Stock Trade: Man vs AI

Junjie Ouyang
NetID: jo326
Email: junjie.ouyang@rutgers.edu

***Abstract*—** The domain of stock market is antithesis to terms like patience and calm - people just read final values and guess a reason for it without knowing. For most ordinary people, how to do stock trading is almost like âsecret skillsâ, requires understanding and deep insight of stock market and companies. AI, on the other hand, have been widely implemented and applied in various fields, including financial industries. In fact many financial companies have their own machine learning techniques to make profit. Such techniques are so complicated and guarded as trading secrets. In real life, AI compete with human in stock trading market to keep the business running, but what if we make it a simple game?

I. Project Description

In this project, we implemented a simple little game to test the user stock trading skills against the market and a machine learning algorithm. The game is trying to simulate a simple scenario in real life: given a single changing stock, how do a human trader make trading decision to maximize the profit as possible, or at least, outrun an AI competitor? In each round of the game, a random stock will be given with a random 365-day period of time. The user and the AI will each start with 3 stocks. During the game period, each of them will do 3 ”buys” and 3 âsellsâ, respectively. In the end of game, the total amount of stock value and cash value will be recorded and compared. For each round we list out the ranking of user, AI and market, and assign them with different score values. Such score values will be accumulated through out the game process.

A. The Design Of The Project.

The deliverables for this stage include the following items:

- Technique Used:
  This project requires a whole web application framework to support our implementations, and various popular techniques are used:
  - Database: MySQL
  - Backend: Flask/Python
  - Frontend: React, HTML, CSS, Javascript
  - Visualization: D3.js
  - Machine Learning: Pandas, Scikit Learn
- Flow Diagram: The flow diagram of our project is shown below:
- High Level Pseudo Code System Description:
  1) Initially, the application collect stock data for the game purpose. These inputs will be fetched from the Alpha Vantage API and stored in the MySQL database.

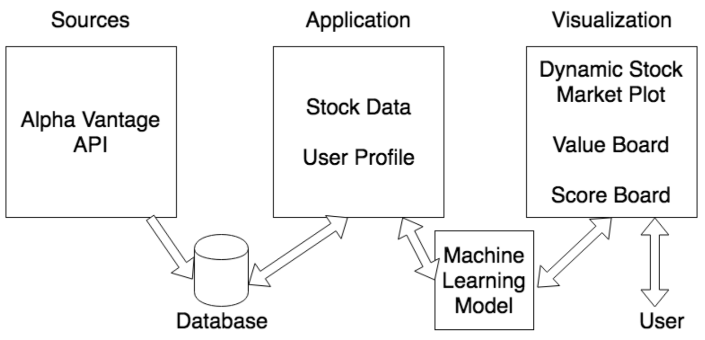


Fig. 1. Flow diagram of this project
.

  2) For each round of the game, a random stock with random 365-day period will be retrieved from the database and visualized using React in front-end, showing dynamically as simulation of real-life stock trading.
  3) User competes with AI in the game to make the most profit in the game period. For each round, each of user and AI has 3 ”buys” and 3 ”sells” choices. User input will be recorded and calculated, while machine learning algorithm make predictions and decisions for every step.
- Algorithms and Data Structures. For ML algorithm, we used Linear Regression Model in order to make predictions. JSON structures are used throughout this project, from parsing the input stock data to keep stack of the user stock data status and ML stock data status.

