# Using AutoGluon to predict Stock Prices - Capstone Proposal

(This subject is based in one of the [Udacity suggestions](https://docs.google.com/document/d/1ycGeb1QYKATG6jvz74SAMqxrlek9Ed4RYrzWNhWS-0Q/pub)[1] for capstone projects)

## Domain Background

Individuals, hedge funds and all kinds of investors have been using different types of financial models to make profitable investments on company stocks. With the [scientific](https://en.wikipedia.org/wiki/Stock_market_prediction)[2] and [technological advances](https://scholar.google.com/scholar?hl=en&as_sdt=0%2C5&q=machine+learning+stock+prediction&btnG=) in the last years, Machine Learning has also become an option to achieve these objectives.

## Problem Statement

A crucial part of making profitable investments is to be able to estimate future close prices of a stock, based on its current trading information (Open/Close/Highest prices, volume, etc.) as well as the historical data, which means, that data in the past.

For example, let’s assume we have daily trading information for Apple (AAPL) for February 2021

**Sample Input, (February Lookback)**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Date** | **Open** | **High** | **Low** | **Close** | **Volume** | **Adjusted Close** |
| 2021-02-16 | 135 | 136 | 133 | 133 | 80576300 | **132** |
| 2021-02-17 | 131 | 132 | 129 | 131 | 97918500 | **130** |
| … | … | … | … | … | … | **…** |
| 2021-02-25 | 125 | 126 | 121 | 121 | 148199500 | **120** |
| 2021-02-26 | 123 | 125 | 121 | 121 | 164560400 | **121** |

And we want to predict the Adjusted Close price for March. The following would be the solution outcome.

**Sample Output (March Forecast)**

|  |  |
| --- | --- |
| **Date** | **Predicted Adj Close** |
| 2021-03-01 | **118** |
| 2021-03-03 | **124** |
| … | **…** |
| 2021-03-30 | **135** |
| 2021-03-31 | **140** |

Needless to say, given the prediction consists of real numbers, this is a regression problem.

## Solution Statement

The proposed solution is to use [AutoGluon](https://auto.gluon.ai/)[3] (for simplicity reasons) within AWS ecosystem to make these predictions for a specific ticker. The input will be the stock data for a date range in the past, and the output will be the predicted close price for a later date range.

We are going to use the best model suggested with Autogluon.

The final user is intended to provide:

* Ticker (for example, AAPL)
* Lookback date range (for example, Jan-Dec 2021)
* Forecast date range (for example, Jan 2022)

After acquiring the lookback data, an AutoGluon model will be trained. Finally, the model will be queried providing the forecast date range to get the predicted stock prices (adjusted close).

## Datasets and Inputs

The dataset is actual trading data from Yahoo Finance. It will be acquired **on demand** by performing a GET request to a parameterized url (dependent on the ticker and a date range).

https://query1.finance.yahoo.com/v7/finance/download/**{ticker}**?period1=**{period1}**&period2=**{period2}**&interval=1d&events=history&includeAdjustedClose=true

For example:

https://query1.finance.yahoo.com/v7/finance/download/AAPL?period1=1539907200&period2=1618790400&interval=1d&events=history&includeAdjustedClose=true

which will return a CSV like the following:

Date,Open,High,Low,Close,Adj Close,Volume

2021-02-16,135.490,136.009,132.789,133.190,132.403,80576300

2021-02-17,131.250,132.220,129.470,130.839,130.066,97918500

2021-02-18,129.199,130.000,127.410,129.710,128.943,96856700

2021-02-19,130.240,130.710,128.800,129.869,129.102,87668800

2021-02-22,128.009,129.720,125.599,126.000,125.255,103916400

This data comes clean, so it does not need transformations or processing.

