive and detailed analysis. After uploading, we provide some basic information on the uploaded data. Number of data points, number of features, the first five rows of their data, and finally for each column we provide the mean, max, standard deviation, 25%, 50%, and 75% percentiles.

Shown are the stock price data for query companies!

Apple Inc.

Apple Inc. designs, manufactures, and markets smartphones, personal computers, tablets, wearables, and accessories worldwide. It also sells various related services. In addition, the company offers iPhone, a line of smartphones; Mac, a line of personal computers; iPad, a line of multi-purpose tablets; AirPods Max, an over-ear wireless headphone; and wearables, home, and accessories comprising AirPods, Apple TV, Apple Watch, Beats products, HomePod, and iPod touch. Further, it provides AppleCare support services; cloud services store services; and operates various platforms, including the App Store that allow customers to discover and download applications and digital content, such as books, music, video, games, and podcasts. Additionally, the company offers various services, such as Apple Arcade, a game subscription service; Apple Music, which offers users a curated listening experience with on-demand radio stations; Apple News+, a subscription news and magazine service; Apple TV+, which offers exclusive original content; Apple Card, a co-branded credit card; and Apple Pay, a cashless payment service, as well as licenses its intellectual property. The company serves consumers, and small and mid-sized businesses; and the education, enterprise, and government markets. It distributes third-party applications for its products through the App Store. The company also sells its products through its retail and online stores, and direct sales force; and third-party cellular network carriers, wholesalers, retailers, and resellers. Apple Inc. was incorporated in 1977 and is headquartered in Cupertino, California.

AAPL data

Number of Rows

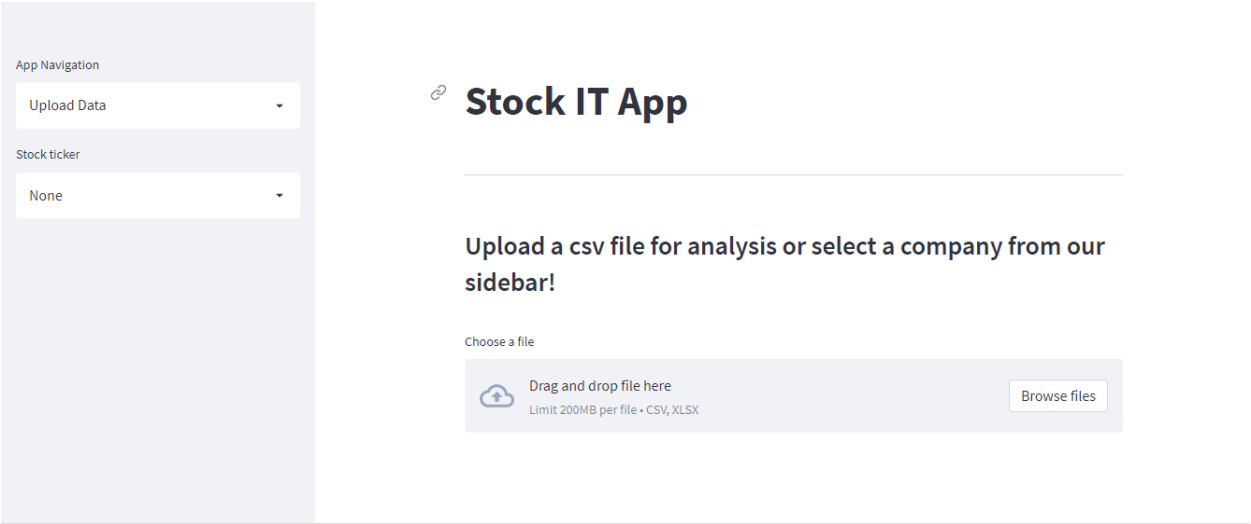
346

Number of Features

6

	date	volume	low	high	open	close
0	2015-07-10	61354500	117.5345	120.0944	118.2423	119.5417
1	2015-07-13	41440500	120.5502	121.9465	121.2386	121.8495
2	2015-07-14	31768100	121.2483	122.5380	122.2180	121.8011
3	2015-07-15	33649200	121.7720	123.2944	121.9077	122.9744
4	2015-07-16	36222400	123.4883	124.6713	123.8665	124.6131

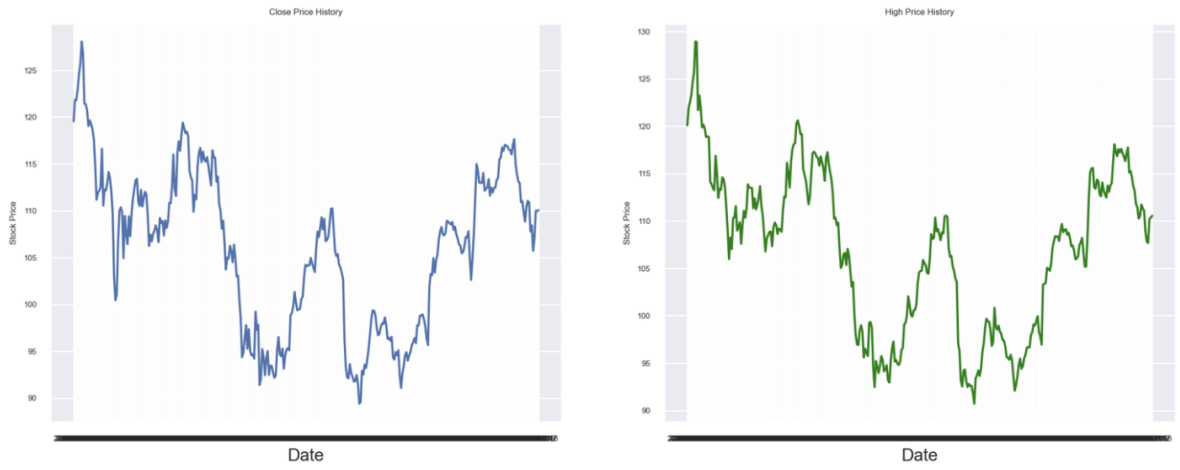
Fig[1]: Some General Information About the Company



