A Project/Dissertation Review-1 Report on

"STOCK PRICE PREDICTION USING QUANTUM"

Submitted in partial fulfilment of the
requirement for the award of the degree of

B. TECH

GALGOTIAS UNIVERSITY
(Established under Galgotias University Uttar Pradesh Act No. 14 of 2011)

Under The Supervision
of Mr. Jaya Kumar

Submitted By

Ritik Varshney(18SCSE1010067)
Ritik Sharma(18SCSE1010180)

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA
INDIA
OCTOBER, 2021

SCHOOL OF COMPUTING SCIENCE AND ENGINEERING
GALGOTIAS UNIVERSITY, GREATER NOIDA

CANDIDATE'S DECLARATION

I/We hereby certify that the work which is being presented in the thesis/project/dissertation, entitled **"**STOCK PRICE PREDICTION USING QUANTUM**"** in partial fulfilment of the requirements for the award of the B.Tech submitted in the School of Computing Science and Engineering of Galgotias University, Greater Noida, is an original work carried out during the period of month, Year to Month and Year, under the supervision of Dr. Jaya Kumar Designation, Department of Computer Science and Engineering/Computer Application and Information and Science, of School of Computing Science and Engineering , Galgotias University, Greater Noida

The matter presented in the thesis/project/dissertation has not been submitted by me/us for the award of any other degree of this or any other places.

Ritik Sharma 18SCSE1010180

Ritik Varshney 18SCSE1010067

This is to certify that the above statement made by the candidates is correct to the best of my knowledge.

Dr. Jaya Kumar

Professor

CERTIFICATE

The Final Thesis/Project/ Dissertation Viva-Voice examination of Ritik Sharma 18SCSE1010180 and Ritik Varshney 18SCSE1010067 has been held on /12/2021 and his work is recommended for the award of Bachelor of Technology (CSE).

Signature of Examiner(s) **Signature of Supervisor(s)**

Signature of Project Coordinator **Signature of Dean**

Date: Dec 21

Place: Greater Noida

Abstract

Stock market prediction aims to determine the future movement of the stock value of a financial exchange. The accurate prediction of share price movement will lead to more profit investors can make.
Existing Solutions were Money related trade judgment making is a strengthening and difficult errand of fiscal data guess. Figure about securities trade with high exactness improvement return advantage for examiners of the stocks. In perspective on the snare of budgetary trade financial data, expansion of productive models for forecast conclusion is very difficult, and it must be precise. This consider attempted to make models for guess of the securities trade and to pick whether to buy/hold the stock using data mining and AI techniques.
The advantage of predicting the stock market is it helps you to invest wisely to make good profits.

Table of Contents

Title

1. Abstract
2. Introduction
3. Problem definition
4. What is Financial instrument?
5. Option Contract
6. Pricing Option
7. Hybrid Quantum Inspired neural network
8. Learning in Quantum neural network
9. Algorithm
10. Tools used for implementation
11. Literature Survey
12. Monte Carlo Method
13. Support Vector Machine (SVM)
14. Diagram Of SVM (Linear and non-linear)
15. Data flow Diagram
16. Source Code
17. Result and Discussion
18. Conclusion
19. References

List of Diagram

S.No.	Title	Page No.
1.	Linear Support Vector Machine	25
2.	Non- Linear Support Vector Machine	26
3.	Data Flow Diagram	27
4.	Day-Price Graph (Output)	29
5.	Price-Frequency Graph (Output)	30
6.	Option Payoff value-price Graph (Output)	31
7.	Probability Graph (Output)	33

Introduction

STOCK price prediction has been at focus for years since it canyield significant profits. Predicting the stock market is not a simple task, mainly as a consequence of the close to random-walk behavior of a stock time series. Fundamental and technical analyses were the first two methods used to forecast stock prices. Artificial Neural networks (ANNs) is the most commonly used technique. Stock market is characterized as dynamic, unpredictable and non- linear in nature. Predicting stock prices is a challenging task as it depends on various factors including but not limited to political conditions, global economy, company's financial reports and performance etc.

"Thus, to maximize the profit and minimize the losses, techniques to predict values of the stock in advance by analyzing the trend over the last few years, could prove to be highly useful for making stock market movements. Traditionally, two main approaches have been proposed for predicting the stock price of anorganization. Technical analysis method uses historical price of stocks like closing and opening price, volume traded, adjacent close values etc. of the stock for predicting the future price of the stock.

The second type of analysis is qualitative, which is performed on the basis of external factors like company profile, market situation, political and economic factors, textual information in the form of financial new articles, social media and even blogs by economic analyst. Now a days, advanced intelligent techniques based on either technical or fundamental analysis are used for predicting stock prices.

Particularly, for stock market analysis, the data size is huge and also non-linear. To deal with this variety of data efficient model is needed that can identify the hidden patterns and complex relations in this large data set. Machine learning techniques in this area have proved to improve efficiencies by 60-86 percent as compared to the past method Quantum computers has inbuilt parallelism and large memories. Visualizing these potential for quantum computers, it is considered that the QNN shall be the next natural step in the evolution of

neuro-computing system. Ezonov and Ventura have introduced the possibilities of combining the unique computational capabilities of CNN and quantum computation. This combination can produce a computational paradigm of incredible potential. Xiao and Cao have proposed a QNN model based on quantum and classical neurons. Miszczak has proposed ways for preparing initial quantum state based on probabilities. Since last two decades, a great attention is visualized in the efforts of researcher on the idea of time series prediction. They have studied these fluctuations through the use of CNN. Various prediction strategies can be applied for improving accuracy of prediction. A classical feed forward back propagation neural network has been successively applied and tested in this paper to the time series of one previous prediction using multiple stocks data using sliding window.

PROBLEM DEFINITION

The Stock market check is an exceptionally fascinating errand which joins high substances of how the budgetary exchange limits, and what unconventionalities can be prompted in a market in light of different conditions. While a few venders may battle that the market itself is functional, and that if there is new check or any assortment from the standard in a market it charms it by auditing itself, thusly making no space for conjectures, while several vendors may battle that on the off chance that the information is orchestrated well, by then machine can make a sort out of procedure that is persuading can affect high continue exchanging or HFT, which is just conceivable through Algorithmic Trading Systems or Automated Systems of Trade. Money related authorities think about the expression, Buy low, Move high yet this does not give enough setting to settle on proper endeavor decisions. Before an investigator places assets into any stock, He should realize how money markets continues .Setting assets into a wonderful stock regardless at a horrible time can have awful results, while vitality for a common stock at the fortunate time can hold up under focal points. Cash related monetary pros of today are going toward this issue of trading as they don't for the most part understand concerning which stocks to buy or which stocks to offer with the authentic objective to get impeccable focal points. Envisioning whole game plan estimation of the stock is commonly clear than foreseeing on day-to-day premise as the stocks change rapidly reliably subject to world events. The answer for this issue requests the utilization of instruments and advances identified with the field of information mining, design acknowledgment, machine learning and information forecast. The application will foresee the stock costs for the following exchanging day. The necessities and the usefulness of this application corresponds it to the class.

What is a financial instrument?

A financial instrument is any asset that can be traded between various parties. For example, shares of a company are equity instruments, and debt, cash and even contracts can be considered as financial instruments.

In general, we can divide financial instruments into several categories, such as cash instruments, asset or debt-based instruments and derivative instruments. In this challenge, we will focus on the last category, namely, derivatives. These instruments get their name from the fact that their price is derived based on the price of a separate underlying quantity. In particular, we will consider derivatives called European options.

Option contracts

Options are financial derivatives that are defined explicitly by contracts. These contracts give the buyer of the option the right, but not the obligation, to buy or sell a specific underlying asset at an agreed-upon price sometime in the future.

- Options that give the buyer the right to buy the underlying asset are called call options • Options that give the buyer the right to sell the underlying asset are called put options.

in both cases, the price to buy or sell the asset at, is agreed upon in the contract and called the strike price.

