User Manual: Stock prediction & stock trading agent

Authors: Chengcheng Ding, Jacky Lin, Hanzheng Wang, Andrew Whitig

What can it do?

Thank you for taking your interest into this project! In this project, you can achieve: price prediction of certain stocks by AI, simulate an market environment and have an AI agent do stock operations to see how they perform!

Introduction

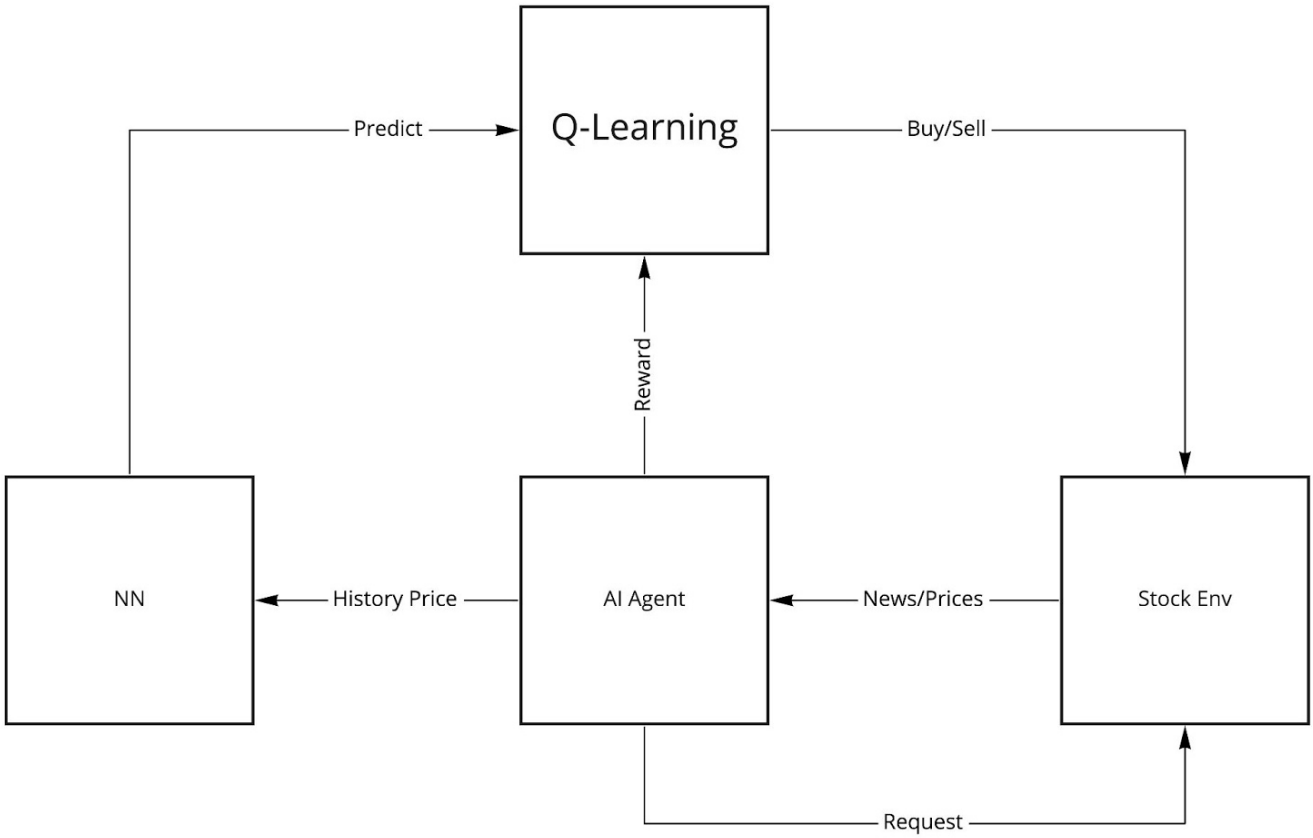
The Why (Motivation)

