Machine Learning Nanodegree

Alfonso Ridolfo

2021

1 Domain Background

Investment firms, hedge funds and even individuals have been using financial models to better understand market behavior and make profitable investments and trades. A wealth of information is available in the form of historical stock prices and company performance data suitable.

Using free or paid sources of historical data, is possible to produce predictions on stock market.

This is a very well known and yet still open challenge investigated by a lot of people in both machine learning and trading fields.

A lot of proposed approaches to guess stock prices future dynamics are present in literature, most of them rely on domain-specific and traditional techniques such as fundamental analysis, technical analysis and quantitative analysis. Other uses more recent and "imported" approaches, based on mathematical models derived from other scientific or engineering fields.

This project will try to approach this problem designing and training a machine learning based model to predict stock prices.

2 Problem Statement

Given historical stock prices and company performance data, is possible to create a stock price predictor that gives a forecast of a given stock price value in a given time series interval.

For the purpose of the project, the following set of stocks data have been analyzed:

1. IBM
2. AAPL (Apple Inc.)
3. AMZN (Amazon.com Inc.)
4. GOOGL (Alphabet Inc.)

This will be a typical time-series prediction problem in the context of machine learning.

Multiple features regarding these stocks are available from multiple data sources and can be leveraged in order to produce a prediction for a given date range.

Not all the feature are actually available in advance, hence a real world scenario evaluation would be useful in these respects.

The purpose of the model is to predict the future values of "Adjusted Close" (closing price adjusted for stock splits and dividends).

3 Datasets and Inputs of the project

3.1 Dataset

The project uses as training input a dataset retrieved from Yahoo finance API

The longest track record contains is about the IBM stock and starts from January 2nd, 1962.

The resulting idataset, constitutes a baseline dataset.

Dataset will be then split in Train, Test and Validation sets.

4 Solution Statement

This project will implement a candidate solutions to the above described problem, based on a SageMaker offered implementation of autoregressive probabilistic Recurrent Neural Network model called DeepAR .

In the project, the model is used to predict daily adjusted close value of the above listed stocks.

These model will work on these inputs:

1. A stock ticker name from paragraph 2 and
2. a target time series start date.

The model is trained on the input dataset and, at inference time, will provide a 20 days long time series target prediction on "*Adjusted Close*" values starting from the input start date.

Context length of the prediction will be same as prediction length (20 days).

For additional details about DeepAR implementation and usage refer to Amazon SageMaker related documentation.

5 Benchmark Model

A Simple Moving Average on the Adjusted Close values will be used as benchmark for the solution model.

The performances will be compared by metrics values as in paragraph 6.

6 Evaluation Metrics

The solution will be ranked against the evaluation metrics of choice and compared to the benchmark model.

The metrics of choice are:

- Root Mean Square Error (RMSE),
- Mean Absolute Percentage Error (MAPE),
- Mean Absolute Error (MAE) and
- Coefficient of determinationknown also as R^2

7 Project Design

A DeepAR model has been implemented in mostly using Python language and Jupyter-Lab Notebook, using SageMaker available container for this kind of Recurrent Neural Network based estimator.

Following tasks has been planned:

- Loading and exploring the data (EDA).
- Feature engineering;
- Slicing of training, test and validation sets on time series, with a custom strategy implementation;
- Formatting of input (test/train/validation/benchmark) data JSON files and upload to S3;
- Instantiation and training of a DeepAR estimator;
- Model deployment and predictor object instantiation to pass data to the resulting endpoint;
- Evaluation of predictor performances on validation data.

An AWS Lambda Function will be implemented to prepare the data before calling the endpoint runtime.