For example, let's assume that the share price of a company called ABC is currently trading at So-50 ZAR You believe that in 1 months time, the share price

will double to 100 ZAR. You can either buy the share now for 50 ZAR, or you could buy a 1-month call option for a much cheaper price of 5 ZAR. If the price of the ABC share does indeed double in 1 months time, then you can exercise your option right to buy the ABC share at the agreed up strike price (which will be lower than the actual share price). This may sound a little tricky, so let's make it concrete by going through an example with various scenarios we can encounter.

Recall that the current price of ABC is S50 ZAR. A 1-month call option with a strike price of K 80 ZAR is available purchase for Pall - 5 ZAR. Let's consider the scenarios of buying an ABC share today or the call option.

Today:

Scenario 1: Buying a share of ABC today Investment S=50 ZAR Scenario 2: Buying a 1-month call option on ABC Investment = Pall - 5 ZAR

In 1-months time:

Since option contracts are valid for a pre-determined period of time, their value at the expiration date is called the payoff that you will recieve. Technically, the price of an ABC share could be trading at any value greater than or equal to 0 ZAR. Let's imagine that the price after 1-month S, can be 40 ZAR, 60 ZAR or 100 ZAR and look at the payoffs for each scenario.

• S= 40 ZAR Scenario 1 Since you own the share of ABC and purchased it at 50 ZAR: you will lose 10 ZAR Le. the payoff will be a loss of -10 ZAR. Scenario 2: The strike price is higher than the actual share price. Thus, you would not execute the option to buy ABC at K80 ZAR when you can purchase the share on the market at 40 ZAR. You will lose out on the price you paid for the call option and so the payoff will be a loss of -5 ZAR.

• 60= ZAR Scenario 1: The payoff will be the profit you made from buying the share at 50 ZAR and it's current price. Le payoff 60-50-10 ZAR. Scenaro 2. The strike price is still higher than the share price. Thus, you would not execute the option and lose 5

• 100= ZAR Scenario 1: payoff = 100-50=50 ZAR 1 Scenario 2: Here, the strike price is lower than the market price so it makes sense to execute the option. By doing so, you can buy the share at the strike price of 80 ZAR and the profit is therefore a = 100 - 80 = 20ZAF . However, remember that you also paid a premium for the option so the total payoff is 100 - 80 - 5 = 15ZAF .

At this stage, you might be wondering why buy the option at all? And it kind of seems like betting? What's important to keep in mind is the initial investment you have to put up in order to buy the actual ABC share vs buying the call option. Buying the share requires a much higher investment and you run the risk of losing a lot more money if the share price drops below the price you initially paid for it. Whereas with the option, if the share price drops below the strike price, you can just let the option expire and lose a maximum of 5 ZAR (the price you paid for the option). There are also other reasons to purchase options, such as hedging against risk or offsetting other trades. In general, you can read more about these strategies here: https://www.investopedia.com/trading/options-strategies/ (https://www.investopedia.com/trading/options-strategies/)

Pricing options

Now, how on earth do we begin to price these option contracts? Previously, we said the price of a 1-month call option linked to the ABC share price was 5 ZAR. But is that price fair? And how would be go about determining the fair price of these options in general?

If we think about what should affect the fair price of an option, it boils down to understanding the what the price of the underlying asset it is linked to will be in the future. Due to the random (also called stochastic) nature of most of the parameters that go into pricing the underlying assets that options are defined on,

calculating the fair price can be a difficult task, and whilst analytical models exist for the basic types of options, the simplifying assumptions required for these models often limit their applicability. Thus, numerical methods that estimate the fair price have to be employed for option pricing, with Monte Carlo being one of the most popular.

HYBRID QUANTUM INSPIRED NEURAL NETWORK

QNN is based on the techniques of quantum computation. Qubit is defined as the smallest unit of information in quantum computation which is a probabilistic representation. A qubit may either be in the “1” or “0” or in any superposition of the two[9].

Figure 1 and 2 represent the concept of hybrid quantum neural network.

The state of the qubit can be represented as:

……..(i)

Where a and are the numbers that indicate the amplitude of the corresponding states such that:

|a|2+|þ|2 =1

A qubit is defined as smallest unit of information in quantum computation. It is defined as a pair of numbers .

Angle , in a more geometrical aspects is defined such that

A state of qubit can be updated by applying the above quantum gate. Application of rotation gate on a qubit can be Quantum gates may be applied for modifying the probabilities as a result of weight updating,. One such rotation gate can be :

……..(ii)

represented as:

= (iii)

The following hybrid quantum inspired neural network is proposed for the stock's price prediction.

- Initialize

a. Quantum hidden neuron:

Start from state , prepare the superposition :

The QNN is based on the techniques of quantum computation. Qubit is defined as the smallest unit of information in quantum computation which is a probabilistic representation. A qubit may either be in the "1" or "0" or in any superposition of the two[9].

Figure 1 and 2 represent the concept of hybrid quantum neural network.

The state of the qubit can be represented as:

$$|\psi\rangle = \alpha\,|0\rangle + \beta\,|1\rangle \quad \text{(i)}$$

Where a and β are the numbers that indicate the amplitude of the corresponding states such that:

$$|a|^2+|b|^2=1$$

A qubit is defined as smallest unit of information in quantum computation. It is defined as a pair of numbers $\begin{bmatrix}\alpha\\ \beta\end{bmatrix}$.

Angle θ in a more geometrical aspects is defined such that

$$\cos(\theta) = |\alpha| \quad and \quad \sin(\theta) = |\beta| \;:$$

Quantum gates may be applied for modifying the probabilities as a result of weight updating. One such rotation gate can be :

$$U(\Delta\theta) = \begin{bmatrix}\cos(\Delta\theta) & -\sin(\Delta\theta)\\ \sin(\Delta\theta) & \cos(\Delta\theta)\end{bmatrix} \quad \text{(ii)}$$

A state of qubit can be updated by applying the above quantum gate. Application of rotation gate on a qubit can be represented as:

$$\begin{bmatrix}\alpha'\\ \beta'\end{bmatrix} = \begin{bmatrix}\cos(\Delta\theta) & -\sin(\Delta\theta)\\ \sin(\Delta\theta) & \cos(\Delta\theta)\end{bmatrix}\begin{bmatrix}\alpha\\ \beta\end{bmatrix} \quad \text{(iii)}$$

The following hybrid quantum inspired neural network is proposed for the stock's price prediction.

- Initialize

  a. Quantum hidden neuron:

  Start from state $|0\rangle$, prepare the superposition :

  $\sqrt{p}\,|0\rangle + \sqrt{1-p}\,|1\rangle$ with $0 \leq |p| \leq 1$;

  Where p represents random probability of initializing the system in the state $|0\rangle$.

The desired state can be reached by using rotation gate R:

$$R(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix};$$

$$\tan(0) = \sqrt{\frac{p}{1-p}} \quad ;$$

$$0 = \arctan\sqrt{\frac{p}{1-p}} \quad ;$$

$$R(\epsilon) = \begin{bmatrix} \sqrt{1-p} & -\sqrt{p} \\ \sqrt{p} & \sqrt{1-p} \end{bmatrix}.$$

$$\begin{bmatrix} \alpha \\ \beta \end{bmatrix} = \begin{bmatrix} \sqrt{1-p} & -\sqrt{p} \\ \sqrt{p} & \sqrt{1-p} \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix}$$

b. Classical neurons:

Initialize classical neurons by random number generation.