Fig[1.b]: Facility to upload Stock Data in CSV and XLSX format

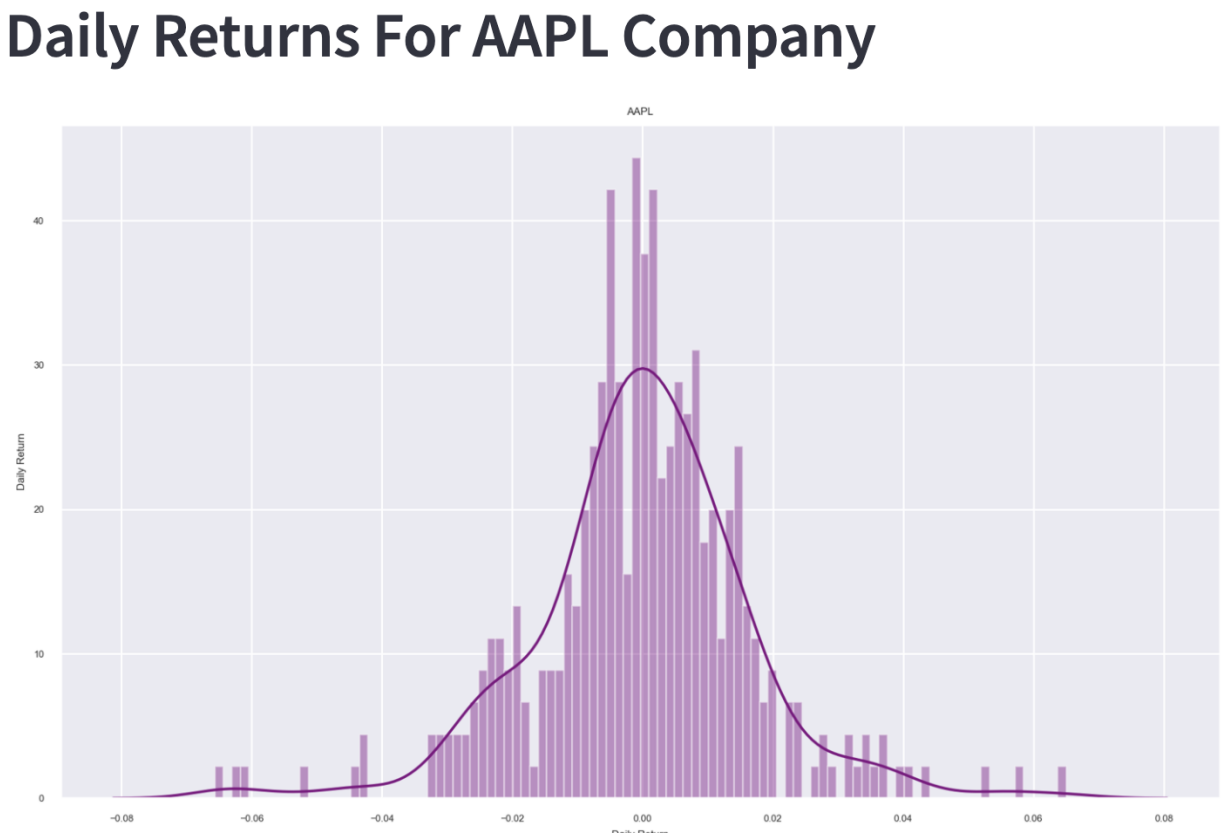
Description for AAPL

	volume	low	high	open	close
count	346.0000	346.0000	346.0000	346.0000	346.0000
mean	44,124,675.4335	104.8623	106.8771	105.8825	105.9070
std	20,659,622.1133	8.4022	8.4095	8.4361	8.4148
min	13,046,400.0000	88.5334	90.7104	89.0578	89.3943
25%	30,153,400.0000	97.1324	98.8542	97.7874	97.8364
50%	38,137,300.0000	106.3653	108.2314	107.2543	107.2596
75%	52,097,925.0000	111.2118	113.4536	112.6902	112.2440
max	162,206,300.0000	126.7367	128.9379	128.8215	128.0652



Fig[2]: Data Description and Graphs

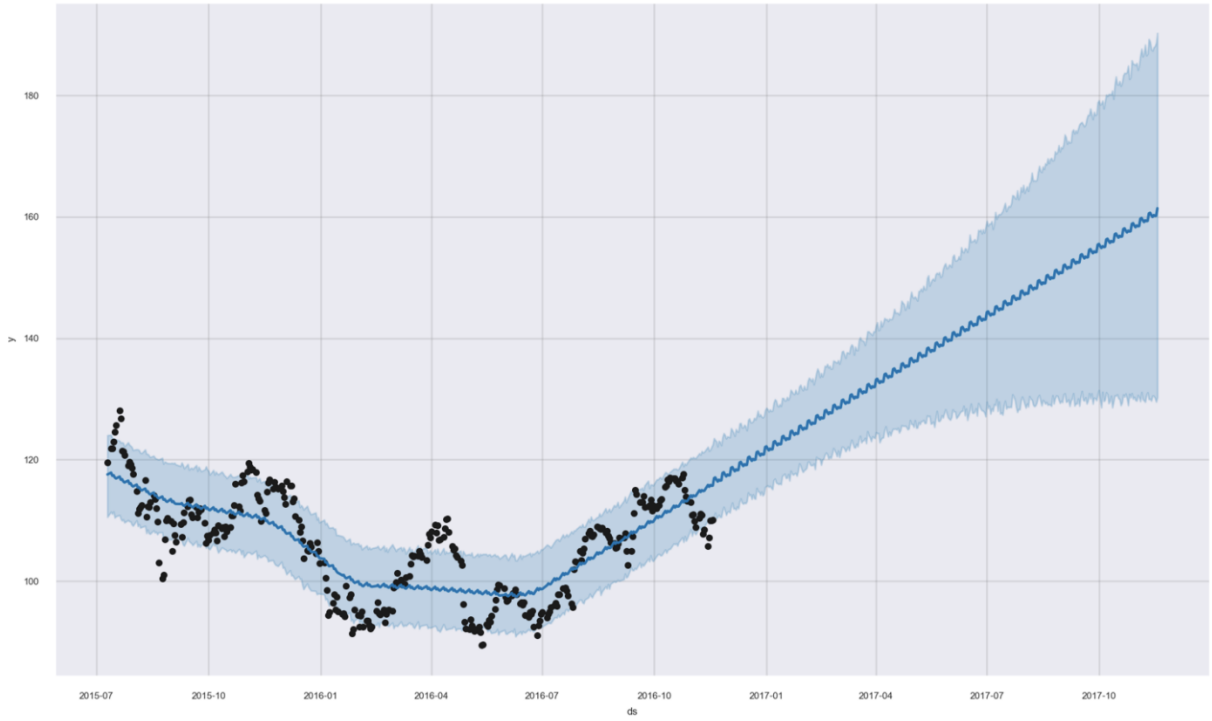
Furthermore, we provide some graphs to users for more insight on the uploaded data. Closing price history, and high price history are two graphs that we show to the user in order to better visualize the stocks performance in the past. Next we are plotting the daily returns for the company. Daily return is defined as the difference between the opening price and the closing price of the company for a specific day. In this graph we are calculating and showing the number of days that the company had positive, negative, or zero returns.



Fig[3]: Daily Returns

For the machine learning aspect of our platform, we are using Facebook Prophet for time series prediction of the stock data. Based on users input, we are providing predictions for 365 days after the last data point. Obviously the more data points the user provides, the more accurate the results of the prediction will be. After the prediction, we are graphing the prediction with the error margins for the user.

Machine Learning



Fig[4]: Stock Data Prediction with Prophet

The user is also able to pick a specific date within the range of predictions to view the exact predicted closing, low, and high prices for that day. Alternatively the user is able to view a wide range of dates from the end of prediction.

