Helwan university,

Statistics and Computer Science

College of Science

Department of Mathematics

B. Sci. Final Year Project

STOCKS PRIECES PREDECTION USING DEEP LEARNING

DISPLAYED ON A WEBSITE

By:

أحمد أيمن سيد محمود

Supervised By: د.أحمد مدبولي

Supervisor(s) *12/7/2021*

DECLARATION

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Bachelor of Science in *(insert title of degree for which registered)* is entirely my own work, that I have exercised reasonable care to ensure that the work is original, and does not to the best of my knowledge breach any law of copyright, and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: ______________________________

Registration No.: ___________

Date: Day, xx Month Year.

Appendix

ABSTRACT

Stock price prediction is one among the complex machine learning problems. It depends on a large number of factors which contribute to changes in the supply and demand. This paper presents the technical analysis of the various strategies deep learning models and data analysis Using indicators of technical analysis, for predicting the price of a stock, and evaluation of a novel approach for the same. Stock prices are represented as time series data and neural networks are trained to learn the patterns from trends.to help traders who do technical analysis, implementing the algorithm into a website will make it easier for amateur traders to make decisions without having a deep understanding of the stock market.

1 INTRODUCTION

1.1 INTRODUCTION

Machine learning uses two types of techniques or methods: **supervised learning**, which trains a model on known input and output data so that it can be able to predict future outputs, and **unsupervised learning**, which finds hidden patterns or intrinsic structures in input data.

But for the purpose of our project, we'll only be talking about **supervised learning,** since **unsupervised learning** is mostly not used in the project.

1.2 MOTIVATION

Efficient Market Hypothesis is one of the most popular theories in financial economics ever. Prices of the securities reflect all the information that is already available, and it seems impossible to outperform the market on a consistent basis. There are three variations of Efficient Market Hypothesis (EMH); that is to say weak form, semi-strong form and the strong form.

Weak form states that the securities reflect all the information that is publicly available in the past. Semi Strong form states that the price reflects all the publicly available data and also, they change instantly to reflect the newly available information. The strong form would include even the insider or private information.

But this theory is often disputed and sometimes highly controversial. Even though predicting the trend of the stock price by doing it manually and interpreting the chaotic market data seems to be and is a tedious task, with the advantage of artificial intelligence, big data and the increasing computational capabilities, automated methods of forecasting the stock prices are becoming more accessible and feasible.

Machine learning models are now able to learn a function by looking at the data without specifically being programmed.

But unfortunately, the time series of any stock is not a function that can be mapped easily. It can be best described more as a random and chaotic walk, which makes the engineering and prediction much harder to make. With Deep Learning, which is a branch of machine learning, one can start training a computer using raw data and features will be automatically made when neural network learns.

Deep Learning techniques and methods are among those famous methods that have been used, to identify the stock trend from large number of data but up until now there is no algorithm or model which could keep consistently predict the price of future stock value correctly. Lot of research is going on both in these academic and industrial challenging problem.

1.3 GOALS AND LIMITATIONS

Stock market is a complicated topic and many things can affect the change in any price. Not only financial factors can affect the price of a stock. Things like daily news or the general public mood can influence the price in several ways positive or negative.

If it was possible to model the stock market with a mathematical function it would be a complex function that works in high-dimensional, maybe infinite dimensional, space. so if we Imagine what would happen if someone knew a method to calculate that function.

That someone would be able to make huge profit by taking advantage of it. However the nature of the space is so complex that finding that function is an almost impossible thing to do. The real challenge is to try and estimate that function using neural-networks in a way that we can make profit by applying it in the stock market. The focus of this thesis is to try to estimate the stock market as clean as possible and try to maximize our profit.

And one of the toughest limitations we had was the gathering of the data itself. Finding a stock market data for the stocks of Google, Apple, and Microsoft was hard enough, but finding the stock price per day for every single one of them was on a different level.

2 FINANCIAL APPROACH

2.1 FINANCIAL INFORMATION

In this section we introduce the reader to the Financial knowledge needed to understand stand completely the context of this thesis.