- Output from quantum neuron:

$$v_j = f(\sum_{i=1}^{n} |\psi_{ji}|^* x_i) \quad \text{.....(iv)}$$

Where f is a problem dependent sigmoid or Gaussian function

- Output from the network:

$$y_k = f(\sum_{j=1}^{l} w_{jk} * v_j) \quad \text{..... (v)}$$

III. LEARNING IN QUANTUM NEURAL NETWORK

The learning follows the rules of feed forward back propagation algorithm

- Updation of output layer weight

  $\Delta w_{jk} = \eta e_k f' v_j$ .....(vii)

- Updation of quantum hidden layer weight

  in quantum BP algorithm the weights are updated by quantum gate according to equation (iii), in this case the equation shall be

$$\begin{bmatrix} \alpha_{ij}' \\ \beta_{ij}' \end{bmatrix} = \begin{bmatrix} \cos(\Delta\theta_{ij}) & -\sin(\Delta\theta_{ij}) \\ \sin(\Delta\theta_{ij}) & \cos(\Delta\theta_{ij}) \end{bmatrix} \begin{bmatrix} \alpha_{ij} \\ \beta_{ij} \end{bmatrix} \text{..(viii)}$$

where $\Delta\theta_{ij} = -\frac{\partial E}{\partial \theta_{ij}}$;

$= -\frac{\partial E}{\partial y_k}\frac{\partial y_k}{\partial v_j}\frac{\partial v_j}{\partial \theta_{ij}}$ by chain rule

$= -E_k f' w_{jk} v_j x_i (\cos(\gamma_{ij}) - \sin(\gamma_{ij}))$

where γ_{ij} is a phase of $|\psi_{ij}\rangle$ such that

$$|\psi_{ij}\rangle = \begin{bmatrix} \cos(\gamma_{ij}) \\ \sin(\gamma_{ij}) \end{bmatrix} ;$$

for γ_{ij} updation shall be:

$\gamma_{ij}' = \gamma_{ij} + \eta\Delta\theta_{ij}$; η is learning rate

IV. ALGORITHM: QUANTUM INSPIRED HYBRID BACK PROPAGATION

Start with randomly chosen weights for classical output neurons and initialize quantum hidden neuron

While mean squared error is unsatisfactory and computational bounds not exceeded

Do

For each input array $x_1, x_2, \ldots, x_m$;

Compute hidden node output

Compute the network output

Compute the output error

Modify weights between hidden and output node

Apply quantum gate and modify hidden node weights

End For

End Do

End while

End

The succession of values in a time series is usually influenced by a number of external information. When this information are not available only past value of the series itself can be used to build a prediction model.

Xt+1= f(x1, ,xn); where xt+1 is estimated next value based

on current and past values of x.

Efficient market hypothesis (EMH) emphasizes that if statistically significant serial dependencies exist within time series of stock prices, the community of business analyst will immediately exploit it. Stock price changes can therefore be only be explained by arrival of new information, which by definition cannot be forecasted. EMH is only true when linear models are applied. It has been generally established that by use of non-linear models like neural networks, the results challenge the EMH hypothesis.

The data used for stock price prediction are monthly closing prices of 10 stocks. A total of 7 years (2003-2010) daily data were considered [13]. The figure 3(3.1 to 3.4) illustrates the price behavior of these stocks and illustrates the prediction performance of each stock while comparing classical neural network stock price prediction with that of quantum neural network.

1.3- Tools used for implementation

- Python
- Jupyter Notebook
- Anaconda
- AI & ML
- Quantum Method

Literature Survey

Stock Price Predictions From the research paper “Machine Learning in Stock Price Trend Forecasting” written by Y. Dai and Y. Zhang in Stanford University, they used features like PE ratio, PX volume, PX EBITDA, 10-day volatility, 50-day moving average, etc. to predict the next-day stock price and a long-term stock price . The machine learning algorithms used in the research are Logistic Regression, Gaussian Discriminant Analysis, Quadratic Discriminant Analysis, and SVM. The accuracy ratio is defined as the number of days that the model correctly classified the testing data over the total number of testing days. With the short term model predicting the next day stock price, it has very low accuracy, the Quadratic Discriminant Analysis is the best among all models, it scored a 58.2% accuracy. With the long term model predicting the next n days stock prices, the longer the time frame, the better in the accuracy for SVM. With a time window of 44 days, the SVM model’s accuracy reached 79.3%. Apart from that, it was found that by increasing the number of features, the accuracy increased. When all of the 16 features were used, the accuracy of the model reached 79%, while it fell to 64% when only 8 features were used, and 55% if only 1 feature was used. Our project will also investigate how the timeframe would affect the accuracy of price predictions of different models. As models have to reach a certain threshold to have significance for the users to work as a reference, it is essential for us to optimize our model to figure out what the optimal parameters and model structure are for our stock price prediction purpose.

The research paper “Predicting stock and stock price index movement using Trend Deterministic Data Preparation and machine learning techniques” written by J. Patel, S. Shah, P. Thakkar, and K. Kotecha for the “Expert Systems with Applications” international journal demonstrated a way to use trend deterministic data to predict stock price movement . They conducted experiments in using 10 technical indicators’ signals as inputs, then they use prediction models to predict

whether the stock will go up or down in the coming 10 days, Technical analysis indicators include SMA, EMA, Momentum, Stochastic SK, Stochastic SK, MACD, RSI, etc. The prediction models they have used include ANN, SVM, Random Forest, and Naive Bayesian models. The model outputs “up” or “down” movement signals. Experiments have shown random forest scored the highest performance with 83.56% accuracy with their inputs. B. Wanjawa and L. Muchemi demonstrated the potential in predicting stock prices using ANN, as shown in the research paper “ANN Model to Predict Stock Prices at Stock Exchange Markets” . They used 70% of the training data to predict the stock prices for the next 60 days. Through optimizations, they were able to predict the actual closing prices within 0.71% mean absolute percentage error (MAPE), with the highest variance -3.2% among all of the 62 days. This demonstrated a high potential for using machine learning to accurately predict stock prices. This is one of the key components in our application where algorithms have to be designed to have high accuracy, such that the platform could be useful for retail investors.

MONTECARLO METHOD

Monte Carlo methods, or Monte Carlo experiments, are a broad class of computational algorithms that rely on repeated random sampling to obtain numerical results. The underlying concept is to use randomness to solve problems that might be deterministic in principle.
Monte Carlo Simulation, also known as the Monte Carlo Method or a multiple probability simulation, is a mathematical technique, which is used to estimate the possible outcomes of an uncertain event.In mathematics, Monte Carlo integration is a technique for numerical integration using random numbers. ... While other algorithms usually evaluate the integrand at a regular grid, Monte Carlo randomly chooses points at which the integrand is evaluated. This method is particularly useful for higher-dimensional integrals.A Monte Carlo simulation can be used to tackle a range of problems in virtually every field such as finance, engineering, supply chain, and science. It is also referred to as a multiple probability simulation.

Understanding Monte Carlo Simulations

When faced with significant uncertainty in the process of making a forecast or estimation, rather than just replacing the uncertain variable with a single average number, the Monte Carlo Simulation might prove to be a better solution by using multiple values.Since business and finance are plagued by random variables, Monte Carlo simulations have a vast array of potential applications in these fields. They are used to estimate the probability of cost overruns in large projects and the likelihood that an asset price will move in a certain way.

Telecoms use them to assess network performance in different scenarios, helping them to optimize the network. Analysts use them to assess the risk that an entity will default, and to analyze derivatives such as options.