Pick a date to view prediction:

2016/11/29

	ds	Predicted Closing Price	Predicted Low Price	Predicted High Price
356	2016-11-29	117.2367	110.8833	123.4752

Select the number of rows for prediction

10 − +

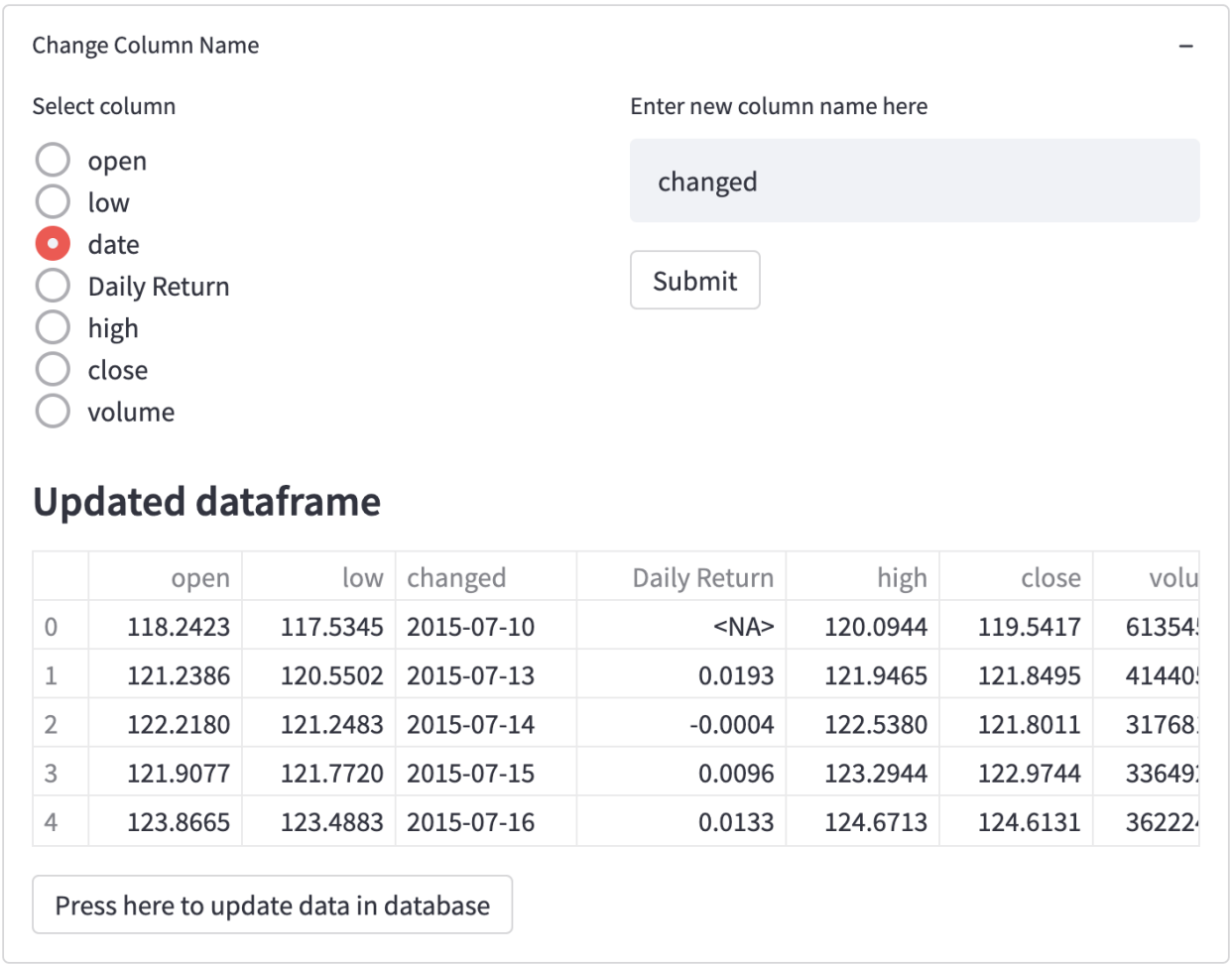
	ds	Predicted Closing Price	Predicted Low Price	Predicted High Price
701	2017-11-09	159.3014	130.6154	186.5897
702	2017-11-10	159.3420	131.0637	186.8623
703	2017-11-11	160.5544	129.9408	189.2937
704	2017-11-12	160.6766	130.0079	188.8595
705	2017-11-13	160.2987	129.6941	187.6216
706	2017-11-14	159.9978	130.5397	187.3704
707	2017-11-15	160.2977	130.4841	187.9492
708	2017-11-16	160.1566	130.1332	188.1164
709	2017-11-17	160.1972	129.5256	188.6735
710	2017-11-18	161.4097	130.2926	190.2841

Fig[5]: Prediction Results

All the above mentioned details are applicable if the user chooses one of the preloaded datasets, that we are storing in Google Firestore, that we are providing.