2.1.1 Information about the Stock Market

2.1.1.1 Stock Market

The Stock Market is the aggregation of buyers and sellers of the shares of stocks, which represent possession claims on companies or businesses. Most of the time, consist of securities indexed on a public stock exchange as however also can be traded privately. An instance of personal trades, consist of stocks of private agencies which might be offered to buyers through equity crowd-funding platforms. Not most effective not unusual place equity shares are indexed in stock exchanges however additionally stocks of different safety kind company bonds and convertible bonds.

2.1.1.2 Stocks

The stock of a company or an organization is expressed as the equity stock of its owners. A single share of the stock acts as a partial ownership of the corporation in ratio to the total amount of shares. The stock of a corporation is split into shares and the total amount of them is stated at the time of the company was created. The existing shareholders can allow the company to issue additional shares. In some cases, each share of stock has a specific declared par value, which is a legal accounting value and it acts as the equity on the balance sheet of the organization. There are also some cases, where shares of a stocks are issued not accompanied by its a par value.

2.1.1.3 Shares of a Stocks

Shares acts as a percentage of ownership in a business. A business may state the different types of shares, each having several different ownership rules, privileges, or share values. The owner of the shares has a documented stock certificate that makes sure of ownership of the stock. A stock certificate is a legal document that describes the amount of shares owned by the shareholder, and other details of them, such as the par value, if any of them, or their class.

2.1.1.4 Shareholders

A shareholder is whither an individual or a company that legally owns an amount of shares in a joint stock company. Shareholders have special advantages that depend on the class of their stock. Some of them have the right to vote on some matters such as elections to the board of directors, the right to take part in distributions of the company's income, the right to buy new shares given by the company and the right to get access to the company's assets througout a liquidation of the company. But, shareholder's rights to a company's assets are limited by the rights of the company's benefactors.

2.1.1.5 Stock market prediction

Stock market prediction is the act of looking to decide the future value of a company's share's price different economic tool traded on an exchange.

2.1.2 Financial Indicators

2.1.2.1 Indicators of stock market

Market indicators are quantitative in nature and are seeking for to interpret stock or financial index information in an try to forecast marketplace moves. Market indicators are a subset of technical indicators and are usually made out of formulation and ratios. They resource investors' investment/buying and selling decisions.

2.1.2.2 Oscillator

An oscillator is a technical analysis tool that constructs high and low bands among intense values, after which builds a trend indicator that fluctuates inside those bounds. Traders use the trend indicator to find out short-term overbought or oversold conditions. When the value of the oscillator approaches the higher excessive value, technical analysts interpret that information to intend that the asset is overbought, and because it tactics the lower excessive, technicians take into account the asset to be oversold.

2.1.2.3 Exponential moving average(EMA)

Exponential Moving average is a weighted moving average. The Weight Multiplier

for n in data point is calculated as:

$$W = \frac{2}{n+1} \qquad \text{Equation 1.1}$$

EMA is calculated as following:

1. Calculate the simple moving average (SMA) the usage of equation

   $$SMA = \frac{1}{n}\sum_{i=1}^{n} x_i \qquad \text{Equation 1.2}$$

   for the chosen quantity of time periods.

2. Calculate the weighted multiplier **the usage of** equation

3. Calculate **preliminary** $EMA\ as\ \{EMA_0\} = (C - SMA) \times W + SMA$

4. Calculate new EMA as $\{EMA_i\} = (\mathrm{C} - \{EMA_{i-1}\} \times W + \{EMA_{i-1}\}$

   As C is the Closing price of the time interval we used .

2.1.2.4 Moving average convergence divergence (MACD)

Moving average convergence divergence (MACD) is a trend-following momentum indicator that indicates the connection among moving averages of a security's price.

The MACD is calculated through subtracting the 26-period exponential moving average (EMA) from the 12-period EMA.

The end result of that calculation is the MACD line. A nine-day EMA of the MACD known as the "signal line," is then plotted on top of the MACD line, which could function as a trigger for purchase and sell signals.