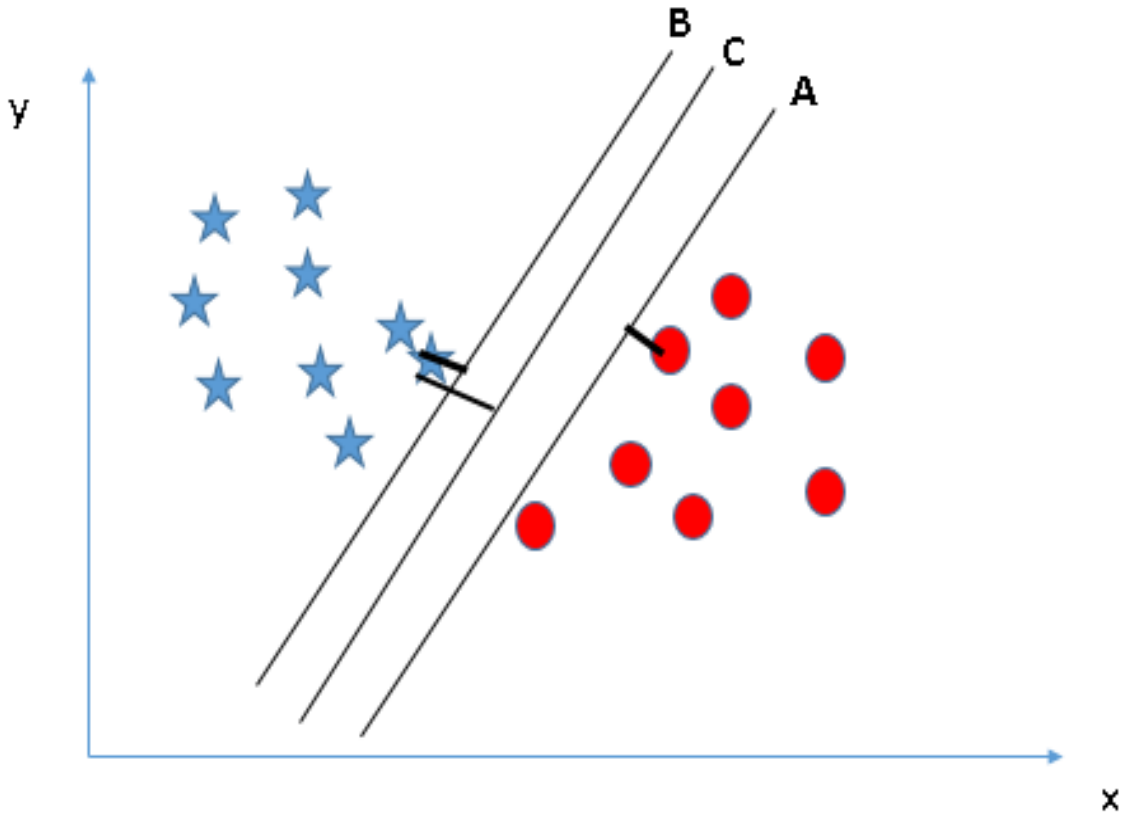
Insurers and oil well drillers also use them. Monte Carlo simulations have countless applications outside of business and finance, such as in meteorology, astronomy, and particle physics.

SVM (Support Vector Machine)

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

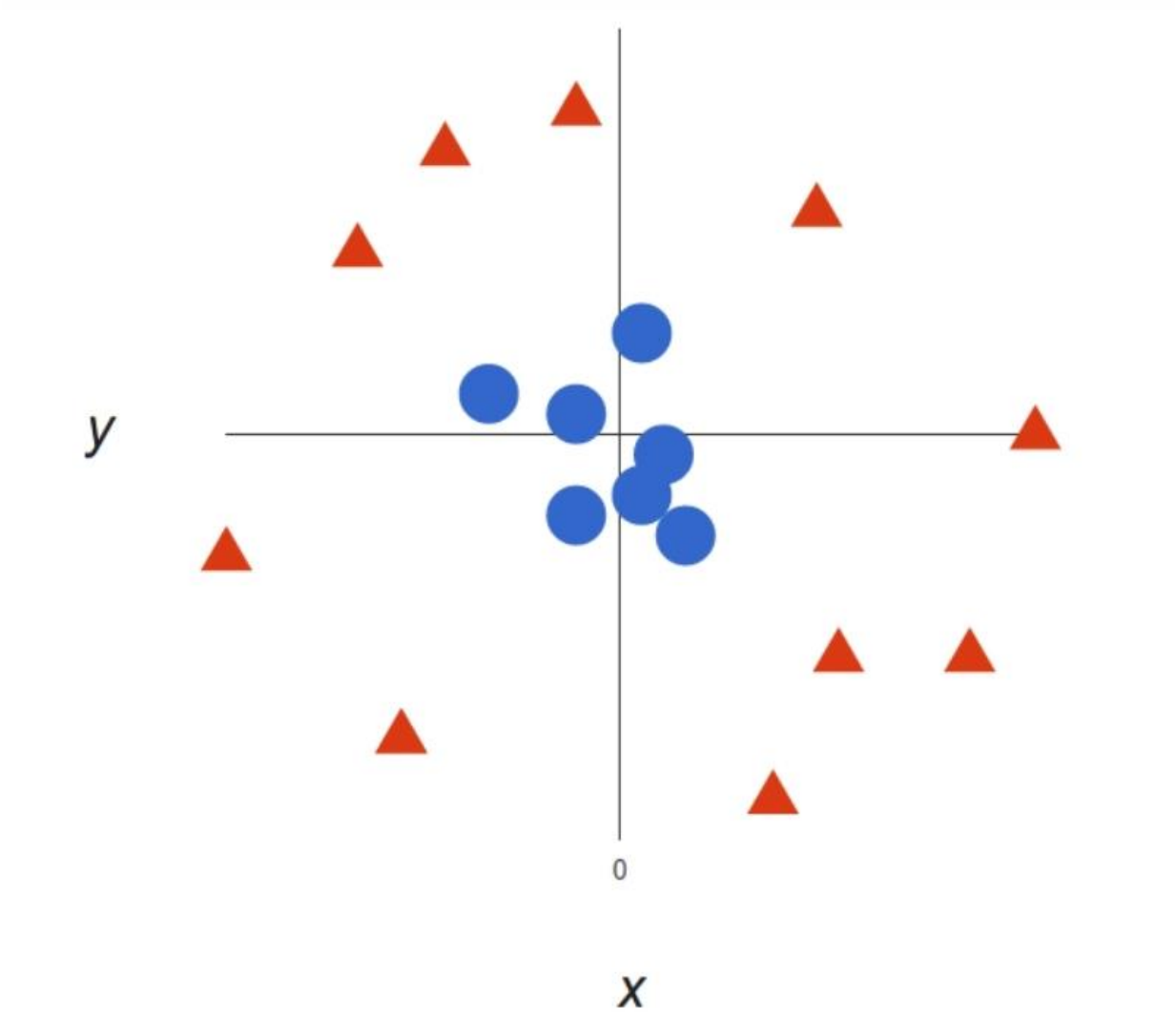
Support Vector Machines are one of the best binary classifiers. They create a decision boundary such that most points in one category fall on one side of the boundary while most points in the other category fall on the other side of the boundary. Consider an n-dimensional feature vector
$x = (X1,...,Xn)$ [8]. We can define a linear boundary (hyperplane) as
$\beta0 + \beta1X1 + ... + \beta nXn = \beta0 + n \sum i=1 \beta iXi = 0$
Then elements in one category will be such that the sum is greater than 0, while elements in the other category will have the sum be less than 0. With labeled examples, $\beta0 + \sum n\ i=1\ \beta iXi = y$, where y is the label. In our classification, $y \in \{-1,1\}$. We can rewrite the hyperplane equation using inner products. $y = \beta0 + \sum \alpha iyix(i) * x$ where $*$ represents the inner product operator. Note that the inner product is weighted by its label. The optimal hyperplane is such that we maximize the distance from the plane to any point. This is known as the margin. The maximum margin hyperplane (MMH) best splits the data. However, since it may not be a perfect differentiation, we can add error variables $\varepsilon1...\varepsilon n$ and keep their sum below some budget B. The crucial element is that only the points closest to the boundary matter for hyperplane selection; all others are irrelevant. These points are known as the support vectors, and the hyperplane is known as a Support Vector Classifier (SVC) since it places each support vector in one class or the other.