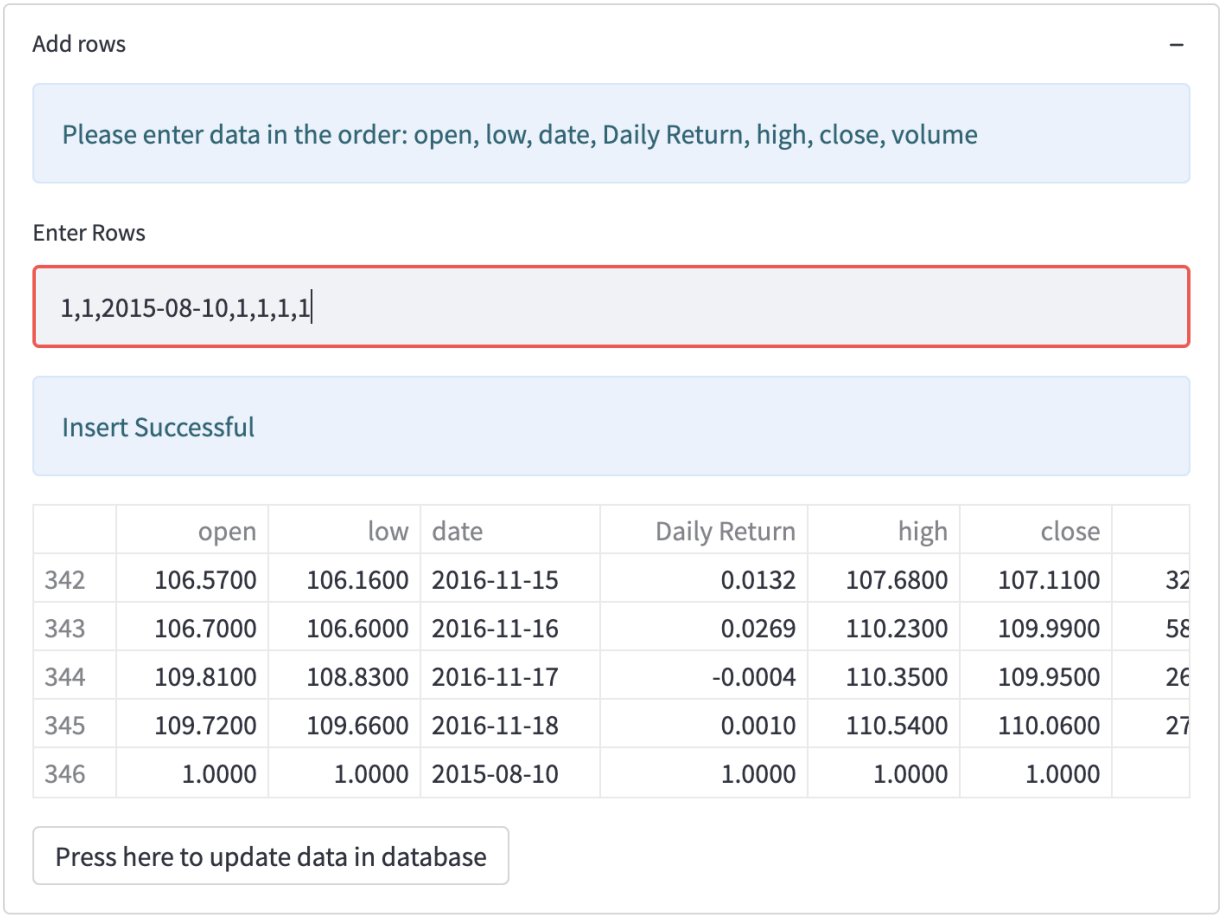
Data and Metadata Updating

Users are provided with the opportunity to update their data by amending rows to existing records and also have the opportunity to update the column names.
For column name update users are provided with radio buttons to choose from the available columns and then enter a name for the selected record. Upon clicking the submit button the updated records are shown which then can be loaded into firebase. Fig. No:



Fig[6]: Updating a Column

Similarly users can add records to the data using a similar feature as shown above. Here the user can add rows to the already existing data. ALl they need to do is to copy paste the records as a comma separated list on the provided text box. This data can be updated to firebase using submit button



	open	low	date	Daily Return	high	close	
342	106.5700	106.1600	2016-11-15	0.0132	107.6800	107.1100	32
343	106.7000	106.6000	2016-11-16	0.0269	110.2300	109.9900	58
344	109.8100	108.8300	2016-11-17	-0.0004	110.3500	109.9500	26
345	109.7200	109.6600	2016-11-18	0.0010	110.5400	110.0600	27
346	1.0000	1.0000	2015-08-10	1.0000	1.0000	1.0000	

Fig[7]: Adding a Row to the Database

Earnings Call Analysis (Non-Text Data)

One other feature that our platform offers is earnings call analysis. The users are able to upload a pdf format of the earnings call transcript of their company of interest for analysis. The users are also allowed to upload their pdf file for later use, for which we are utilizing Amazon S3 data storage. After the user has uploaded their file, our platform provides some basic metadata information on the file such as, length of the transcript, number of pages, and number of sentences in the transcript document. Furthermore we are provided sentiment analysis on the transcript which can help the user make a decision on whether he/she should invest in the company of their interest. For the sentiment analysis we are providing a polarity score of the whole document. Polarity score is a score between -1 and 1. The closer the score is to 1, the more positive is the tone of the document. Next we are also calculating the subjectivity score for the document. The subjectivity score is a score between 0 and 1. If the score is too close to 1, then that indicates that the writer of the document is trying to persuade the readers to agree with his/her opinion. Similarly if the score is close to 0, then the document is written more neutrally. Since the overall analysis of the polarity score may not be very detailed, we are calculating the

polarity score for each individual sentence and categorizing them to positive, negative, and neutral.

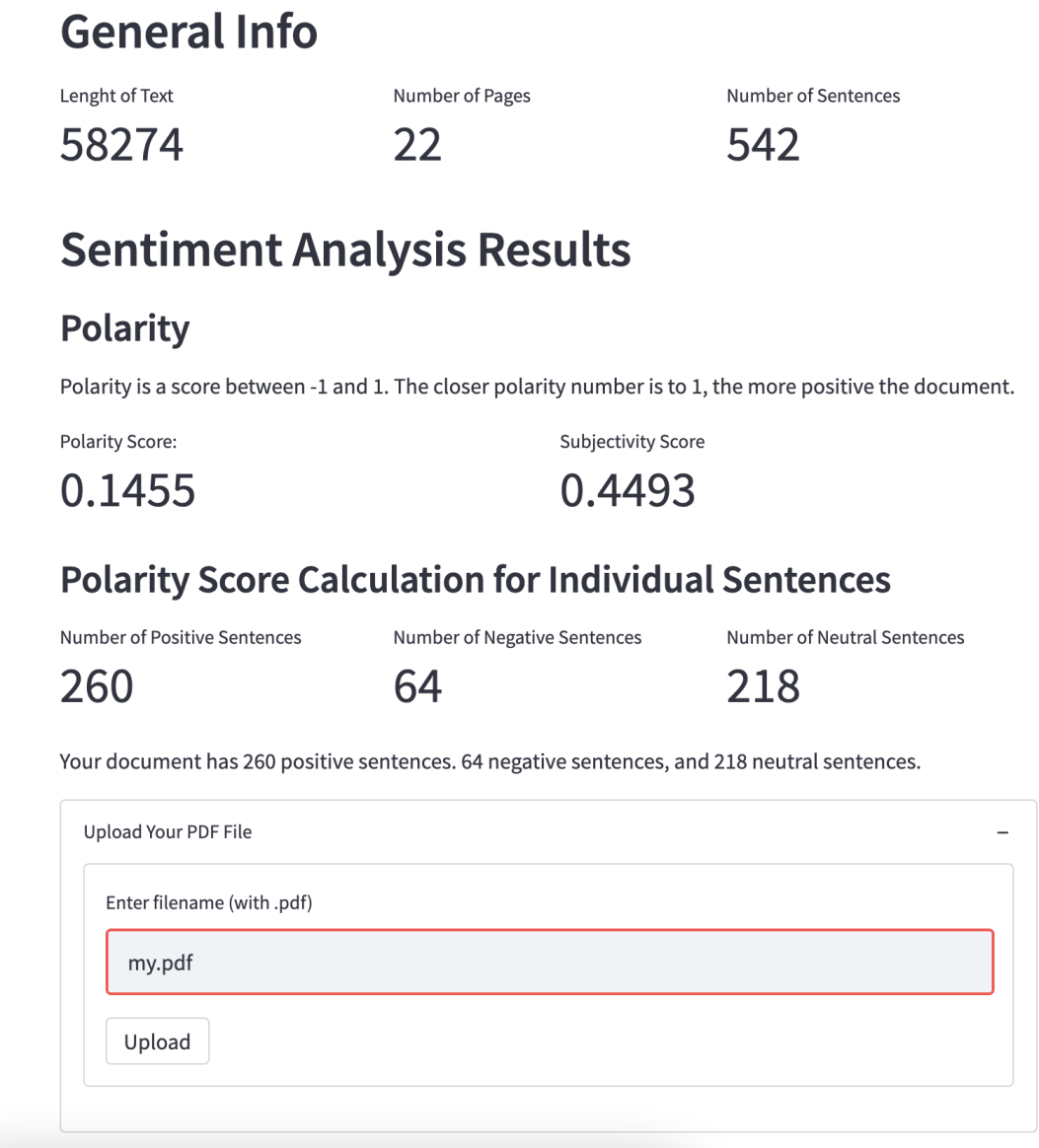
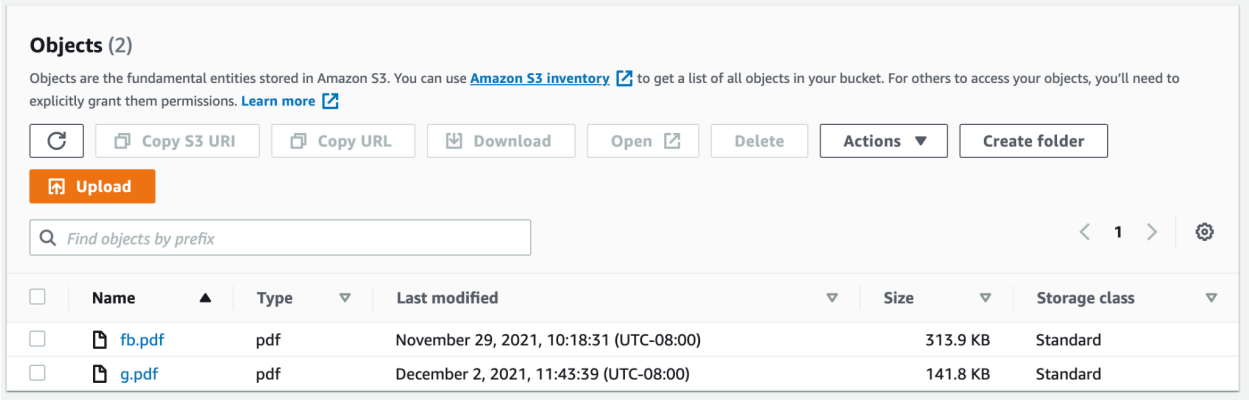