Traders might also additionally purchase the security while the MACD crosses above its signal line and sell—or short—the security while the MACD crosses beneath the signal line. Moving average convergence divergence (MACD) indicators may be interpreted in several ways, however the greater common strategies are crossovers, divergences, and speedy rises/falls.

MACD line is the subtraction of a 26-day EMA from a 12-day EMA.

We calculate EMA the use of the steps in 1.3.1.8 and MACD is given with the aid of using the subsequent formula

$MACD = EMA_{12} - EMA_{26}$ Equation 1.3

Equation 1.4

The sensitivity of the oscillator to marketplace moves is reducible via way of means of adjusting that term or via way of means of taking a moving average of the result.

It is used to generate overbought and oversold buying and selling signals, utilizing a 0–100 bounded range of values.

Its formula

$$\%K = (\frac{C - L_{14}}{H_{14} - L_{14}}) \times 100$$

C = recent closing price

L14 = The lowest price traded of the 14 previous trading sessions

H14 = The highest price traded during the same 14-day period

%K = current value of the stochastic indicator

2.1.2.5 Bollinger Bands

A Bollinger Band is a technical analysis tool described via way of means of a set of trendlines plotted standard deviations (positively and negatively) farfar from a simple moving average (SMA) of a security's price, however which may be adjusted to user preferences.

Bollinger Bands had been evolved and copyrighted via way of means of well-known technical trader John Bollinger, designed to find out opportunities that provide investors a better chance of nicely figuring out whilst an asset is oversold or overbought.

From Equation 1.2

$$UpperBB = SMA + D\sqrt{\frac{\sum_{i=1}^{x}(y_i - SMA)^2}{n}}$$ Equation 1.5

$$UpperBB = SMA - D\sqrt{\frac{\sum_{i=1}^{x}(y_i - SMA)^2}{n}}$$ Equation 1.6

2.1.2.6 Chande Momentum Oscillator

The Chande momentum oscillator is a technical momentum indicator introduced with the aid of using Tushar Chande in his 1994 book The New Technical Trader. The formula calculates the distinction among the sum of new profits and the sum of new losses after which divides the end result with the aid of using the sum of all price actions over the equal period.

Formula :

Chande Momentum Oscillator = $\frac{sH - sL}{sH + sL}$ % 100

where:

sH=the sum of higher closes over N periods

sL=the sum of lower closes of N periods

How To Calculate the Chande Momentum Oscillator ?

1.Calculate the sH closes over N periods.

2.Calculate the sL closes over N periods.

3.Subtract the sL closes over N periods from the sH closes over N periods.

4.Add the sL closes over N periods to the sH closes over N periods.

5.Divide 4 from 3 and multiply by 100.

2.1.2.7 The Commodity Channel Index (CCI)

The Commodity Channel Index (CCI) is a momentum-primarily based totally oscillator used to assist determine when an investment vehicle is achieving a situation of being overbought or oversold.

Developed through Donald Lambert, this technical indicator assesses price trend route and strength, permitting traders to decide in the event that they need to go into or go out a trade, chorus from taking a trade, or add to an current position. In this way, the indicator can be used to offer trade signals when it acts in a certain way.

Formula :

$$CCI = \frac{TP - MA}{.015 \times MD}$$

TP : typical price > TP = $\sum_{i=1}^{P}((High + Low + Close)/3)$

P = number of periods

MA = Moving average

Moving Average = ($\sum_{i=1}^{P}(TP) /P$

Mean Deviation = $\sum_{i=1}^{P}(|TP - MA|)/p$

2.1.2.8 Rate of Volume Change

The volume rate of change is the indicator that indicates whether or not or now no longer a volume trend is developing in both an up or down direction. You may be acquainted with price rate of change (discussed here), which indicates an investor the rate of change measured via way of means of the issue's closing price.

To calculate this, you need to divide the volume change during the last n-durations (days, weeks or months) through the extent n-periods ago. The solution is a percent change of the extent during the last n-periods. Now, what does this mean? If the volume nowadays is higher than n-days (or weeks or months) ago, the price of change might be a plus number. If volume is lower, the ROC might be minus number. This permits us to examine the rate at which the volume is changing.