Diagram of Linear SVM



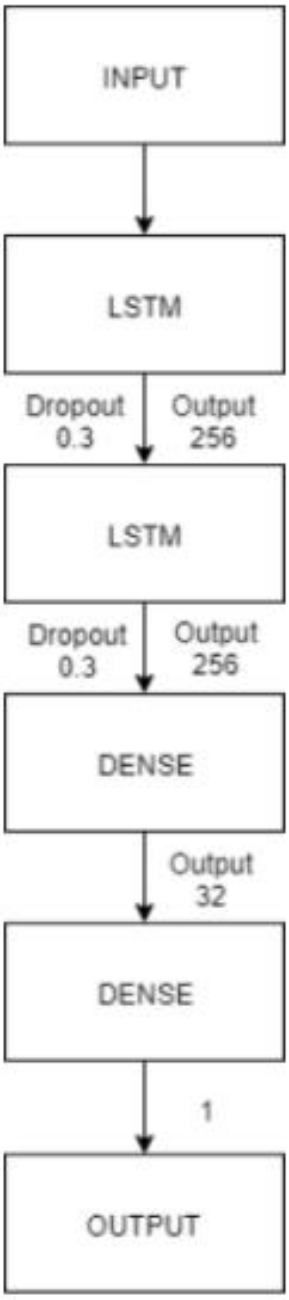
Nonlinear data

Now this example was easy, since clearly the data was linearly separable — we could draw a straight line to separate *red* and *blue*. Sadly, usually things aren't that simple. Take a look at this case:



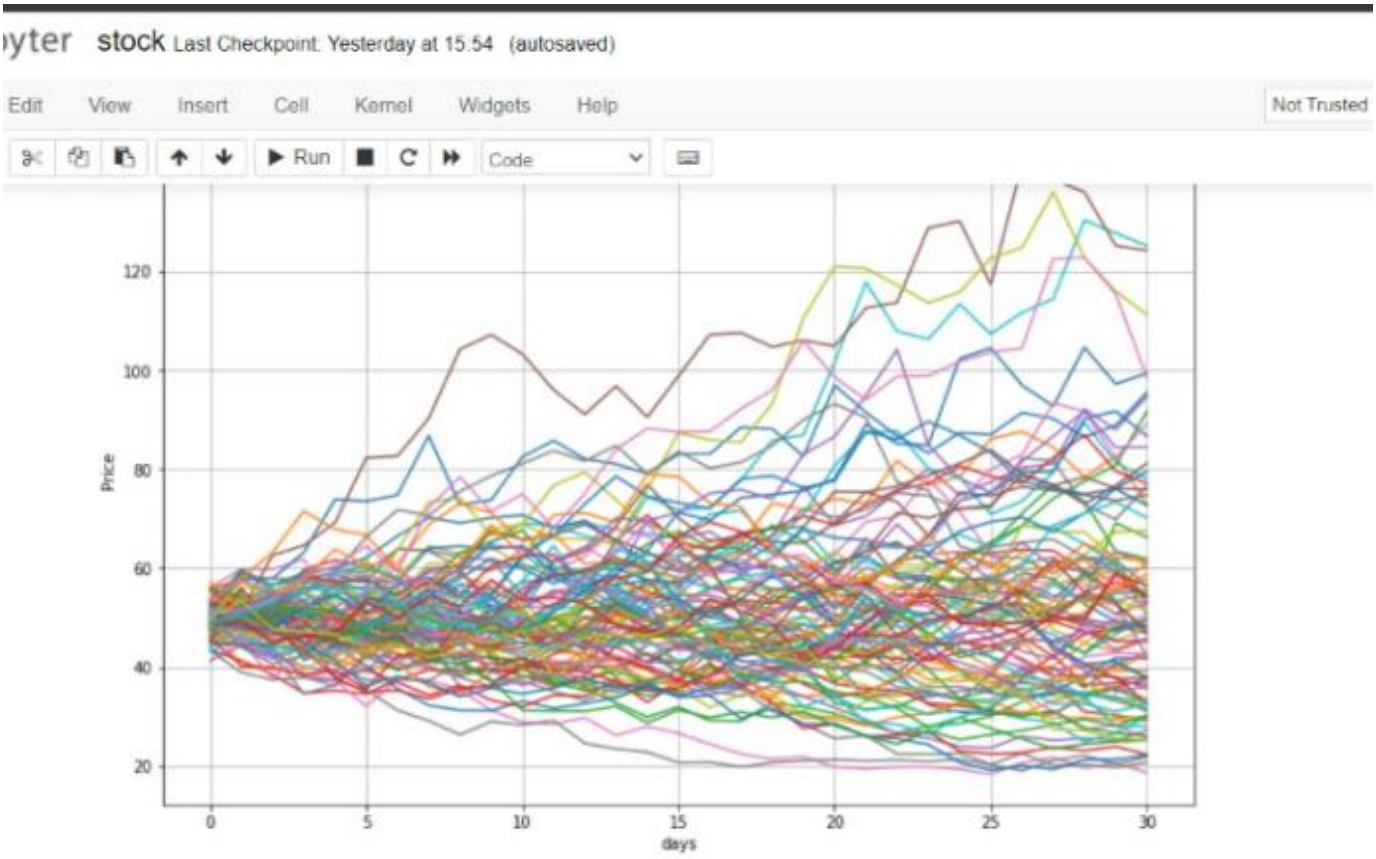
A more complex dataset

Data Flow Diagram


INPUT
LSTM
Dropout
0.3
Output
256
LSTM
Dropout
0.3
Output
256
DENSE
Output
32
DENSE
1
OUTPUT

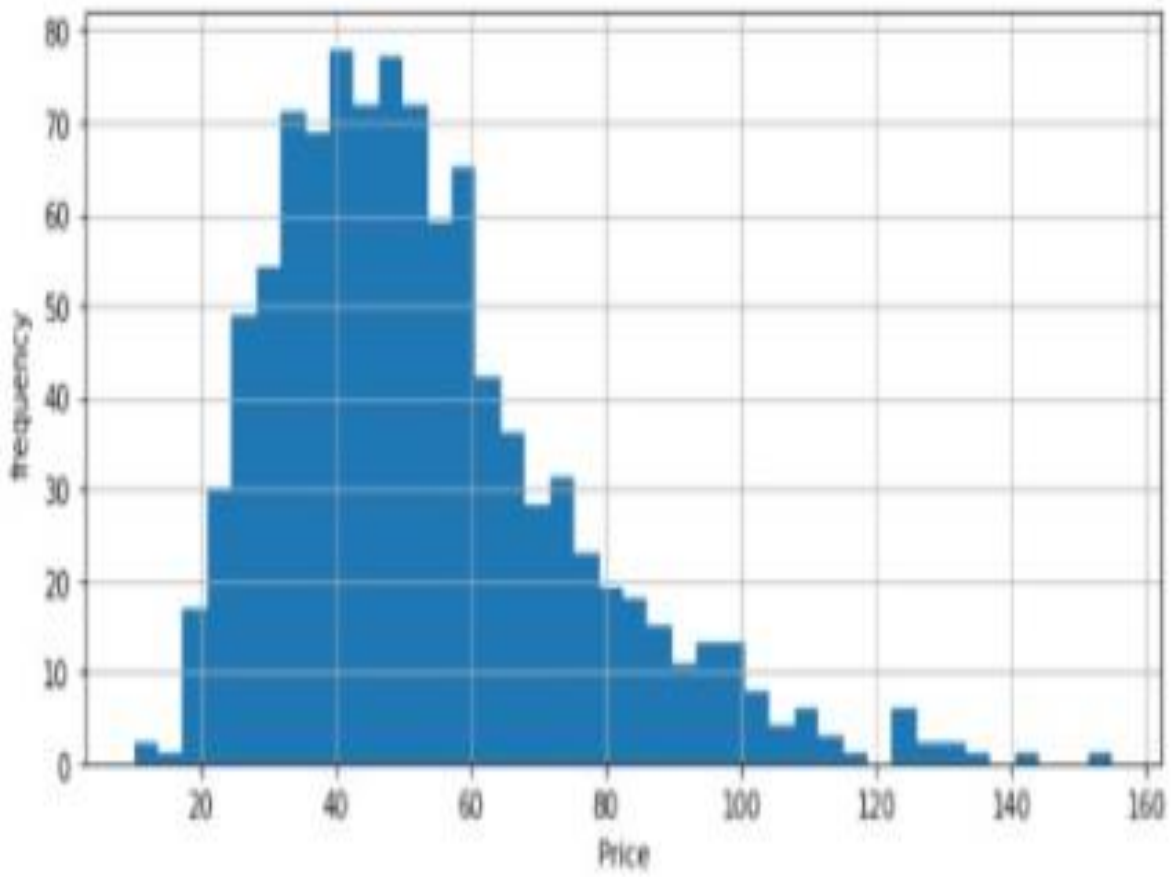
Source Code:-

```
import math
import numpy as np
import matplotlib.pyplot as plt
np.random.seed(42)
S0 = 50
K = 55
r = 0.05
sigma = 0.4
T = 1
t = 30
dt = T/t
M = 1000
S = S0*np.exp(np.cumsum((r- 0.5 sigma **2) *dt + sigma math.sqrt(dt)*
np.ran
dom.standard_normal((t+1, M)), axis=0))
P_call =sum(np.maximum(S[-1] - K, 0)) / M
print("The call option value is: {0.2f} ZAR.\n".format(P_call))


num_paths_to_plot = 100

plt.figure(figsize =(12,8))

plt.plot(S[:, :num_paths_to_plot])

plt.grid (True)

plt.xlabel('days')

plt.ylabel('Price')

plt.title('Possible price paths for a QuantumTech share')
plt.show()
```