For long, researchers have been interested in finding ways for AI to predict stocks or do stock operations (see our report to find more details). Obviously, the applications of accurate stock price prediction are alluring. One can imagine putting one's feet up and running a program which successfully executes trades and accumulates massive wealth without lifting a finger. Unfortunately this fantasy is unlikely to ever become a reality. Stock prices are volatile and at times subjective making them extremely difficult to predict. For this reason, a wide variety in approaches and perspectives have been taken in the pursuit of AI stock price prediction. In this project, we provide a neural network that predicts the price of a certain stock relatively precisely (0.1 huber loss after trained and tuned), and an AI agent that does stock trading moderately well.

The What (Technical aspect)

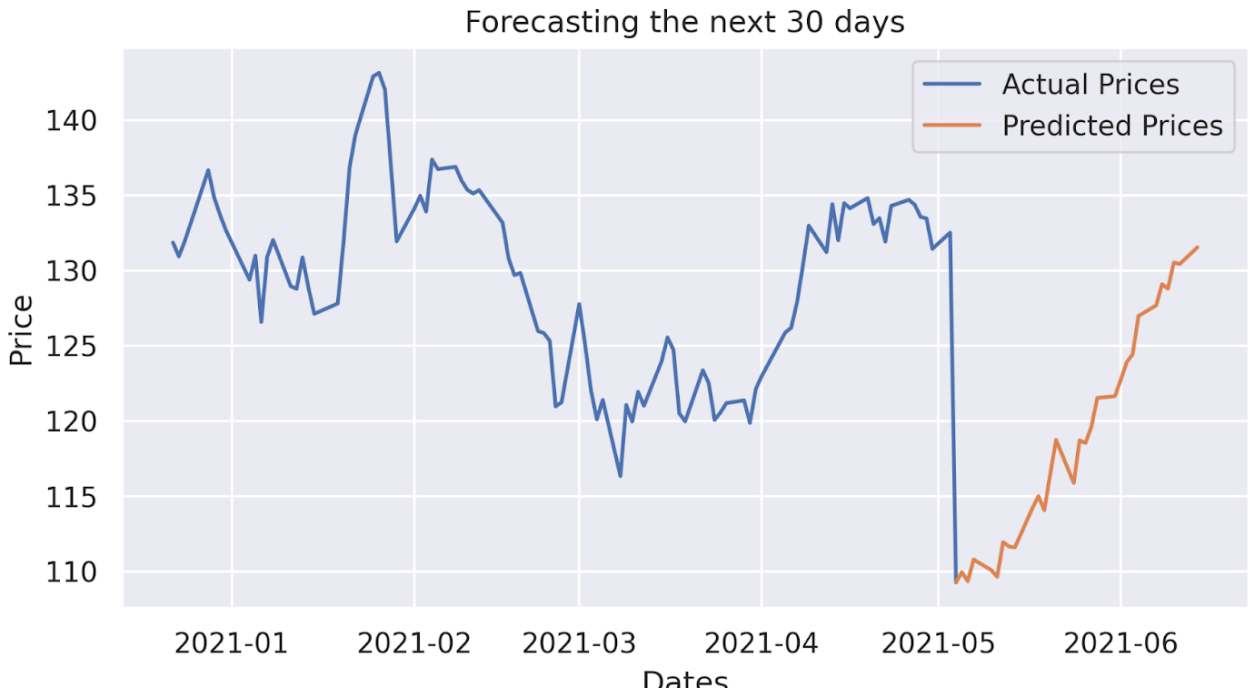
The project can be broken down into 2 generic parts: stock prediction with Neural Network (LSTM), and an agent designed to operate on a simulated stock market (currently only 1 stock) to achieve max profit.