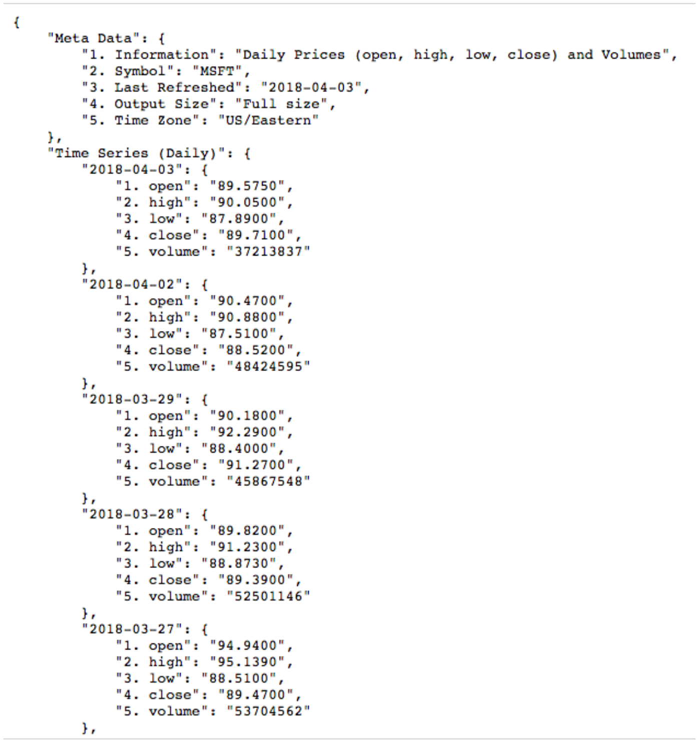
B. The Implementation Of The Project.

Develop environment: Our project is using the following language and frameworks:

1) Back-end: Python and Flask;
2) Front-end: React and D3.js;
3) Database: MySQL.

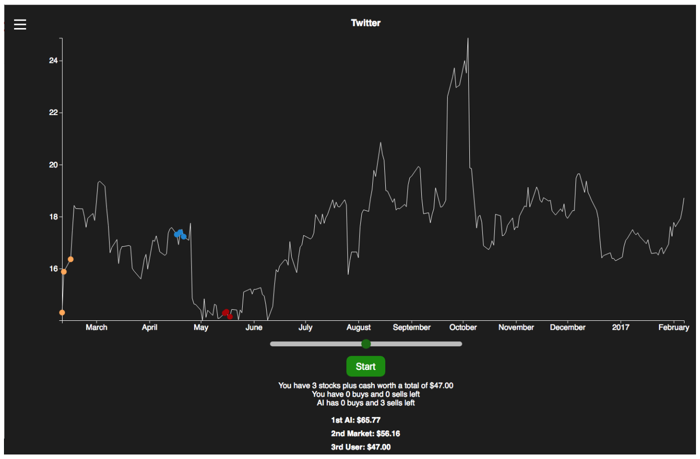
The data source is stock market price, in daily basis. The data sources are in JSON format.

- Sample small data snippet
  - Sample data of stock market: Figure 2
- Sample small output
  - Sample output of stock market: Here we plot ”close” data against date to get a stock price, show in Figure 3.



```
{
    "Meta Data": {
        "1. Information": "Daily Prices (open, high, low, close) and Volumes",
        "2. Symbol": "MSFT",
        "3. Last Refreshed": "2018-04-03",
        "4. Output Size": "Full size",
        "5. Time Zone": "US/Eastern"
    },
    "Time Series (Daily)": {
        "2018-04-03": {
            "1. open": "89.5750",
            "2. high": "90.0500",
            "3. low": "87.8900",
            "4. close": "89.7100",
            "5. volume": "37213837"
        },
        "2018-04-02": {
            "1. open": "90.4700",
            "2. high": "90.8800",
            "3. low": "87.5100",
            "4. close": "88.5200",
            "5. volume": "48424595"
        },
        "2018-03-29": {
            "1. open": "90.1800",
            "2. high": "92.2900",
            "3. low": "88.4000",
            "4. close": "91.2700",
            "5. volume": "45867548"
        },
        "2018-03-28": {
            "1. open": "89.8200",
            "2. high": "91.2300",
            "3. low": "88.8730",
            "4. close": "89.3900",
            "5. volume": "52501146"
        },
        "2018-03-27": {
            "1. open": "94.9400",
            "2. high": "95.1390",
            "3. low": "88.5100",
            "4. close": "89.4700",
            "5. volume": "53704562"
        },
```