yter stock Last Checkpoint: Yesterday at 15:54 (autosaved)
Edit View Insert Cell Kernel Widgets Help
Not Trusted
Run
Code
120
100
80
60
40
20
Price
0
5
10
15
20
25
30
days

```
plt.figure(figsize P=(10,5))
plt.hist (S[-1], bins=50)
plt.grid (True)
plt.xlabel('Price')
plt.ylabel('frequency')
```

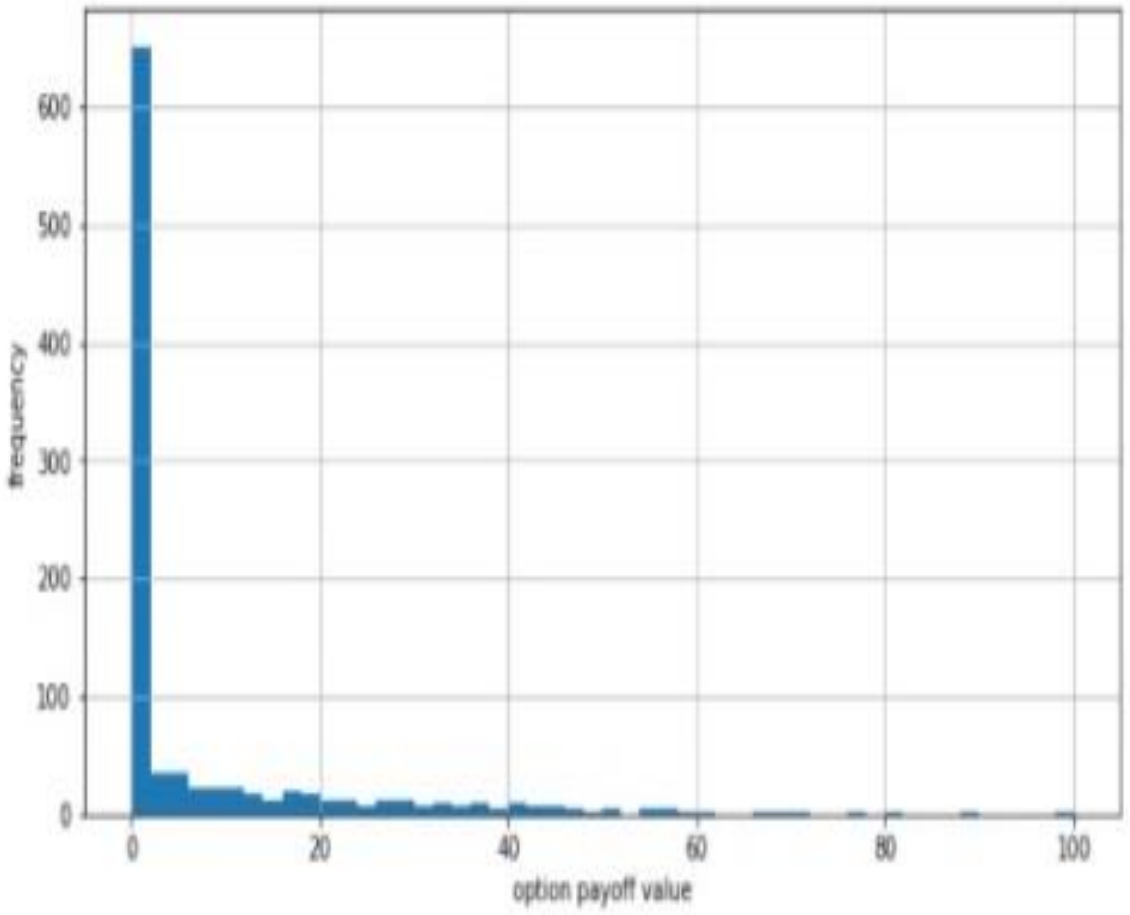
```
Out[10]: Text(0, 0.5, 'frequency')
```



```
plt.figure(figsize= (10,5))
plt.hist(np.maximum(S[-1] - K, 0), bins=50)
plt.grid(True)
plt.xlabel('option payoff value')
plt.ylabel('frequency')
```

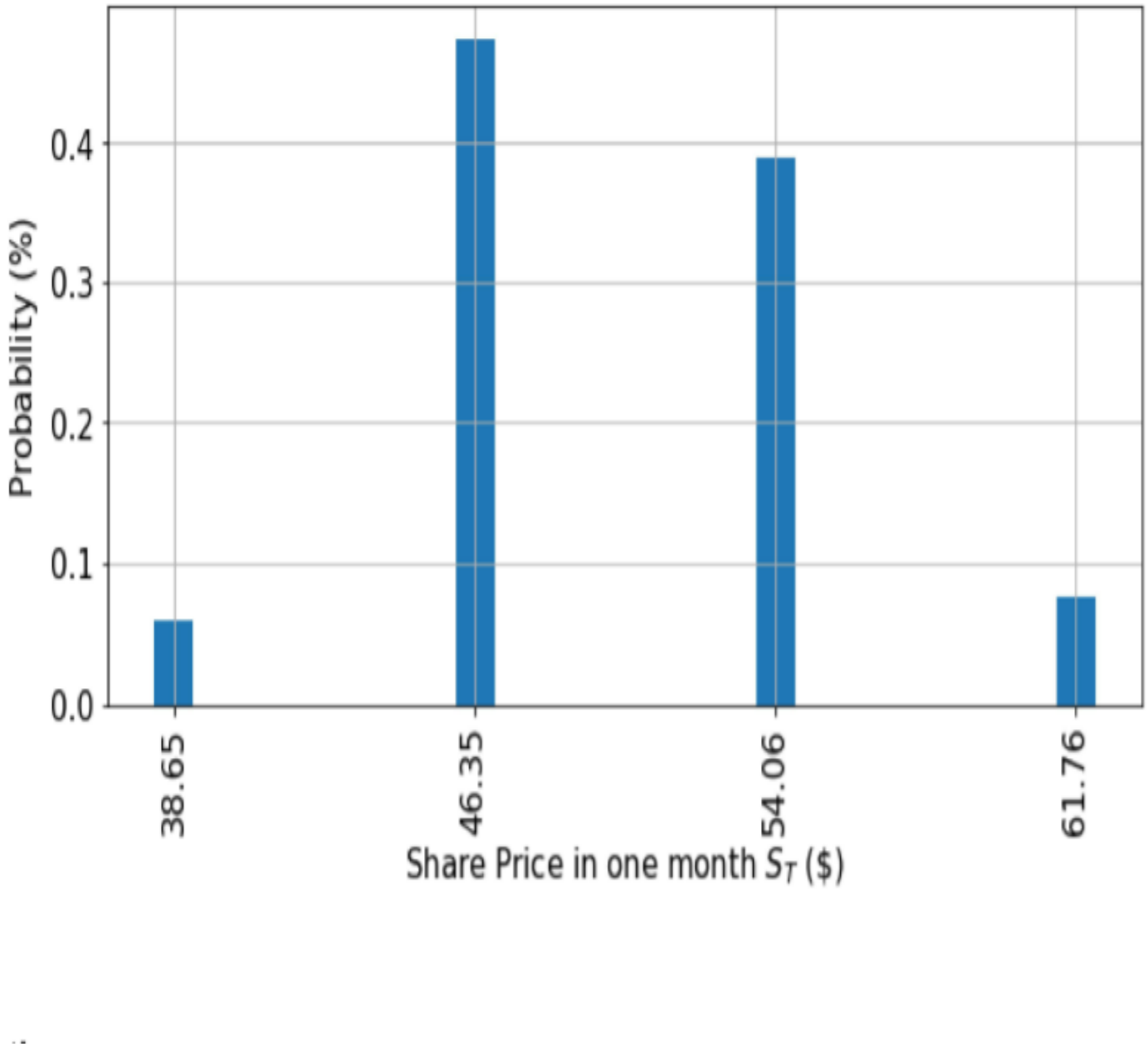
```
plt.xlabel('option payoff value')
plt.ylabel('frequency')
```

```
Out[11]: Text(0, 0.5, 'frequency')
```

