Project for software engineering

members

name	roll number
ishant kumar	185501
moulik bhardwaj	185506
vishal guleria	185511
shivang upadhyay	185515

Aim

to simulate the stock market price index and to prediction model

Approch

via this project we tried to predict the next stock index based on simulated interinsic parameters. we chose the following randomized parameters

- buyer and seller interinsic value multiplier
  - which includes how much it can differ from the current price while placing price order in the next iteration.
- buyer and seller euqilibrium
  - which windows our allover random distribution to a positive or negative side.
  - implementation example

  ```
  s.currentPrice + buyerMultiplier *(rand.Float64() - equi)
  ```

- traders count
  - number of people we are considering for simulation
- trading policy
  - policy which governs the conditions for a successful trade.

source for trading policy

```
ratio1 = self.quantity/User.totalStock
ratio2 = 1/User.totalUsers
self.equilibrium = _normalize(ratio1/ratio2, 0, User.totalUsers
maxVal = User.model.getCurrentPrice() + self.buyerMultiplier*(random()-self.equilibrium)-1000
        if maxVal >= price or random()<self.randomProbability:

            boughtAmount = min(int(self.money/price) ,quantity)
            self.quantity += boughtAmount
            self.money -= boughtAmount*price
            seller.money += boughtAmount*price
```

we are running the simulation with different parameters to get a single predicted value and feeding it back to out prediction model.

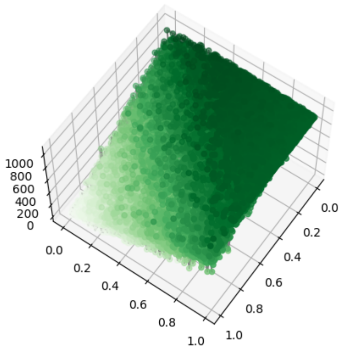
```
        ┌─────────────────────┐
     ┌──│ simulate many times │<───────┐
     │  └─────────────────────┘        │
     │  ┌────────────────────────────┐ │
     └─>│ pretrained prediction model │─┘
        └────────────────────────────┘
```

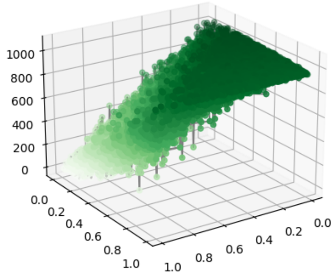
source for our main event loop

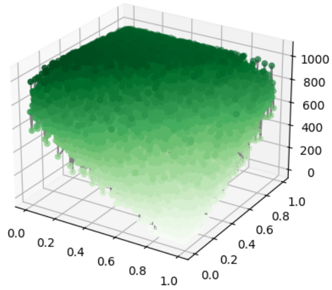
```
for i in range(50):
  newClose = Server.simulate(userList)
  User.model.update(value=newClose)
  newData.append(newClose
```

Simulation parameters experiments

we tried the simulating the full range of buyer and seller equilibrium (0 - 1) to the a random dataset. and plotted relation between the `buyer equilibrium` , `seller equilibrium` and `traded volume`.







source

```
for beq := 0.0; beq < 1; beq += 0.01 {
    for seq := 0.0; seq < 1; seq += 0.01 {
        s.volume = 0
        s.multiSellers(seq)
        s.multiBuyers(beq)
        log(beq, seq, s.volume)
    }
}
```

Dataset

here is a standard stock index data schema

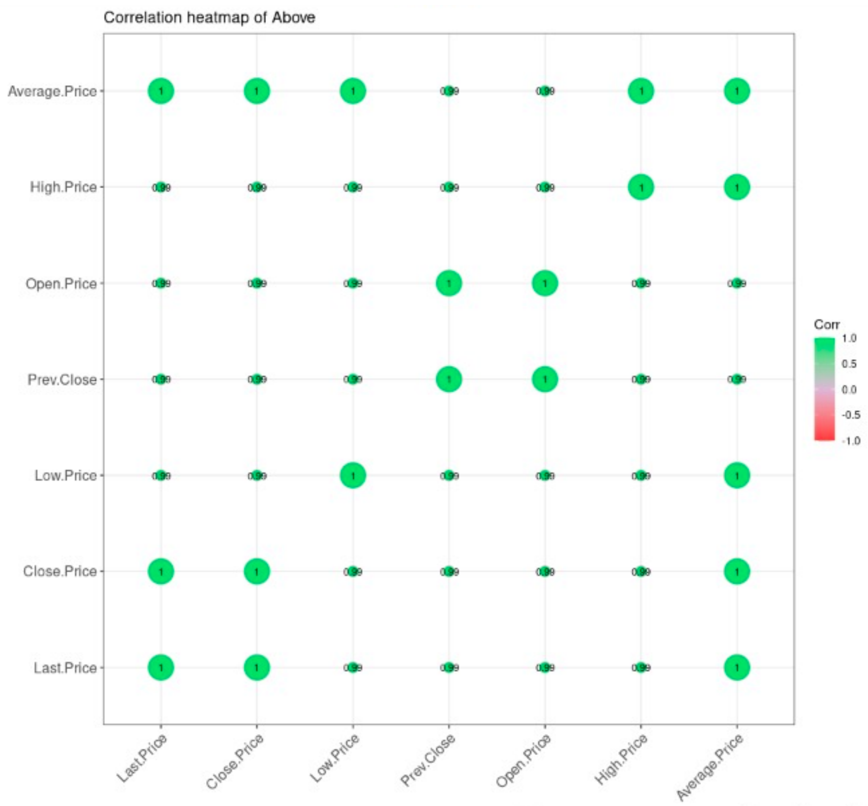
date	close	high	low	open	volume	adjClose	adjHigh	adjLow	adjOpen	adjVolume	divCash
2016-04-27	00:00:00+00:00	97.82	98.71	95.68	96.000	114602142	22.741853	22.948766	22.244331	22.318727	458408568
2016-04-28	00:00:00+00:00	94.83	97.88	94.25	97.610	82242690	22.046717	22.755802	21.911875	22.693030	328970760
2016-04-29	00:00:00+00:00	93.74	94.72	92.51	93.990	68531478	21.793307	22.021144	21.507348	21.851428	274125912
2016-05-02	00:00:00+00:00	93.64	94.08	92.40	93.965	48160104	21.770058	21.872352	21.481774	21.845616	192640416