Fig. 2. Sample stock market data

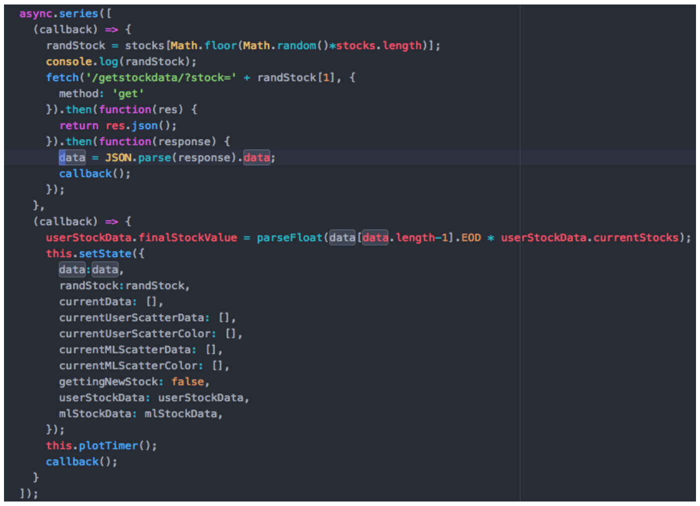


You have 3 stocks plus cash worth a total of $47.00
You have 0 buys and 0 sells left
AI has 0 buys and 3 sells left

1st AI: $65.77
2nd Market: $56.16
3rd User: $47.00

Fig. 3. Sample stock market output graph

- Working code
  - Back-end: sending Http requests to APIs and retrieving JSON files, parse JSON files and put into memory, as shown in Figure 4.
  - Front-end: getting data from back-end and do visualization on web page, using D3.js and React, shown in Figure 5.
- Demo and sample findings
  - This novelty of this project is to provide a fun game experience to users who tend to analyze the historical trend of one stock and try to gain a better insight of what happened behind human vs AI in financial industry in real life. To make it interesting, we make animations to simulate data streaming, and making UI good looking and user-friendly. // Thus the most interesting and also the most challenging part in this project finding a suitable ML algorithm to play as AI role. A suitable algorithm does not only mean it is accurate and yield reasonable results, but also it should be less expensive and can calculate fast enough in order to maintain a smooth game experience. Previously we have tried some sophisticated model, but we found that such models are expensive, meaning it will hinder the game experience and yield intermediate results. Therefore, we decided to use an simple Linear Regression Model, which is faster and still has reasonable results.



```
async.series([
  (callback) => {
    randStock = stocks[Math.floor(Math.random()*stocks.length)];
    console.log(randStock);
    fetch('/getstockdata/?stock=' + randStock[1], {
      method: 'get'
    }).then(function(res) {
      return res.json();
    }).then(function(response) {
      data = JSON.parse(response).data;
      callback();
    });
  },
  (callback) => {
    userStockData.finalStockValue = parseFloat(data[data.length-1].EOD * userStockData.currentStocks);
    this.setState({
      data:data,
      randStock:randStock,
      currentData: [],
      currentUserScatterData: [],
      currentUserScatterColor: [],
      currentMLScatterData: [],
      currentMLScatterColor: [],
      gettingNewStock: false,
      userStockData: userStockData,
      mlStockData: mlStockData,
    });
    this.plotTimer();
    callback();
  }
]);
```

Fig. 4. Sample code of back-end

C. User Interface.

In this project, the main purpose is to provide users a simple interface, where he/she can easily logging in and start enjoying the game.
The user interface of this project is divided into three major areas - Login/Sign up pages, Start pages and Game page.