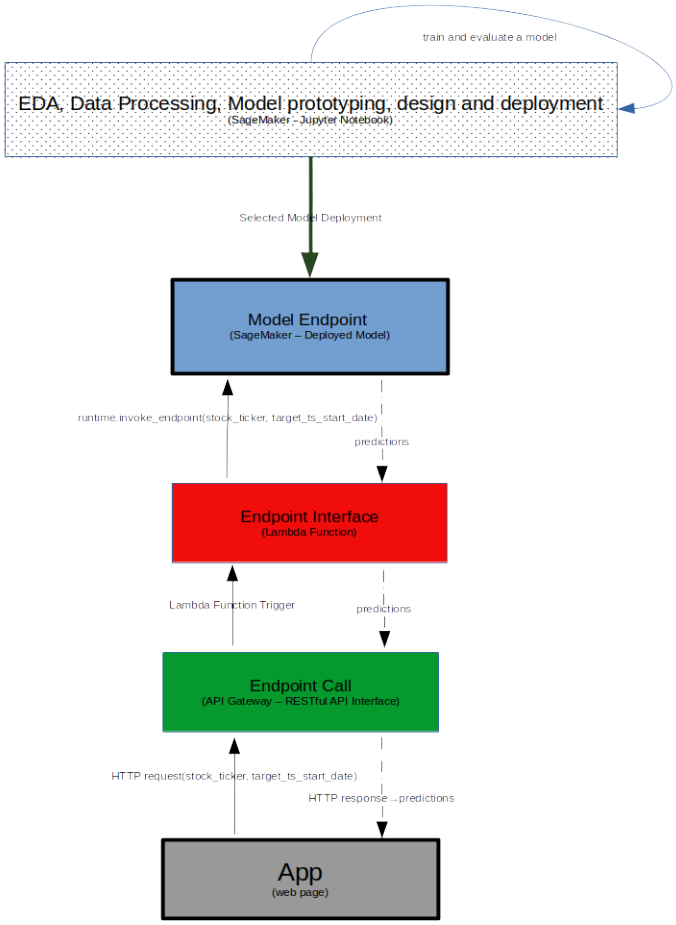
The AWS Lambda Function will be called by a RESTful HTTP web API.

A Web Page/Application will be provided as application entry-point to the user, allowing him to select:

1. one of the analyzed stocks (ref. paragraph 2) and
2. an input time series start date value.

Prediction values will be then reported to the user.

A general sketch of the architecture is hereby reported:



8 Project development process description

The project revealed to be challenging and yet very useful to understand time series prediction problem in all of its differences respect to other machine learning predictions problems.

8.1 Technologies used

Python and SageMaker have been the technologies mostly used to implement it, with a little exploration of web based front-end frameworks and technologies (HTML5, JavaScript, RESTful APIs, AWS Lambda).

8.2 Deviations from initial decisions

The project has been developed as defined in project proposal document, with some minor changes and refinement in choices about data, hereby listed:

- Data sources selection
- Data splitting
- Additional features usage

8.2.1 Changes and refinements explanations

Hereby follows additional details and explanations about the above described changes, refinement and deviations respect to initial considerations.

Data sources selection Data used for the project has been retrieved from from Yahoo finance only.

Initially, two other sources have been evaluated, with the aim to improve model performances:

- Alpha Vantage API and
- Quandl python API.

After some analysis on their documentation and some trial with their APIs, has been observed that the additional information provided (at a price) by these sources are almost only higher dataset frequency respect to the daily frequency offered by free Yahoo finance API.

Since Adjusted Close is a daily registered value, no real advantage could be expected to come from an higher frequency sampling on input data. Hence, Yahoo Finance is the only source of data used for the project.

Data splitting Input time series has been temporarily sliced into 3 different dataset Train, Test and Validation. Although not prescribed by DeepAR documentation, using an additional split on data allows to better evaluate the performance of proposed model on previously unseen data (the validation set).

Additional Features Usage A model that does not make used additional features aside from the target value alone has been produced. In a real world scenario, most of the additional features gathered and available in the dataset are not available beforehand when a prediction would be useful (e.g. one week ahead of time or a day ahead of time).

8.3 Notebooks

A set of Jupyter Notebook has been implemented to conduct the following steps:

1. Data Gathering
2. Exploratory data analysis
3. Benchmark model
4. DeepAR model implementation

These notebooks have to be run in sequence, each of them reusing the kernel from the previous notebook in order to access the data as already processed:

1. the second notebook shall reuse the kernel from the first,
2. the 3rd shall reuse the kernel from the second and
3. the 4th the kernel from the 3rd.