here for example this stock index for apple `APPL`. and data is taken by tiingo api for pandas datareader.

feature selection

following table discribes the corelation between the fields for the data fields above

cov()	Prev.Close	Open.Price	High.Price	Low.Price	Last.Price	Close.Price	Average.Price
Prev.Close	1.0000000	0.9977647	0.9925048	0.9923783	0.9850994	0.9856923	0.9927074
Open.Price	0.9977647	1.0000000	0.9937166	0.9932381	0.9860046	0.9865810	0.9938009
High.Price	0.9925048	0.9937166	1.0000000	0.9925937	0.9946247	0.9949331	0.9981933
Low.Price	0.9923783	0.9932381	0.9925937	1.0000000	0.9946378	0.9949878	0.9973750
Last.Price	0.9850994	0.9860046	0.9946247	0.9946378	1.0000000	0.9998530	0.9971622
Close.Price	0.9856923	0.9865810	0.9949331	0.9949878	0.9998530	1.0000000	0.9975444
Average.Price	0.9927074	0.9938009	0.9981933	0.9973750	0.9971622	0.9975444	1.0000000

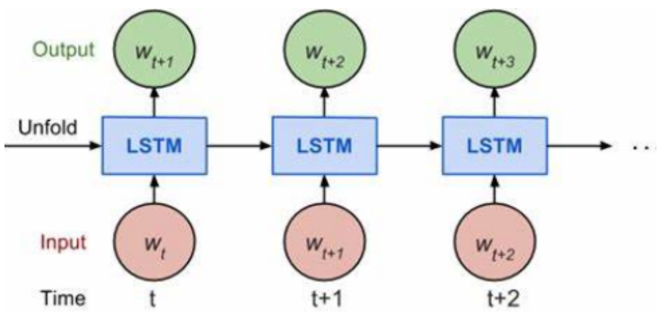


for getting the best correlation with the average we chose only to work the `Close Price` value.

Model

we preferred using recurrent neural cells for the given time series. based on this discription of lstm on wikipedia :

> LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the vanishing gradient problem that can be encountered when training traditional RNNs. Relative insensitivity to gap length is an advantage of LSTM over RNNs, hidden Markov models and other sequence learning methods in numerous applications.



and we used 50 sells in first 3 layers based on our try and error approch. also we modelled out input data with 30 values in a row, for simulating time series behaviour.

```
| 50 x lstm | ---> | 50 x lstm | ---> | 50 x lstm | --> | 1 x dense |
```

source

```
m = Sequential([
    LSTM(50, return_sequences=True, input_shape=(30, 1)),
    LSTM(50, return_sequences=True),
    LSTM(50),
    Dense(1),
]
m.compile(
  loss='mean_squared_error',
  optimizer='adam'
)
```

model summary

```
Model: "sequential_22"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
lstm_39 (LSTM)               (None, 30, 50)            10400
_________________________________________________________________
lstm_40 (LSTM)               (None, 30, 50)            20200
_________________________________________________________________
lstm_41 (LSTM)               (None, 50)                20200
_________________________________________________________________
dense_28 (Dense)             (None, 1)                 51
=================================================================
Total params: 50,851
Trainable params: 50,851
Non-trainable params: 0
_________________________________________________________________
```

Development model

prototype model

- with many interations we chose different approches for out prediction model and simulation techniques.
- while analysis period we prepared some temporary simulation results, for weekly assesment.

Simulation result

by running the model given above on the apple `APPL` stock index we got the following results. orange values shows the values that are pretrained, and blue values are the one that are predicted.