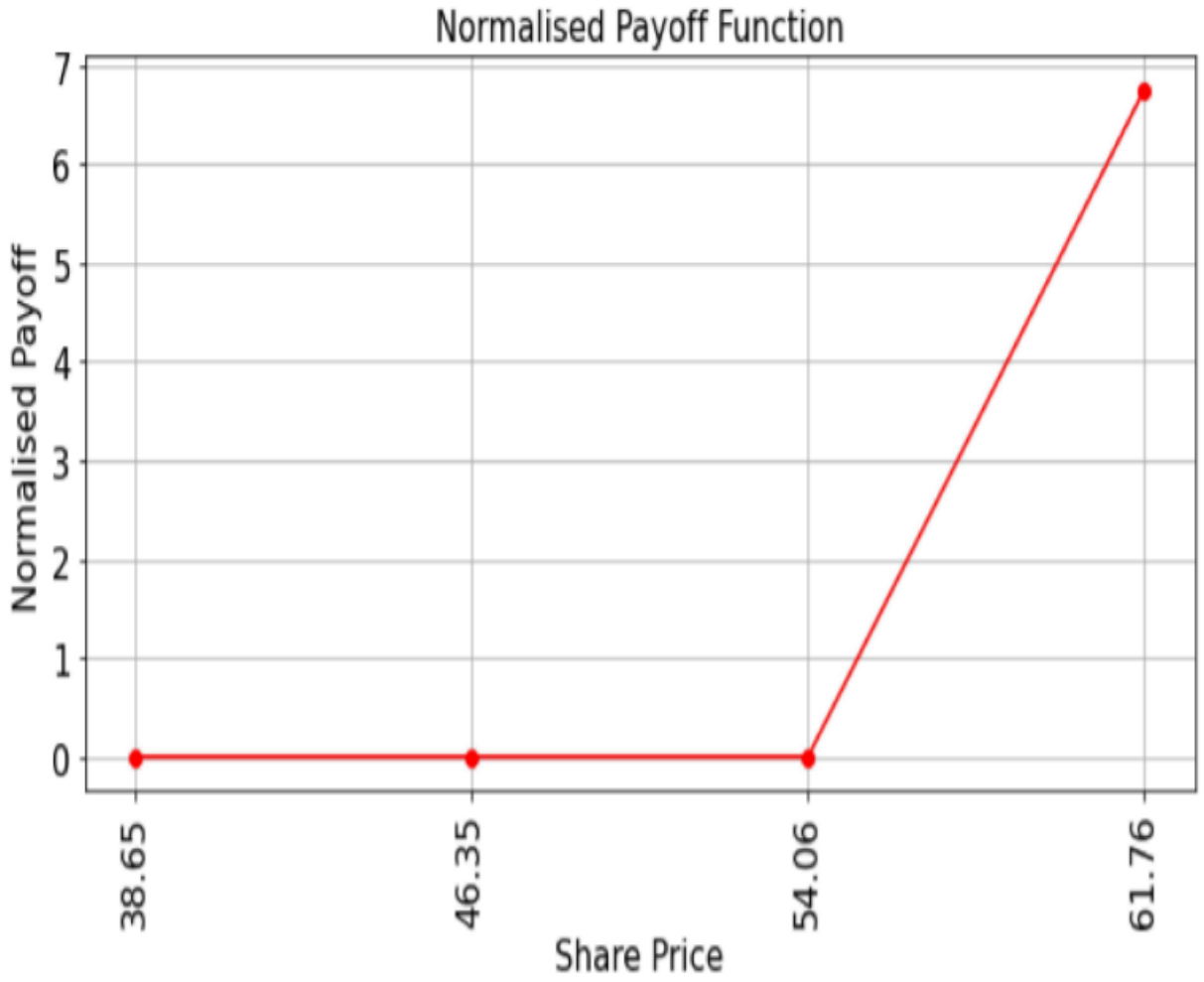


```
from qiskit import Aer
from qiskit.utils import QuantumInstance
from qiskit.algorithms import IterativeAmplitudeEstimation
from qiskit finance.circuit.library import LogNormalDistribution,
EuropeanCallPricing
Objective



num_uncertainty_qubits = 2
# parameters for considered random distribution
S = 50 # initial spot price
strike_price = 55
vol = 0.4 # volatility of 40%
r = 0.05 # annual interest rate of 5%
T = 30/365
mu = ((r - 0.5* vol**2) * T + np.log(S))
sigma = vol* np.sqrt(T)
mean = np.exp(mu + sigma**2/2)
variance = (np.exp(sigma*2) - 1)*np.exp(2*mu + sigma*2)
stddev = np.sqrt(variance)
low = np.maximum (0, mean -2*stddev)
high = mean + 2*stddev
uncertainty_model= LogNormalDistribution(num_uncertainty_qubits, mu=mu,
sigma=sigma**2, bounds=(low, high))




# plot probability distribution

x = uncertainty_model.values
y = uncertainty_model.probabilities
plt.figure(figsize = (10,5))
plt.bar(x, y, width=1)
plt.xticks(x, size = 15 , rotation=90)
plt.yticks(size=15)
plt.grid()
plt.xlabel('Share Price in one month $S_T$ (\$)', size=15)
```

```
plt.ylabel('Probability ($\%$)', size=15)
plt.show()
```



```
x = uncertainty_model.values
y = np.maximum (0, x-strike_price)
plt.figure(figsize= (10,5))
plt.plot(x, y, 'ro-')
plt.grid()
plt.title('Normalised Payoff Function', size=15)
plt.xlabel('Share Price', size=15)
plt.ylabel('Normalised Payoff', size=15)
plt.xticks(x, size = 15 , rotation=90)
plt.yticks(size = 15)
plt.show()
```



```
from qiskit_finance.applications.estimation import EuropeanCallPricing

european_call_pricing =
EuropeanCallPricing(num_state_qubits=num_uncertainty_qubits,
strike_price=strike_price, rescaling_factor=0.25, bounds=(low, high),
uncertainty_model=uncertainty_model)
```

```
epsilon = 0.01
alpha = 0.05
shots = 100
simulator ='qasm_simulator'
qi = QuantumInstance (Aer.get_backend (simulator), shots=shots,
seed_simulator=42, seed_transpiler = 42 )
problem = european_call_pricing.to_estimation_problem()
ae = IterativeAmplitudeEstimation(epsilon, alpha=alpha, quantum_instance=qi)
result = ae.estimate(problem)
```

```
exact_value = np.dot(uncertainty_model.probabilities, y)
conf_int= np.array(result.confidence_interval_processed)
print('Exact value:  \t%.4f' % exact_value)
print('Estimated value:  \t%.4f' % (european_call_pricing.interpret(result)))
print('Estimation error: \t%.4f' %(np.abs(exact_value-
european_call_pricing.interpret(result))))
print('Confidence interval: \t[%.4f, %.4f]' % tuple(conf_int))
```

```
Exact value:           0.5199
Estimated value:       0.5648
Estimation error:      0.0448
Confidence interval:   [0.4871, 0.6424]
```

```
num_uncertainty_qubits = 2
low = np.maximum(0, mean - 2*stddev)
high = mean + 2*stddev
epsilon = 0.01
alpha = 0.05
shots = 100
simulator = 'qasm_simulator'
# Run this cell once you are ready to submit your answer


from qc_grader import grade_ex2a
solutions = [num_uncertainty_qubits, low, high, epsilon, alpha, shots,
simulator]
grade_ex2a(solutions)
```

```
Your score is:  0  try changing more parameters to get an estimation error less than
0.03!
```

```
num_uncertainty_qubits = 4
S = 200 # initial spot price vol = 0.3 # volatility of 30%
vol = 0.3
r = 0.08 # annual interest rate of 4%
T = 60/365 # 60 days to maturity
strike_price = 230
epsilon = 0.01
alpha = 0.05
shots = 100
simulator ='qasm_simulator'
# set the approximation scaling for the payoff function
rescaling_factor = 0.25
# resulting parameters for Log-normal distribution
mu = ((r-0.5 vol2) T + np.log(S))
sigma = vol *np.sqrt(T)
mean = np.exp(mu + sigma**2/2)
variance = (np.exp(sigma*2)- 1) np.exp(2*mu + sigma**2)
stddev = np.sqrt(variance)
```

```
low = np.maximum(0, mean -2*stddev)
high = mean + 2*stddev
breakpoints = [low, high]
slopes = [- 1, 0]
offsets =[strike_price - low, 0]
f_min = 0
f_max = strike_price -low



uncertainty_model = 'insert lognormal model function here'
from qiskit.circuit.library import LinearAmplitudeFunction
european_put_objective = LinearAmplitudeFunction('insert parameter here')
qi = QuantumInstance(Aer.get_backend(simulator), shots=shots,
seed_simulator = 42, seed_transpiler = 42)
ae = IterativeAmplitudeEstimation('insert parameters here')


from qu_grader import grade_ex2b
grade_ex2b(uncertainty_model, european_put_objective, ae)
from qiskit.algorithms import EstimationProblem
european_put_delta_objective = LinearAmplitudeFunction(
num_uncertainty_qubits,
slopes,
offsets,
domain=(low, high),
image=(f_min, f_max),
breakpoints=breakpoints
)
european_put_delta = european_put_delta_objective.compose
(uncertainty_model, front=True)
```

```
problem = EstimationProblem(state_preparation=european_put_delta,
objective_qubits=[num_uncertainty_qubits])
ae_delta = IterativeAmplitudeEstimation(epsilon, alpha=alpha,
quantum_instance=qi)
result_delta = ae_delta.estimate(problem)
x = uncertainty_model.values
exact_delta = -sum(uncertainty_model.probabilities[x <= strike_price])
conf_int = -np.array(result_delta.confidence_interval)[::-1]
print('Exact delta: \t%.4f' % exact_delta)
print('Esimated value: \t%.4f' % -result_delta.estimation)
print('Estimation error: \t%.4f' %(np.abs(exact_delta + result_delta.estimation)
print('Confidence interval: \t[%.4f, %.4f]' % tuple(conf_int))
```

```
import qiskit.tools.jupyter
%qiskit_version_table
%qiskit_copyright
```