8.3.1 Data Gathering

This notebook simply retrieves selected stock data from Yahoo finance. It is very simple and needs no real explanation, asides from the fact that some additional feature has been prepared to be used by a future model implementation that could leverage it.

8.3.2 Exploratory data analysis

This notebook explores data features. No real seasonal pattern has been found in the data.

Some features engineering on the input data has been proposed.

A couple of indicators also used in traditional technical analysis from stock market theory have been computed and added to the dataset as additional features:

1. Simple Moving Average and
2. Bollinger Bands indicators.

Three different window sizes has been used to compute the above cited indicators:

- 10 days
- 20 days
- 50 days

Some hints (from correlation matrix) on features that could be used for future model training that would use additional feature (not part of this project).

8.3.3 Benchmark model

Dataset splitting In this notebook, data has been split into:

- train set, = the entire dataset, less the last 40 days of the time series,
- test set = the entire dataset, less the last 20 days of the time series,
- validation set = last 20 days of input time series.

This kind of slicing has been chosen given two considerations:

1. best practices from DeepAR documentation recommends to use a test set that includes the training set and adds exactly prediction_length elements at the end of the time series.
2. I would like to check model performance on previously unseen data, so I sliced an additional 20 days of data from the end of the time series..

Benchmark model implementation In this notebook, **Simple Moving Average** (SMA) computed in the previous notebook (an used there as an indicator) is used as benchmark model to predict future values for the price **Adjusted Close** value.

This model is really simple and hasn't a real training phase that relies on the test set to be accomplished. However, to avoid leakage from train-test set into validation set, all the **SMA** predicted values values from validation set has been set to the last value computed on the training set. This has been also applied to Bollinger Bands indicators.

The 20 days window based **SMA** has been used as reference for all metrics evaluation.

A prediction from this model should be interpreted as follows: a constant value set to last SMA value computed on training data is the predicted value of future Adjusted Close value both from Test set and from Validation set.

Metrics evaluation on test data Given the presently data available today, 26/04/2021, the results given by the benchmark model on test data, starting from 01/03/2021, are the following:

Stock	**RMSE**	**MAPE**	**MAE**	R^2
IBM	8.137	0.0534	6.936	-2.61
AAPL	9.842	0.0785	9.523	-14.65
AMZN	200.624	0.0635	193.726	-13.79
GOOGL	26.737	0.0109	22.454	-0.0004

Metrics evaluation on validation data Given the presently data available today, 26/04/2021, the results given by the benchmark model on validation data, starting from 29/03/2021, are the following:

Stock	RMSE	MAPE	MAE	R^2
IBM	16.019	0.114	15.625	-19.56
AAPL	5.171	0.032	3.983	-0.1
AMZN	115.034	0.031	102.771	-0.07
GOOGL	196.566	0.079	178.499	-4.68

Consideration about SMA performances Clearly, Simple Moving Average, from the fact that it is computed on training data, tends to perform better on test data, which follows in time training data.

8.3.4 DeepAR model implementation

In this notebook, the model of choice has been implemented.

DeepAR is a probabilistic autoregressive Recurrent Network based model. This project uses its AWS SageMaker implementation.

This model is trained on Adjusted Close data from the training set and then used to predict Adjusted Close value on both test set and validation set.

The model can be also used to predict future data.

In any case the model will produce a

`prediction_length=20 days`

long prediction.

Data preparation The Adjusted Close feature has been isolated in each time series to be fed to the model.

The data to been serialized into JSON format for future use.