Fig [8]: Earnings Call Analysis



Fig[9]: Amazon S3 Storage After Uploading pdf Transcripts

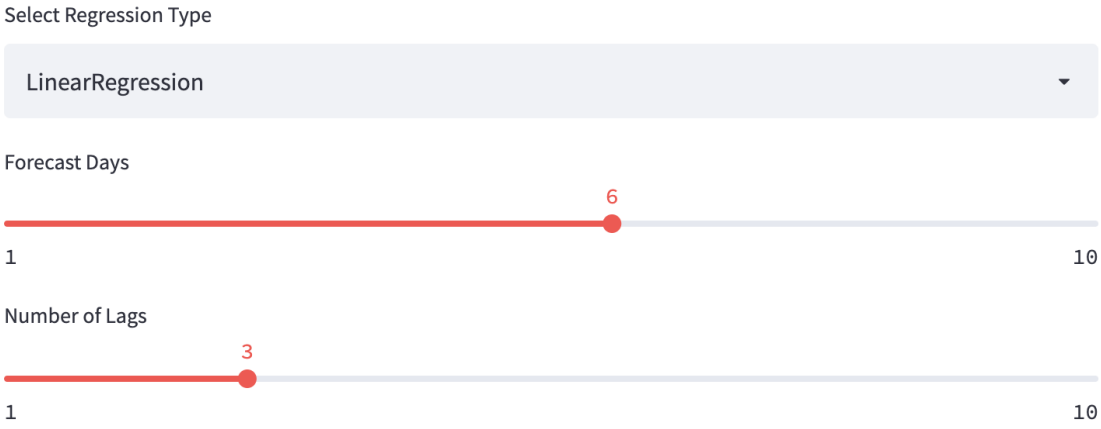
Machine Learning and Feature Extraction with Spark

This part of the website is designed to help the user to see the forecast for the selected stock data. Users have multiple options to choose the algorithms and also adjust the predictions using the lag and forecast days. Slider in the page allows the user to make these selections.
Once the selections are made, the algorithm automatically runs in the background and shows the results to the user. We use RMSE as an evaluation metric.
If the user wants to make predictions with fine tuning the parameters then they just need to adjust slider positions. Internally the code splits the data into training and test sets with the split ratio as 0.7.

Spark Data Analysis for analysis

AAPL data

	Date	VOLUME	LOW	HIGH	OPEN	CLOSE
0	2015-07-10	61354500	117.5345	120.0944	118.2423	119.5417
1	2015-07-13	41440500	120.5502	121.9465	121.2386	121.8495
2	2015-07-14	31768100	121.2483	122.5380	122.2180	121.8011
3	2015-07-15	33649200	121.7720	123.2944	121.9077	122.9744
4	2015-07-16	36222400	123.4883	124.6713	123.8665	124.6131