One of the issues that analysts have with the V-ROC is determining the time frame to measure the rate of change. A shorter duration of 10 to 15 days, for example, could display for us the peaks created through a unexpected change, and, for the most part, trendlines could be drawn.

2.2 WILLIAMS %R

WILLIAMS %R developed by Larry Williams, Williams %R is a momentum indicator that is the opposite of the Fast Stochastic Oscillator. Also know to as %R, Williams %R show the level of the close relative to the highest high for the look-back period. Meanwhile, the Stochastic Oscillator shows the level of the close relative to the lowest low. %R improves the inversion by multiplying the raw value by -100. So, the Fast Stochastic Oscillator and Williams %R make exactly the same lines, but with different scaling. Williams %R oscillates from 0 to -100; readings from 0 to -20 are considered to be overbought, while readings from -80 to -100 are considered to be oversold. Predictably, signals made from the Stochastic Oscillator are also applicable to Williams %R.

2.2.1 Williams %R

The Williams %R has default setting is 14 periods, which can be days, weeks, months or it can be an intraday timeframe. A 14-period %R would use the most recent close, the highest high over the last 14 periods and the lowest low over the last 14 periods.

```
%R = (Highest High - Close)/(Highest High - Lowest Low) * -100


Lowest Low = lowest low for the look-back period
Highest High = highest high for the look-back period
%R is multiplied by -100 correct the inversion and move the decimal.
```

2.3 PRICE RATE OF CHANGE INDICATOR (ROC)

The Price Rate of Change also known as (ROC) is a momentum-based technical indicator that calculate the percentage change in price between the price a certain number of periods ago and the current price. The ROC indicator is plotted against zero, with the indicator moving upwards into positive territory if price changes are to the upside, and moving into negative region if price changes are to the downside.

The indicator can be used to spot separation, overbought and oversold conditions, and centre line crossovers.

2.3.1 Price Rate of Change formula

$$ROC = (\frac{Closing\ Price_{p} - Closing\ Price_{p-n}}{Closing\ Price_{p-n}})$$

Where:

$Closing\ Price_{p-n} = Closing\ price\ of\ most\ recent\ period$

$Closing\ Price_{p-n} = Closing\ price\ n\ periods\ before\ most\ recent\ period$

3 TECHNICAL APPROACH

3.1 WHAT IS MACHINE LEARNING?

Machine learning is a data science analytics technique that teaches computers to do what us humans do naturally when we were born: which is learn from experience. Machine learning algorithms use computational and math mathematical methods to "**learn**" information directly from data without relying on a already chosen equations as a model. The algorithms adaptability improves their performance as the number of samples for learning increases.

Machine learning algorithms find naturally occurring patterns in data that generate awareness and help us make better decisions and predictions. They are used every day to make critical decisions in medical diagnosis, stock trading, weather forecasting, and more.

We consider using machine learning when we have a very complex task or problem that involves a large amount of data and lots of variables, but no existing formula or equation.

3.1.1 How Machine Learning Works

Machine learning uses two types of techniques or methods: **supervised learning**, which trains a model on known input and output data so that it can be able to predict future outputs, and **unsupervised learning**, which finds hidden patterns or intrinsic structures in input data.

But for the purpose of our project, we'll only be talking about **supervised learning,** since **unsupervised learning** is mostly not used in the project.