Model training A model has been trained using the following hyperparameters:

Hyperparameter Name	Value
prediction_length	20
context_length	20
time_freq	D
epochs	300
early_stopping_patience	40
num_layers	4
num_cells	40
mini_batch_size	128
learning_rate	1e-3
dropout_rate	0.12
likelihood	student-T

Model deploy The model has been deployed to a SageMaker endpoint named "DeepAR-ml-spp"

A python class named

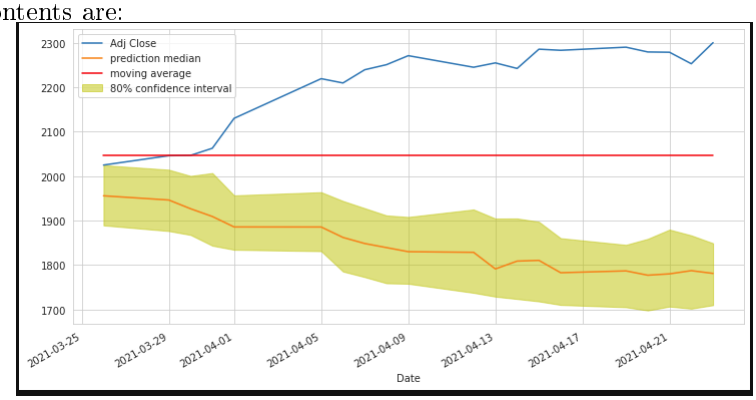
DeepARPredictor

, that takes as input a Pandas dataframe and, invoking the endpoint, asking to predict Adjusted Close value from the first date not contained into the input dataframe.

The resulting predictions have been also serialized for future analysis.

Performance visualization The predictions made on validation data have been visualized using python matplotlib visualization library.

One of the plots is here reported as an example of what notebook plots contents are:



- Adjusted close ground truth is shown in blue,
- prediction median (the 50% quantile) is shown in orange,
- the area depicted in yellow is the area comprised between the 10% and the 90% quantiles,
- the red line is the benchmark model (as said before, it is a constant value respect to the validation dataset).

Metrics evaluation on test data Given the presently data available today, 25/04/2021, the results given by the DeepAR model on test data, starting from 01/03/2021, are the following:

Stock	**RMSE**	**MAPE**	**MAE**	R^2
IBM	25.33	0.175	22.62	-33.98
AAPL	14.54	0.109	13.41	-33.21
AMZN	189.584	0.055	168.538	-12.21
GOOGL	230.015	0.106	217.567	-73.04

Metrics evaluation on validation data Given the presently data available today, 25/04/2021, the results given by the DeepAR model on test data, starting from 01/03/2021, are the following:

Stock	RMSE	MAPE	MAE	R^2
IBM	24.30	0.159	21.82	-46.17
AAPL	26.29	0.179	23.77	-27.52
AMZN	542.32	0.152	506.99	-22.98
GOOGL	404.20	0.168	378.35	-22.97

8.3.5 Considerations about resulting metrics values and benchmark performances comparison

As can be seen from the above tables, is evident that on this kind of dataset and with the above described set of hyperparameters, DeepAR has very poor results in predicting future Adjusted Close values. The benchmark model outperforms the DeepAR model with respect to all the metrics both on test data and on validation data set.

8.4 Web App

A frontend to interrogate DeepAR deployed model endpoint has been implemented using Flask web framework, JavaScript and HTML5.