RESULTS AND DISCUSSION

The hybrid quantum back propagation model uses 30 quantum hidden neurons and 30 classical output neurons. The performance of the hybrid quantum model has been compared with classical back propagation neural network model having the same number of hidden and output neurons, parameters and data.

The ability of neural networks to discover nonlinear relationships in input data makes them ideal for modeling nonlinear dynamic systems of the stock market has been established. It has been found that the average error using lots of data is smaller than that using less amount of data. That is, the more data for training the neural network, the better prediction it gives. If the training error is low, predicted predictive stock prices are close to the real values.

The back propagation network has performed and predicted stock price trends and thus challenges the EMH. Since, if a neural network can outperform the market consistently or predict its direction with reasonable accuracy, the validity of the EMH is questionable.

The objective of this study was to establish the fact that quantum neural network can perform with almost the same accuracy as the classical ones. Even though, our hybrid quantum model uses quantum hidden neurons containing [8:50 PM, 12/16/2021] Ritik Var: single qubit registers and interfaces with its classical component. All the results, in the figure. 3 for four stocks show that there is a close matching of results in both the cases with slight improvements (around 4 %) using QNN.

It can be further concluded that the processing time in quantum neural networks shall be very small due to the inbuilt capacity of the quantum computers to process the data in parallel. It is evident from the structure of the neural network that the calculations in all the neurons of the layer can be performed concurrently. Quantum computers can have huge memories associated with them and thus the future quantum neural network shall be able to handle large neural networks, thus ensuring the coverage of a wide range of stocks across all corners of stock markets globally.

Finally it is observed that:

- The prices move in a close range with an improvement of about 4% using QNN over CNN.
- Processing time will reduce due to quantum parallelism
- Availability of huge memory will ensure coverage for a wide range of stocks.

The comparison of results between classical and quantum neural networks has revealed that the results match closely and QNN results are better.

Conclusion

The development and application of stock price prediction with both classical and hybrid quantum neural network has revealed that hybrid QNN can produce slightly better results around 4%. The QNN result has established that classical algorithms can be successfully replaced by hybrid QNN when quantum computers become a reality with improved performance and enhanced speed. It can be further concluded that the processing time in quantum neural networks shall be very small due to the inbuilt capacity of the quantum computers to process the data in parallel. It is evident from the structure of the neural network that the calculations in all the neurons of the layer can be performed concurrently. The advantage of quantum computing can be exploited by extending the quantum computation concepts to recurrent Hopfield nets and stochastic Boltzmann neural networks. The extension so achieved can be applied to the solution of combinatorial problems in financial engineering.

References

[1] Masoud, Najeb MH. (2017) “The impact of stock market performance upon economic growth.” International Journal of Economics and Financial Issues 3 (4) : 788–798.

[2] Murkute, Amod, and Tanuja Sarode. (2015) “Forecasting market price of stock using artificial neural network.” International Journal of Computer Applications 124 (12) : 11-15.

[3] Hur, Jung, Manoj Raj, and Yohanes E. Riyanto. (2006) “Finance and trade: A cross-country empirical analysis on the impact of financial development and asset tangibility on international trade.” World Development 34 (10) : 1728-1741.

[4] Li, Lei, Yabin Wu, Yihang Ou, Qi Li, Yanquan Zhou, and Daoxin Chen. (2017) “Research on machine learning algorithms and feature extraction for time series.” IEEE 28th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC): 1-5.

[5] Seber, George AF and Lee, Alan J. (2012) “Linear regression analysis.” John Wiley & Sons 329

[6] S. Gupta and R. Zia, “*Quantum neural networks*”, Technical report, Available:http://www.arxiv.org/PS_cache /quant- ph/ pdf/ 0201/ 0201144.pdf, 2002.

[7] T. Menneer, “Quantum Artificial Neural Networks”, PhD thesis, University of Exeter, 1998.

[8] Ajit Narayanan and Tammy Menneer, “Quantum artificial neural network architectures and components.”, *Information Sciences*, volume 124 nos. 1-4, pages 231–255, 2000.

[9] Ezhov, A. and D. Ventura, “Quantum neural networks”, in: *N. Kasabov(ed) Future Directions for Intelligent Systems and Information Sciences*, Springer Verlag 2000.

[10] Hong Xiao and M. Cao, “Hybrid quantum neural networks model algorithm and simulation” in *the proceedings of the fifth International Conference on Neural Computation Tiaingiin, china*, 2009.

[11] J. A. Miszczak. "*Initialisation of quantum registers based on probability distribution*." Technical report, Available: http://zksi.iitis.gliwice.pl/wiki/projects:kulka
IITiS PAN, 2007.

[12] Michael A. Nielsen and Isaac L. Chuang, *Quantum computation and quantum information*, Cambridge University Press, 2000.