Fig[10]: View of Dataset and Options for Spark Analysis

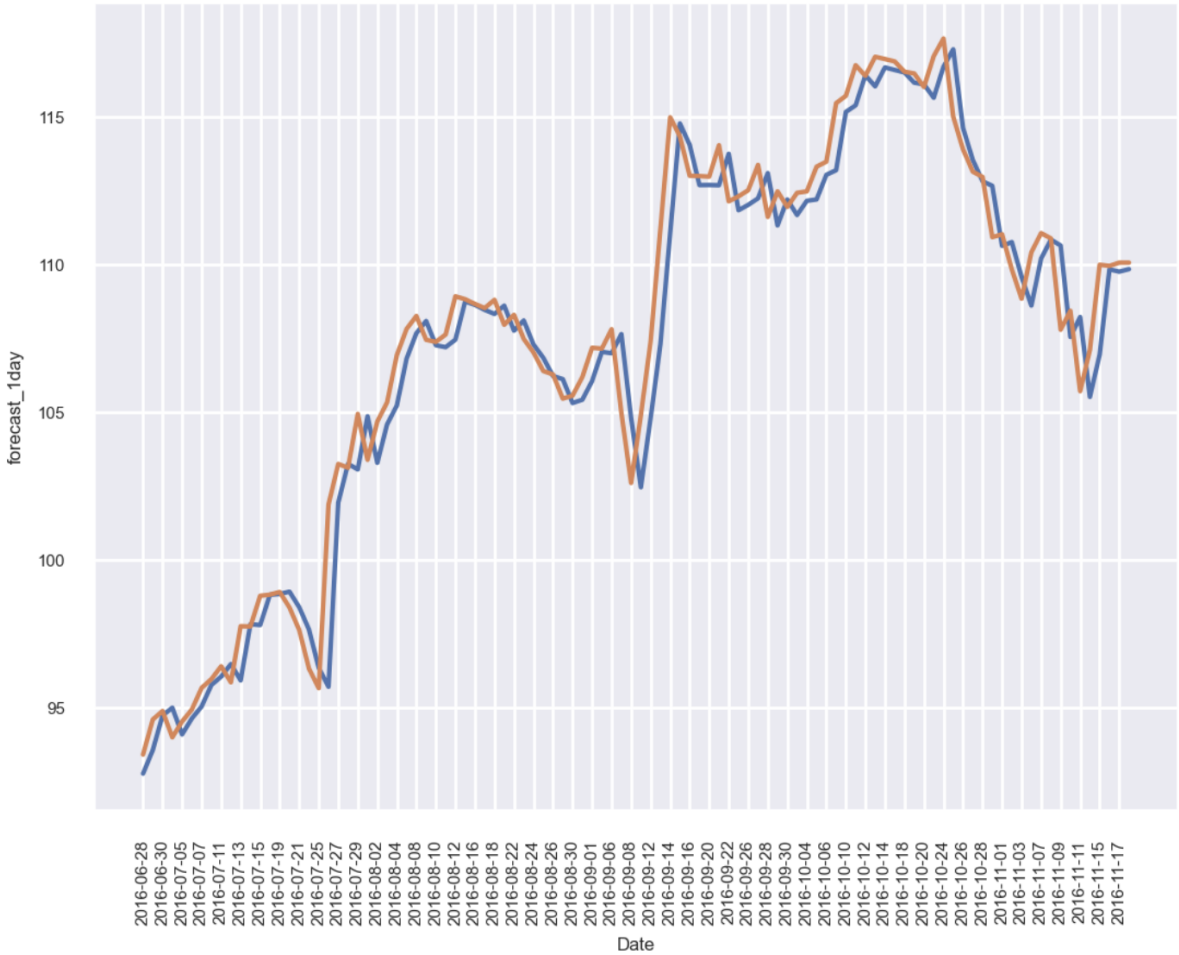
We use autoregressive model of the form

$$Z(t+n) = f(Z(t), Z(t-1), Z(t-2), Z(t-3), Z(t-m))$$

Where $Z(t+n)$ is the value of the n-day forecast. $Z(t)$ is the current value at the time step t and $Z(t-m)$ is the value at time-step t-m (with a lag-length m). And $Z(t)$ is obtained by differencing the raw stock price data.
We currently did not employ polynomial features and features generated from residuals in previous steps (ARIMA model). There would have been better results with these algorithms but since these would take a large amount of memory and computing power these were not utilized.

	Row Number	Date	CLOSE	DeltaZ1	forecast_1day	actual_1day	De
0	241	2016-06-28	92.6103	0.1552	92.7655	93.4118	0
1	242	2016-06-29	93.4118	0.1529	93.5646	94.5992	0
2	243	2016-06-30	94.5992	0.1345	94.7337	94.8862	0
3	244	2016-07-01	94.8862	0.1117	94.9979	93.9956	0
4	245	2016-07-05	93.9956	0.0978	94.0934	94.5299	0
5	246	2016-07-06	94.5299	0.1036	94.6336	94.9357	0
6	247	2016-07-07	94.9357	0.1037	95.0394	95.6679	0
7	248	2016-07-08	95.6679	0.0967	95.7646	95.9648	0
8	249	2016-07-11	95.9648	0.0841	96.0488	96.4002	0
9	250	2016-07-12	96.4002	0.0743	96.4745	95.8559	0

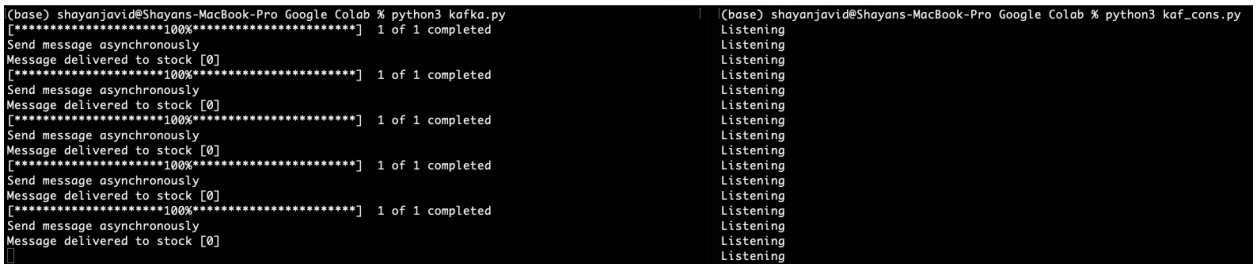
Fig[11]: Complete Spark Results



Fig[12]: Spark Forecast Results for Day 1

Data Streaming :

We are using Kafka for the streaming of live stock data from Yahoo Finance. Yahoo Finance offers a wide range of market data on stocks. So we chose Yahoo Finance.
Kafka has a producer and a consumer where the producer creates a topic onto the Kafka server and the Consumer subscribes and receives messages from that Topic. When a user selects a company from the drop menu, the live stock data related to that particular company is fetched from Yahoo Finance and given to the Producer. The Kafka producer creates a Topic and publishes data onto that Topic to the Kafka server. Then the Kafka consumer subscribes to that particular topic and receives the live stock data. Then the live stock data received is converted into a csv file and saved locally. The csv file is loaded and displayed onto the Streamlit UI for the user to view the necessary live stock market data.



Fig[13]: Kafka Streaming (Consumer & Producer)

Here, when the user selects a company, the live streaming data like the Opening price, Closing price. Volume etc is being displayed. But sometimes, when the market is closed we get the constant values from Yahoo Finance.