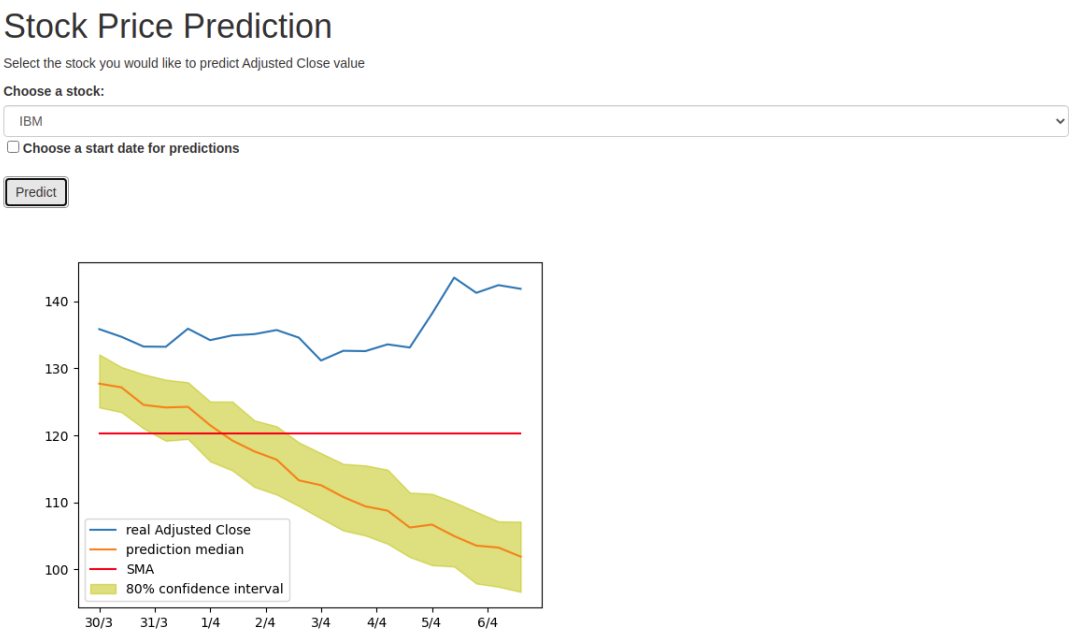
The web application allow to select:

1. a stock and
2. a prediction start date (optional)

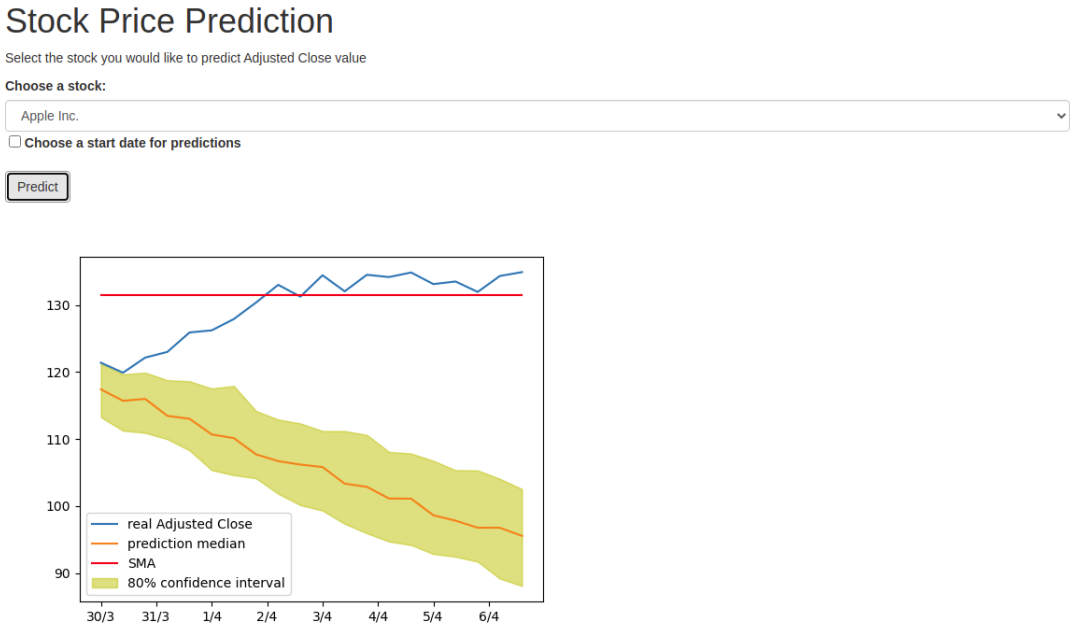
If no start date has been selected, predictions will be done on validation, feeding the model with test data.

Results will be visually compared to predictions made by benchmark model (SMA) and to ground truth values.