The dataset includes the following fields

|  |  |  |
| --- | --- | --- |
| **Field Name** | **Type** | **Description** |
| Date | Datetime | Date of the stock data |
| Open | Float | Open price at the beginning of the day |
| High | Float | Highest price during the day |
| Low | Float | Lowest price during the day |
| Close | Float | Close price |
| Adj Close | Float | Adjusted close price, takes into account dividends and splits |
| Volume | Int | Transaction volume for the day |

## Benchmark

To have a reference point to compare to, the forecast range will be always in the past, so we can compare with the ground truth, this is called **backtest**.

According to the example in the problem statement, we would train the data for February but we would make a prediction for March and then compare it with the ground truth (actual prices from march, which is the reference point).

It would be ideal to use another trained model as a benchmark, a well known. However, running another model would add complexity to the project. Therfore, for simplicity we’re going to use the moving average (average of the previous n-points) as benchmark model.

## Evaluation metric

Given the regression nature of this problem, the model is going to use the RMSE (root mean squared error) as evaluation metric.

RMSE is Autogluon’s default for regression. However, for the inference reports, it transforms the RMSE (and any metric) to a “score” where lower is better, including negatives. So, RMSE will be used internally but we’ll see a different normalized score in the final reports.

## Project Design

Let’s start by the Web UI (hosted on S3 with Cloudfront). There will be a form where the user will specify the ticker, lookback and forecast periods in months:

Interfaz de usuario gráfica, Texto, Aplicación

Descripción generada automáticamente

When clicking the “Backtest Forecast” button, the whole ML flow will be executed on AWS.

Gráfico

Descripción generada automáticamente con confianza baja

1. When the use clicks the “Backtest forecast” button, it will call the Gather Data lambda (through API Gateway), sending it the user params filled in the form (ticker, looback period, forecast period)
2. The lambda will use the user params form a url to request Yahoo a CSV with the input data. Then, it will create the train and test split CSV files. If we find processing is needed, will be done here.
3. The data CSVs will be stored in S3. Actually, it will consist on 2 files, a train split csv and a test split csv. A training job name will be formed.
4. The training job name is returned to the UI (to start polling, step 8)
5. The S3 data storage will trigger the training lambda. (This has been based on this [AWS blog post](https://aws.amazon.com/es/blogs/machine-learning/code-free-machine-learning-automl-with-autogluon-amazon-sagemaker-and-aws-lambda/)[4]).
6. The training lambda will launch a training job (using AutoGluon algorithms), which will also run a batch transform to get the forecast prediction
7. The model artifact and the inference results (csv) will be saved to S3
8. The UI, which knows the training job name, polls the Read Results lambda to know if the inference results are ready. (Actually, polling started right after step 4).
9. The Read Results lambda checks in the output bucket if the results are ready
10. Once ready, the lambda reads the results and converts them to JSON
11. The final JSON results are sent to the UI to be displayed

Finally, the React UI will show the backtest vs the ground truth in a chart

Interfaz de usuario gráfica, Gráfico, Histograma

Descripción generada automáticamente

This chart will allow to easily compare the prediction and the ground truth.

## Further work

There are several improvements for this project that became out of scope due to time reasons, but could be done in a future. Some of them are:

* Use data of other tickers to make current prediction more accurate
* Add more features like Bollinger bands, moving average, etc.
* Use an algorithm specifically tailored for time series, like TS-Gluon or DeepAR.
* Use Sagemaker’s Processing and Batch Transform Jobs
* Add serverless inference as an alternative to Batch Transform
* Add monitoring and profiling
* Use AWS SNS and websockets to notify the client when batch transform is done
* Compare with a proven domain specific benchmark model

## References

[1] Udacity, *MLND Capstone Project Description*, <https://bit.ly/3sUY53e>

* [2] Wikipedia, *Stock Market Prediction,* <https://en.wikipedia.org/wiki/Stock_market_prediction>
* [3] AutoGluon: *AutoML for Text, Image and Tabular Data*, <https://auto.gluon.ai/>
* [4] AWS Machine Learning Blog, *Code free machine learning with AutoGluon* <https://go.aws/3BEQrhh>