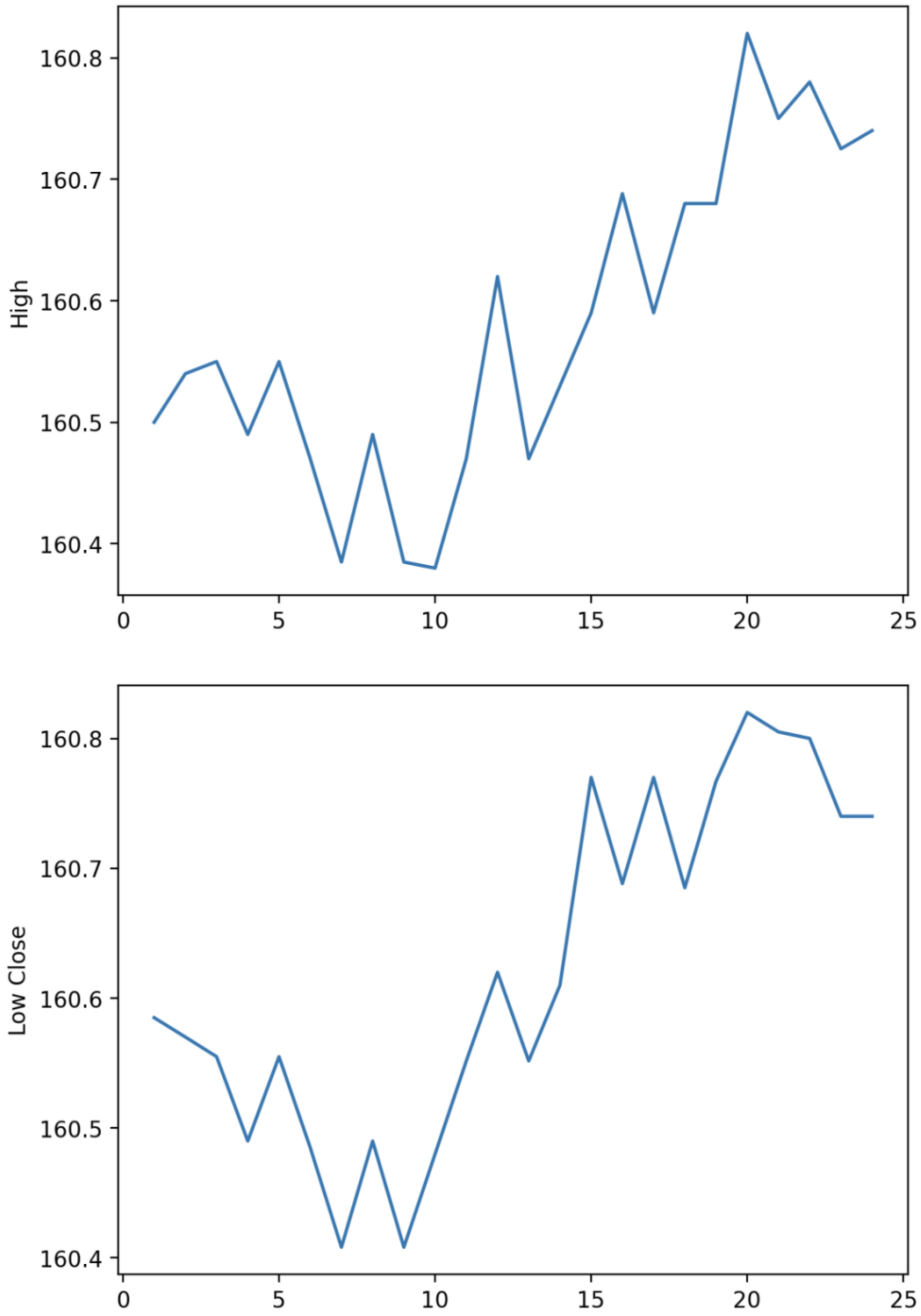
LIVE DATA

Select one of the 5 top S&P 500 companies for live data!

AAPL

	Open	High	Low Close	Adj Close	Volume
0	160.5000	159.9800	160.2300	160.2300	0
1	160.5000	159.9800	160.2300	160.2300	0
2	160.5000	159.9800	160.2300	160.2300	0
3	160.5000	159.9800	160.2300	160.2300	0
4	160.5000	159.9800	160.2300	160.2300	0
5	160.5000	159.9800	160.2300	160.2300	0
6	160.5000	159.9800	160.2300	160.2300	0
7	160.5000	159.9800	160.2300	160.2300	0
8	160.5000	159.9800	160.2300	160.2300	0
9	160.5000	159.9800	160.2300	160.2300	0