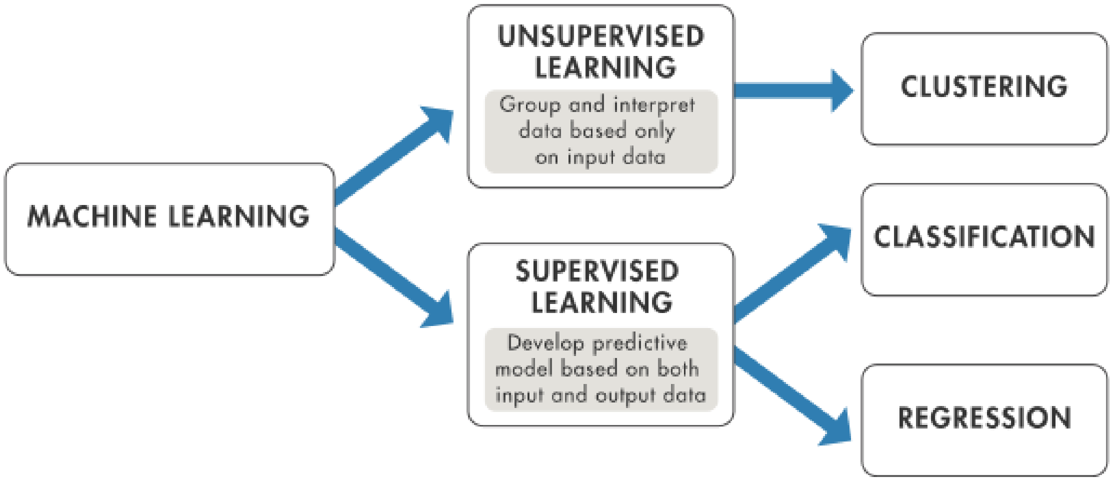

MACHINE LEARNING
UNSUPERVISED LEARNING
Group and interpret data based only on input data
SUPERVISED LEARNING
Develop predictive model based on both input and output data
CLUSTERING
CLASSIFICATION
REGRESSION

Figure 1. Machine learning techniques include both unsupervised and supervised learning.

3.1.1.1 Supervised Learning

Supervised machine learning builds or makes a model that produce predictions based on evidence in the presence of uncertainty. A supervised learning algorithm takes a known set of input data and their known responses to the data (output) and trains a model to make reasonable predictions for the response to new data. We use supervised learning if we do have known data for the output of the thing we are trying to predict.

Supervised learning has two techniques which are classification and regression, and we use the to develop machine learning models.

- **Classification techniques** predict discrete responses—for example, to know whether an email is genuine or spam, or whether a tumor is cancerous or not. Classification models classify input data into categories. Typical applications would include medical imaging, speech recognition, and credit scoring.
  We use classification if our data can be tagged, categorized, or separated into small specific groups or classes.
  Common algorithms for classification would include support vector machine (SVM), boosted and bagged decision trees, k-nearest neighbor, Naïve Bayes, discriminant analysis, logistic regression, and neural networks.
- **Regression techniques** predict continuous responses—for example, to predict changes in temperature or fluctuations in power demand. Typical applications would include weather forecasting and algorithmic trading. We use regression techniques if we are working with a data range or if the nature of our response is a real number.

  Common regression algorithms or methods would include linear model, nonlinear model, regularization, stepwise regression, boosted and bagged decision trees, neural networks, and adaptive neuro-fuzzy learning.

So, to keep it simple we choose supervised learning if we need to train a model to make a prediction—for example, the future value of a continuous variable, such as temperature or a stock price, or a classification—for example, identify makes of cars from webcam video footage.

3.1.2 Problems with machine learning

Machine Learning is a very powerful tool but even it had some problems that it couldn’t deal with correctly.

- Traditional Machine Learning algorithms are not useful while working with data that is high dimensional, that is when we have a huge number of inputs and outputs. For example, in the case of handwriting recognition we have a huge amount of input where we’ll have different types of inputs associated with different types of handwriting.
- Second major challenge or flaw is to tell the computer what kind of features are it should be looking for that will play an important role in predicting the outcome as well as to achieving better accuracy while doing so. This process is known as **feature extraction**.

giving raw data to the algorithm rarely works and this is the reason why feature extraction is an important part of the traditional machine learning workflow. That's why, without feature extraction, the work for the programmer becomes harder as the effectiveness of algorithm very much depends on how accurate the programmer is.

So, it is very hard to apply these ML models or algorithms to complex problems like object recognition, handwriting recognition, Natural Language Processing, etc.