1) Login/Signup Page: User will first see the welcome page of stock stories, from where user has to login or signup. In login page shown in Figure 6.
2) Start Pages: After successfully logging in, user will see the start page of the game describing the game rules and goals, as shown in Figure 7.
3) Game Page: In game page the stock data is shown dynamically with expanding of the x axis. This is trying to simulate the real-life scenario that stock price is renewed along with the time, and that the future is unknown. User must make decisions based on the past data (currently presented line path) to decide whether to buy, to sell or to rest. Two bottoms, ”buy” and ”sell”, are provided, as well as a slider to adjust the speed of animation. At the bottom of the page it shows the

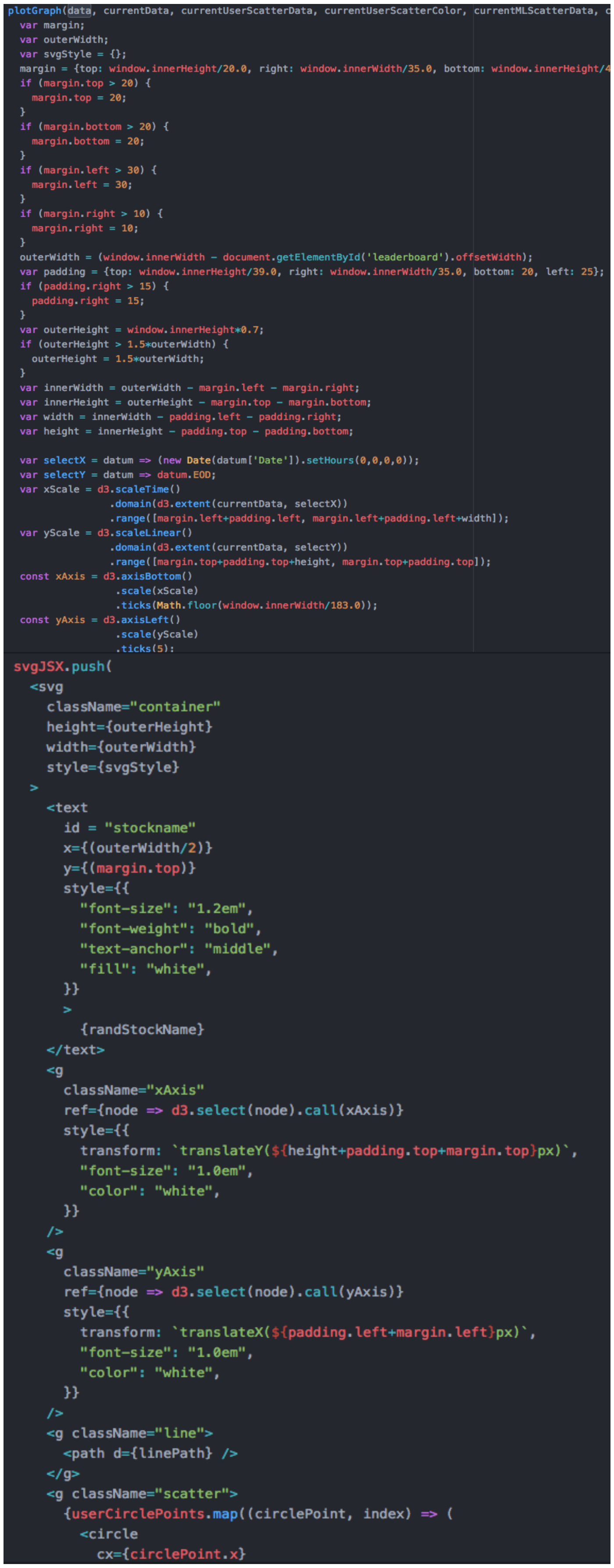
summation value of stock and cash of the user, and how many buys and sells left for the user and AI. The page is shown in Figure 8.

When game is over, a summery of this round is shown at the bottom of the page. If the user win this round, a green text ”Winner Winner Chicken Diner!” is shown, otherwise it is a red text ”Better Luck Next Time. Please Try Again!”. Also each buy and sell price of both user and AI are shown and calculated. The ranking list of User, AI and Market is also shown sorted by the total profit. On the left, a score ranking records the cumulative scores of all the rounds played so far. User can see his/her total score after playing multiple round and evaluate his/her stock trading skills. Such page is shown in Figure 9.