General Design

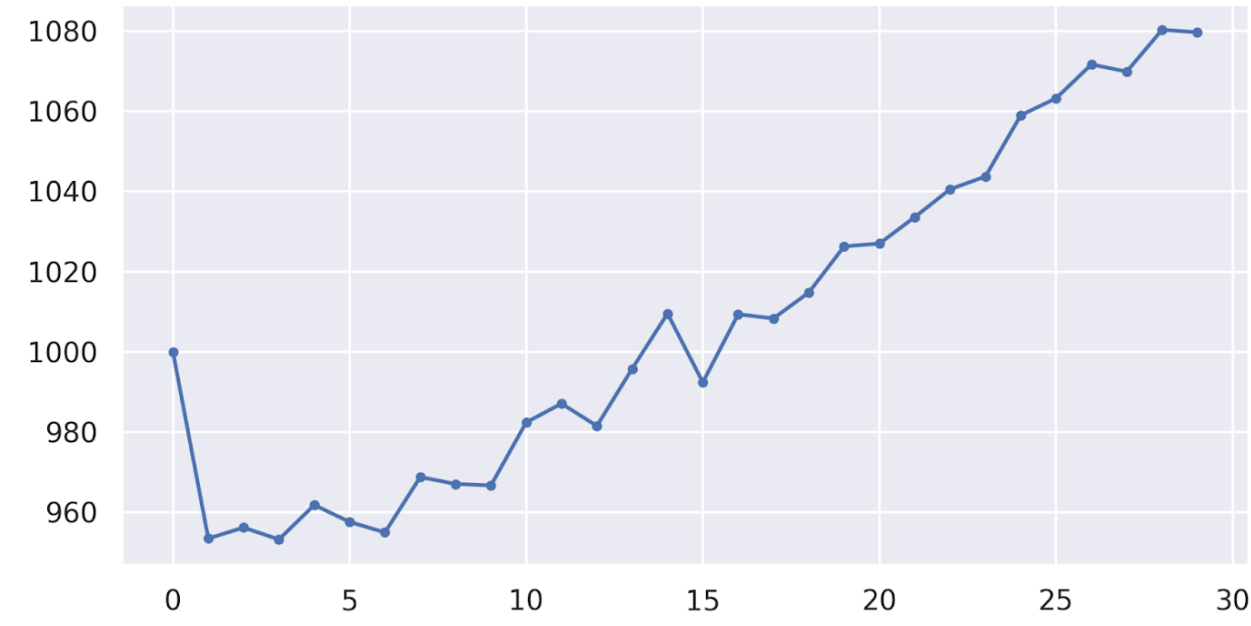


If you are interested in our source code, 3 folders that are worth looking into are :

1. `price_pred/` , which focuses on using neural network to predict future prices. Especially the nn.py, which contains the most important class that deals with price prediction NN. Here's a graph of its result:



2. `stock_env/` , which is mainly for creating a stock market environment for the q-learning agent. Especially the custom_env.py, which defines a stock trading environment that works with the gym api.
3. `agent/` , which deals with creating a q-learning agent to operate on the existing environment and try to achieve max profit. Especially the QLAgent.py, which is the main class that use the gym api to utilize reinforcement learning for operating and profiting in the designed stock environment. Here's a graph of the result achieved by the agent.



How to use the program:

Finally! The exciting part! You can run the program by yourself by simply doing:

```
python main.py [-path] [-model] [-load] [-oname] [-save_model]
```

Of course ensure you have the most up-to-date version of the following python libraries or following the version:

```
gym >= 0.18.0
numpy >= 1.19.1
pandas >= 1.1.4
tensorflow >= 2.3.0
seaborn >= 0.10.1
matplotlib >= 3.3.1
```

Options chart:

option	description
`-path`	Directory path of stock data, default is the provided `AAPL.csv`
`-model`	Type of NN model can be chosen from `lstm` or `drnn`. Default is `lstm`.
`-load`	`0` or `1`, if load saved model or train a new model. Default is `0`.
`-oname`	Name for saved files, any string is legal
`-save_model`	`0` or `1`, if save the newly trained model. Default is `0`.

If you are more familiar with computer science and want to experiment with our code yourself, feel welcomed to install Anaconda 3 with the above packages, and the ipynbs are for you to have fun.