# DQN MODEL

## Train Data:

* Trained the model on stock of google from 2015 to 2019

## Agent Class:

* Requires Money,Maximum Transactions allowed , Window size( data of how many previous days you want to use for prediction ) and model name just to load any previously trained model
* Instance Attributes-
  + State\_size = window\_size+2(money and transition) #input shape
  + Memory = stores state , next\_state , reward and done
  + money = money in hand at any moment
  + Is\_eval = true if using the model for testing
  + epsilon = for epsilon greedy approach , initially 1
  + epsilon\_min = minimum value of epsilon
  + Gamma = discount factor
  + Max\_t = maximum transaction allowed
  + Epsilon\_decay = rate of epsilon decay
  + Action\_size = 3 , buy sell and hold
  + Inventory = stock in hand , initially zero
  + Transactions = no. of transactions done
  + Intial\_money = Money
  + model = model used for predicting Q values
  + money\_before
* Class Attributes-
  + \_model = creating the model that will be used for predicting q values , takes state as input and returns Q value for every action.
  + Act = gets state and returns the appropriate action to be taken by the agent
  + expReplay = stores data of previous states,actions ,rewards in a batch and use them to get Q values for the state,action pair using bellman equation , model is also trained inside the this function

## Functions

* Buy
  + Buy stocks if transactions are less than max\_t
  + return -1 if transactions = max\_t , to stop unnecessary buying calls
  + return 0 otherwise
* Sell
  + Sell all the stocks present in inventory
  + return -1 if we have no stock to sell , to stop unnecessary selling calls
  + return max(0,profit between to consecutive sell call ) otherwise
* Get\_state
  + takes agent and data of previous days and return us state of the env
  + Standardise the closing price so that model can be used for any stock
  + Take sigmoid of the normalised values (model just performed better this way)
  + Add current money and tractions no. to the state and return it.
* Sigmoid : take x , return sigmoid(x)
* formatPrice : print price

## Training

Initially I started with reward equals to profit or zero , but this resulted in constant sell call by the agent . Then I started giving negative reward for unnecessary sell call and increased it exponentially with consecutive unnecessary calls. I also added negative reward for consecutive hold and unnecessary buy calls.

## Testing

Tested on 4 companies for the previous year , got an average profit of 28%