Since Stock Trade: Man vs AI is an game and simulation application, and no data should be manipulated or modified. Users are not allowed to update or delete any information related to application. Users can only modify his/her profile. Each user interface event of mouse click by user will result in updating the current game statues and game results.

II. Reference

1) IH Witten, E Frank, MA Hall, CJ Pal, Data Mining: Practical machine learning tools and techniques, Fourth Edition, Elsevier, 2017.
2) Daniel A. Keim, Florian Mansmann, Jorn Schneidewind, and Hartmut Ziegler (2006). ”Challenges in Visual Data Analysis”. Tenth International Conference on Information Visualisation (IV’06). pp. 9-16.

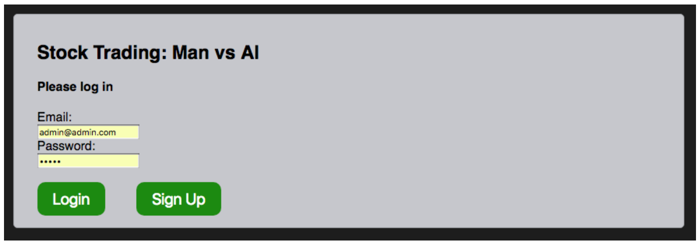


```
plotGraph(data, currentData, currentUserScatterData, currentUserScatterColor, currentMLScatterData, c
  var margin;
  var outerWidth;
  var svgStyle = {};
  margin = {top: window.innerHeight/20.0, right: window.innerWidth/35.0, bottom: window.innerHeight/4
  if (margin.top > 20) {
    margin.top = 20;
  }
  if (margin.bottom > 20) {
    margin.bottom = 20;
  }
  if (margin.left > 30) {
    margin.left = 30;
  }
  if (margin.right > 10) {
    margin.right = 10;
  }
  outerWidth = (window.innerWidth - document.getElementById('leaderboard').offsetWidth);
  var padding = {top: window.innerHeight/39.0, right: window.innerWidth/35.0, bottom: 20, left: 25};
  if (padding.right > 15) {
    padding.right = 15;
  }
  var outerHeight = window.innerHeight*0.7;
  if (outerHeight > 1.5*outerWidth) {
    outerHeight = 1.5*outerWidth;
  }
  var innerWidth = outerWidth - margin.left - margin.right;
  var innerHeight = outerHeight - margin.top - margin.bottom;
  var width = innerWidth - padding.left - padding.right;
  var height = innerHeight - padding.top - padding.bottom;

  var selectX = datum => (new Date(datum['Date']).setHours(0,0,0,0));
  var selectY = datum => datum.EOD;
  var xScale = d3.scaleTime()
                 .domain(d3.extent(currentData, selectX))
                 .range([margin.left+padding.left, margin.left+padding.left+width]);
  var yScale = d3.scaleLinear()
                 .domain(d3.extent(currentData, selectY))
                 .range([margin.top+padding.top+height, margin.top+padding.top]);
  const xAxis = d3.axisBottom()
                  .scale(xScale)
                  .ticks(Math.floor(window.innerWidth/183.0));
  const yAxis = d3.axisLeft()
                  .scale(yScale)
                  .ticks(5);
```

```
svgJSX.push(
  <svg
    className="container"
    height={outerHeight}
    width={outerWidth}
    style={svgStyle}
  >
    <text
      id = "stockname"
      x={(outerWidth/2)}
      y={(margin.top)}
      style={{
        "font-size": "1.2em",
        "font-weight": "bold",
        "text-anchor": "middle",
        "fill": "white",
      }}
      >
        {randStockName}
    </text>
    <g
      className="xAxis"
      ref={node => d3.select(node).call(xAxis)}
      style={{
        transform: `translateY(${height+padding.top+margin.top}px)`,
        "font-size": "1.0em",
        "color": "white",
      }}
    />
    <g
      className="yAxis"
      ref={node => d3.select(node).call(yAxis)}
      style={{
        transform: `translateX(${padding.left+margin.left}px)`,
        "font-size": "1.0em",
        "color": "white",
      }}
    />
    <g className="line">
      <path d={linePath} />
    </g>
    <g className="scatter">
      {userCirclePoints.map((circlePoint, index) => (
        <circle
          cx={circlePoint.x}
```