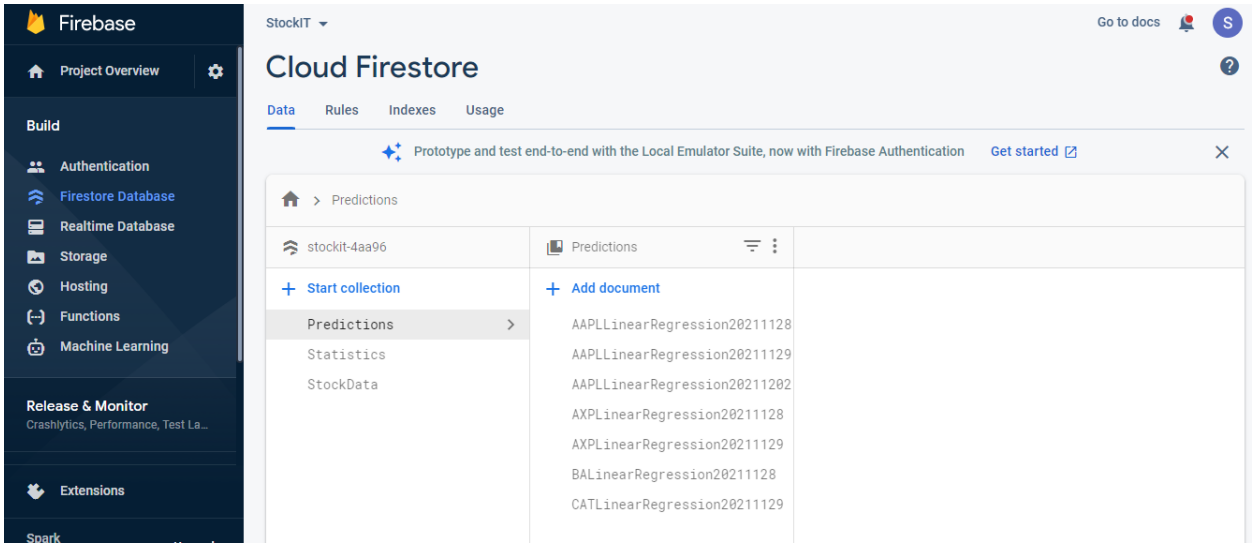
Fig [14]: Live Data



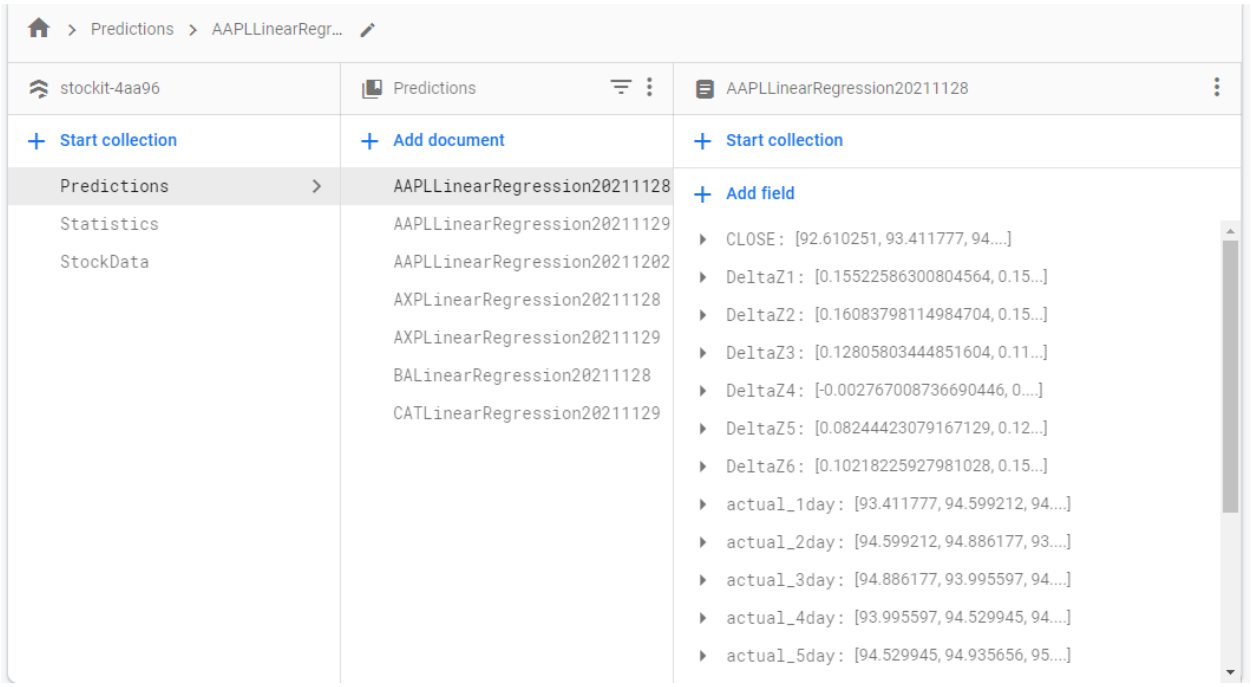
Fig[15]: Live Data

Firestore Database

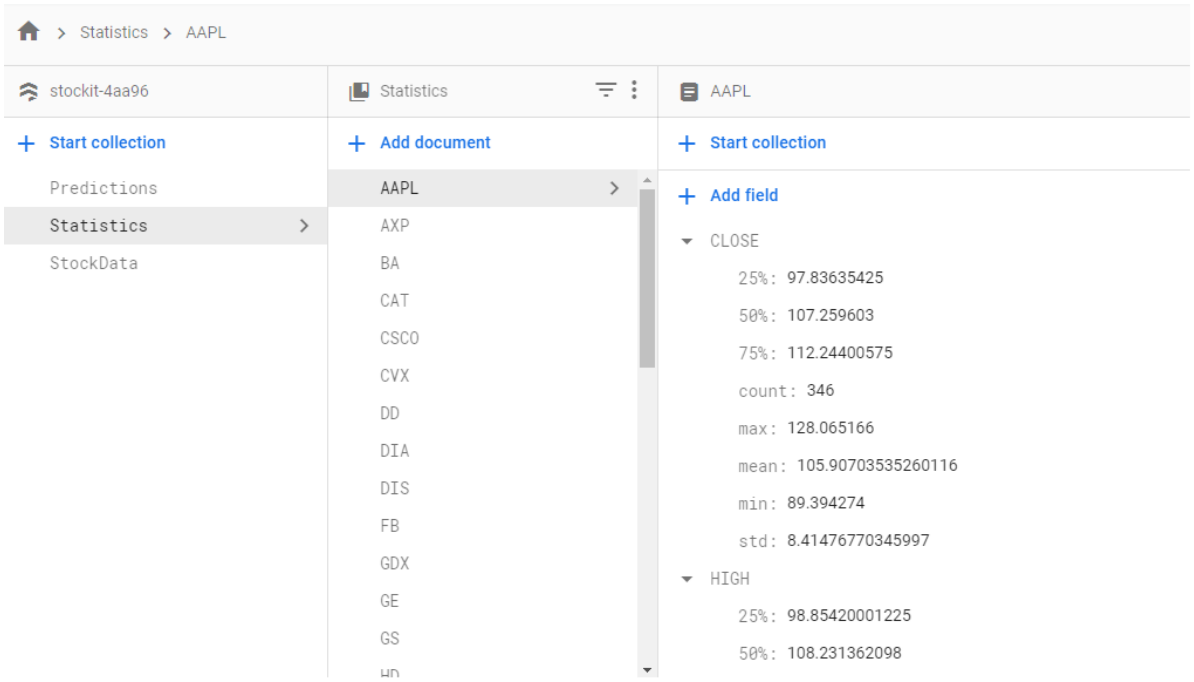
We have utilized firestore as our cloud database. The application frequently interacts with the database to fetch results and also to update the existing data. We perform a few data cleansing activities before pushing the data and fetched data is transformed to required format before being displayed in the app. The database in itself



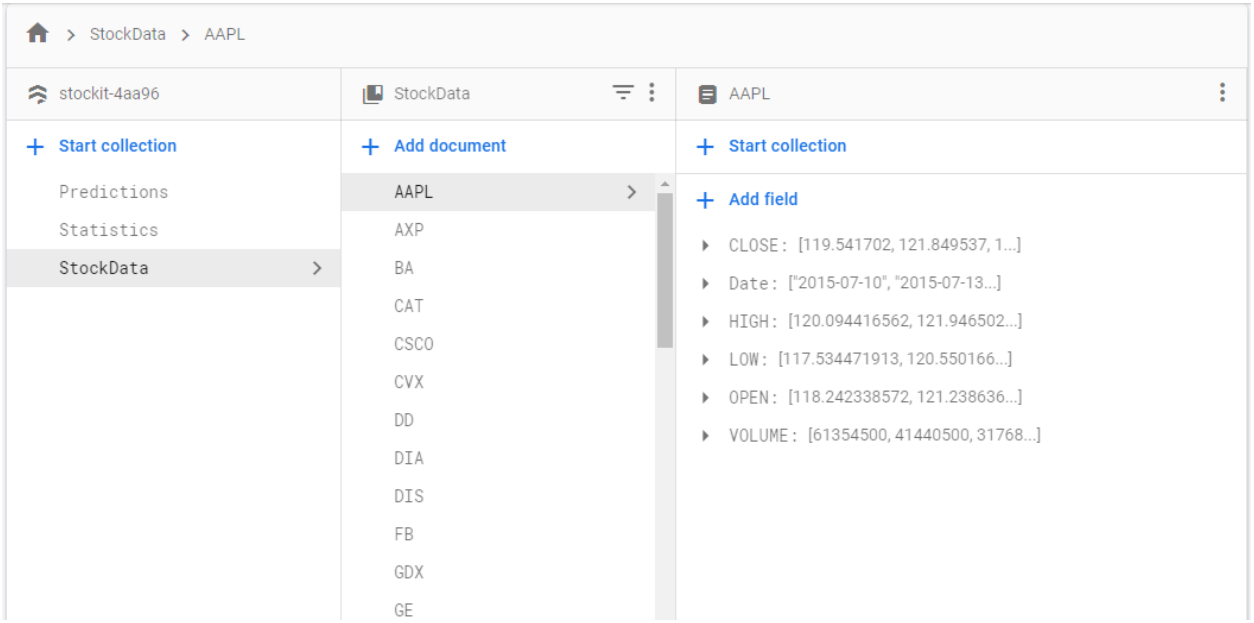
Fig[16]: Database Structure



Fig[17]: Predictions Document



Fig[18]: Statistics



Fig[19]: Stock Data

Learning Experiences

- Caching results in Streamlit for faster seek time of results
- Managing live data using Apache Kafka
- Using Spark Mlib
- Utilizing Prophet for time series forecasting
- Sentiment analysis and pdf metadata extraction
- Integrating database and data storage into our project

Project Link:
https://drive.google.com/drive/folders/162EoqORXaC4KpLqdu5_vPCfvagxbCJaW?usp=sharing