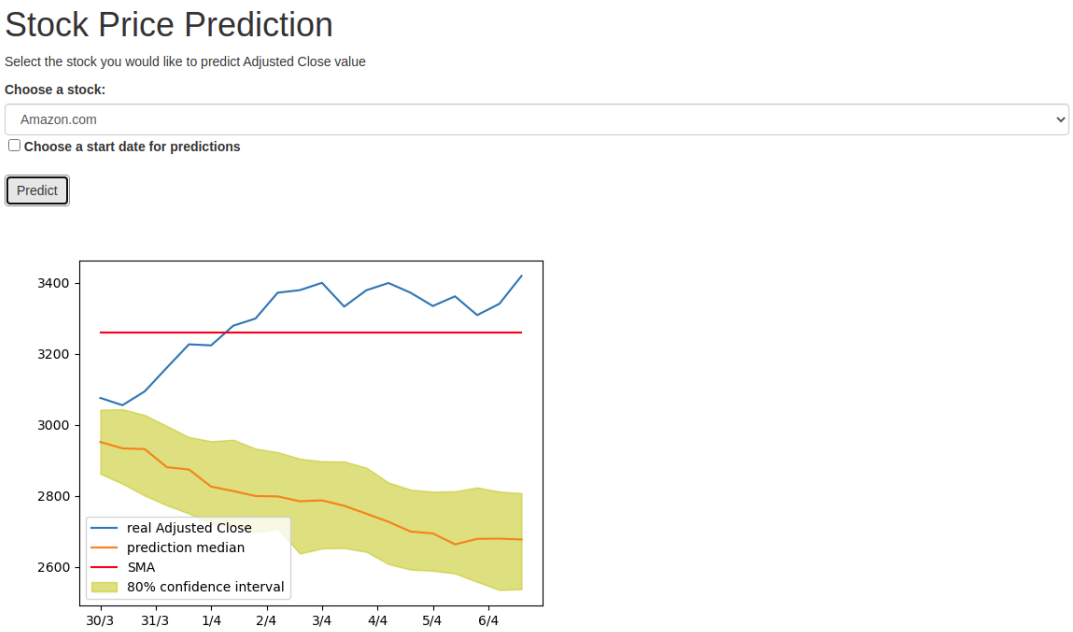
Here follows a screenshot a prediction on IBM stock test set:



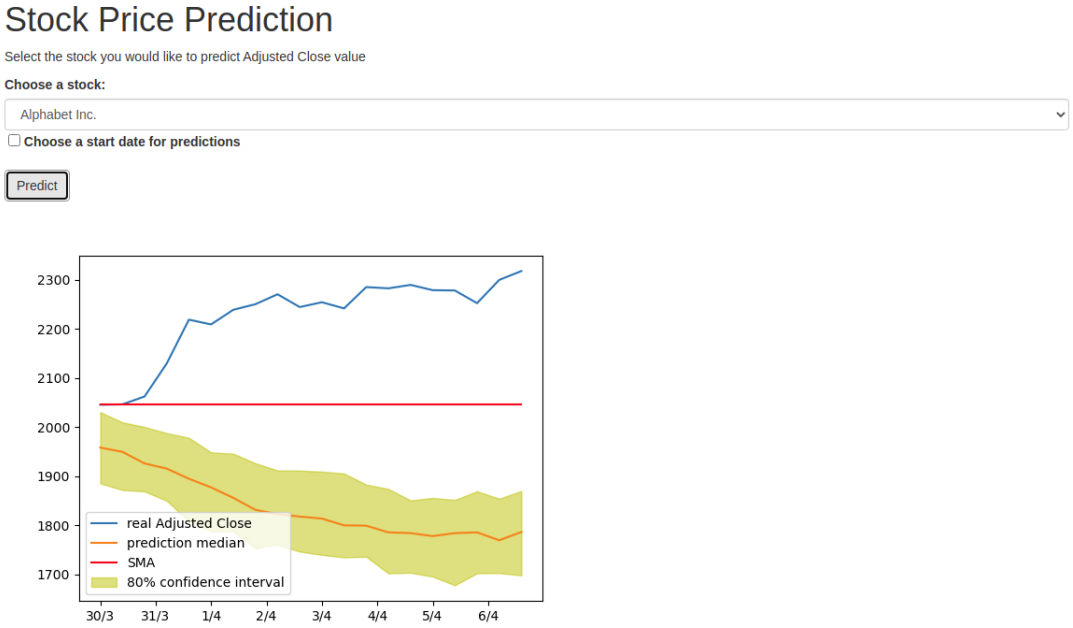
A screenshot of AAPL stock price predictions on test data:



A screenshot of AMZN stock price predictions on test data:

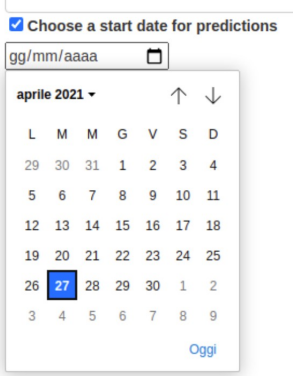


A screenshot of GOOGL stock price predictions on test data:

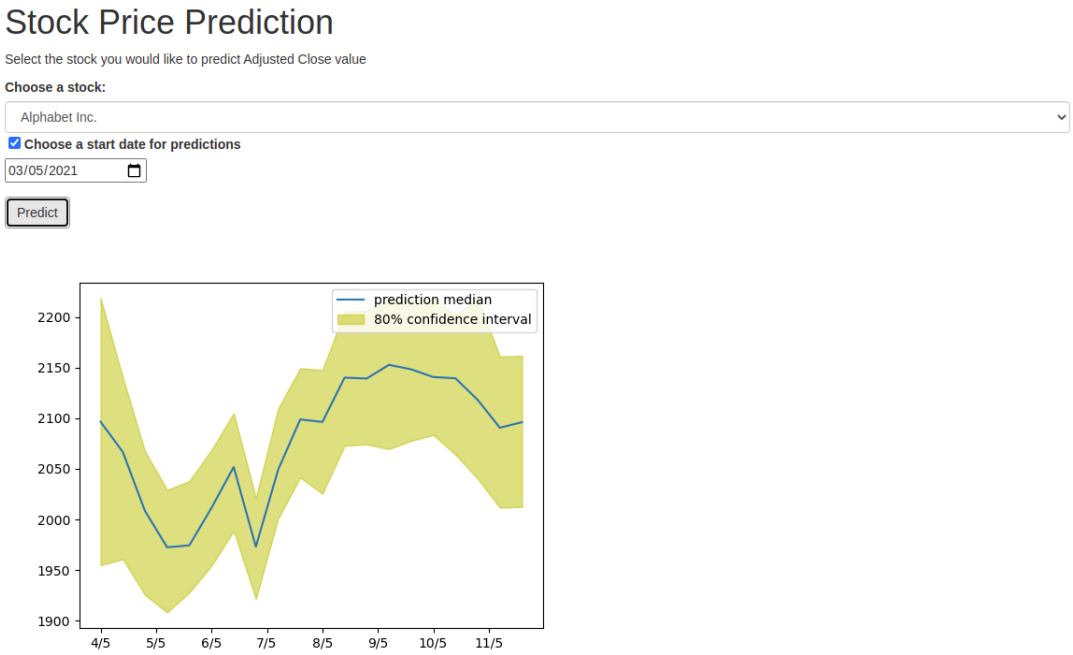


If a start date is chosen by the user, the model will be fed to DeepAR model endpoint, by means of API Gateway and Lambda function, with a start date as

chosen from the calendar widget:



A prediction request for an arbitrary date selection will show the following output (made on GOOGL stock for date 03/05/2021):



9 Conclusions

The DeepAR model appears to have poor performances in stock price prediction task. As already observe, a Simple Moving Average could outperform it very easily and respect to all most used metrics.

Just to make some hypotesis on the motivation behind the poor performance, it is possible to observe that from its documentation it seems that this kind of model is actually defined to leverage a couple property of input data:

1. some kind of correlation among the time series provided as input,
2. seasonalities (that the model itself detects and exploits) in timeseries data,
3. a huge amout of time series as input (hundreds of related time series)

The point 1 is also in some contraddiction to the Random Walk Hypotesis. Because of these reasons, its performance on this dataset are quite justifiable. Maybe better performance could be achieved using dyanmic feature data as train and inference input data. But, as said in 8.2.1, its results would be probably not usable in a real world scenario.