Fig. 5. Sample code of front-end



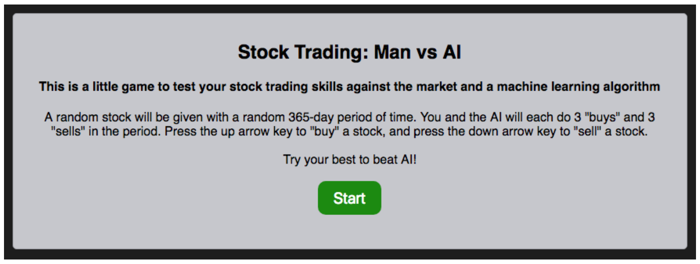
Stock Trading: Man vs AI

Please log in

Email: admin@admin.com

Password: •••••

Fig. 6. Login page



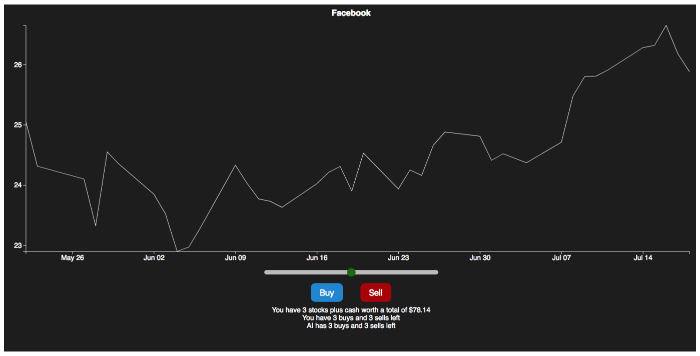
Stock Trading: Man vs AI

This is a little game to test your stock trading skills against the market and a machine learning algorithm

A random stock will be given with a random 365-day period of time. You and the AI will each do 3 "buys" and 3 "sells" in the period. Press the up arrow key to "buy" a stock, and press the down arrow key to "sell" a stock.

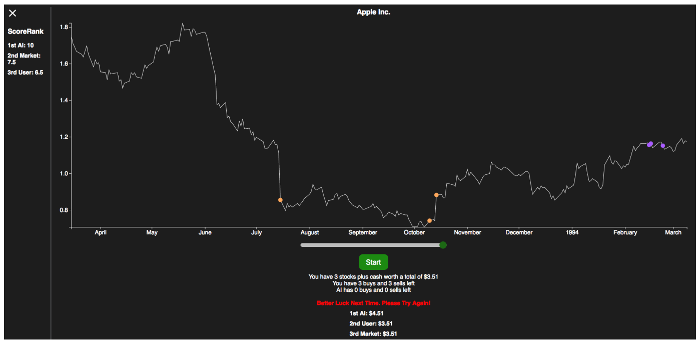
Try your best to beat AI!

Fig. 7. Start page



You have 3 stocks plus cash worth a total of $78.14
You have 3 buys and 3 sells left
AI has 3 buys and 3 sells left

Fig. 8. Game page



ScoreRank
1st AI: 10
2nd Market: 7.5
3rd User: 6.5

You have 3 stocks plus cash worth a total of $3.51
You have 3 buys and 3 sells left
AI has 0 buys and 0 sells left

Better Luck Next Time. Please Try Again!
1st AI: $4.51
2nd User: $3.51
3rd Market: $3.51

Fig. 9. Game Over page