But over time developers a solution to these problems but without sacrificing the accuracy of the algorithms which was **Deep Learning**.

3.1.3 Support Vector Machines (SVM)

A support vector machine (SVM) is a supervised machine learning model or algorithm that uses classification algorithms for two-group classification problems. After feeding an SVM model sets of labelled training data for each category, they're able to categorize new Samples.

It relies on statistics. And can be used for both regression and classification problems.

Compared to newer algorithms like neural networks algorithms, they have two main advantages over them: higher speed and better performance with a limited number of given samples (in the thousands at most). This makes the algorithm very suitable for classification problems, where it's common to have access to a dataset of a couple of thousands at most of tagged samples.

3.1.3.1 How does SUPPORT VECTOR MACHINES (SVM) work

For example: Let's imagine that we have two tags: red tags and blue tags, and our data has two features: x and y. We want a classifier that, when it is given a pair of (x,y) coordinates, it outputs if it's either red or blue. We plot our training data which is already labeled on a plane:

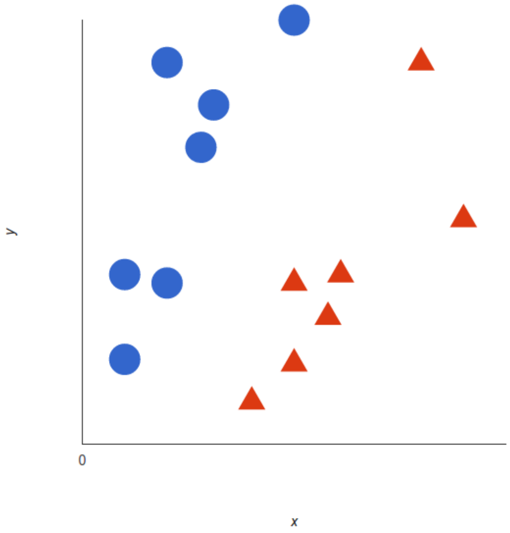


Figure 3.1

A support vector machine takes these data points and gives back the hyperplane (which happens to be a line in two dimensions) that gives the best separation between the tags. This line is called the decision boundary: anything that falls to one side of it will be classified as blue, and anything that falls to the other side classified as red.

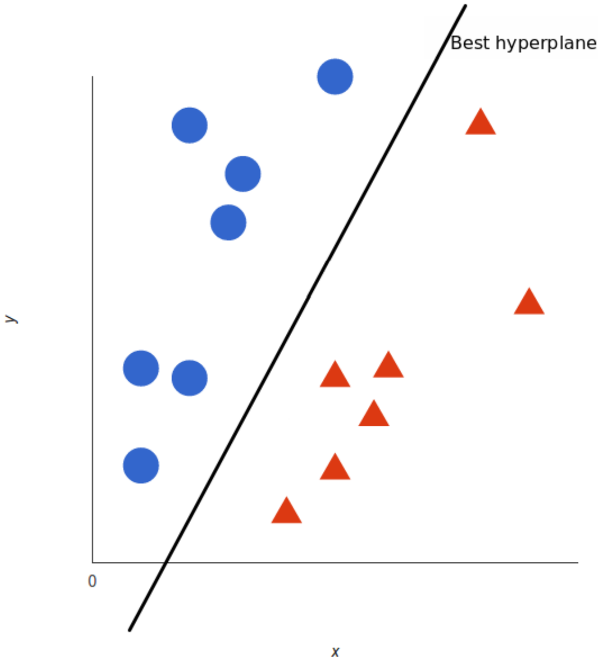


Figure 3.2

But how do we find the best hyperplane? For SVM, it’s the one that maximizes the margins from both of the tags. In other words: the hyperplane (remember it's a line in this case) whose distance to the nearest element or sample of each tag is the largest and equal in distance. Look at figure 3.3

The previous example was too easy not every problem will have two classes or will be represented in a linear fashion, which means the higher complexity of a model the harder it is to find the hyperplane. And the job of the programmer is to find which hyperplane that suits which problems.

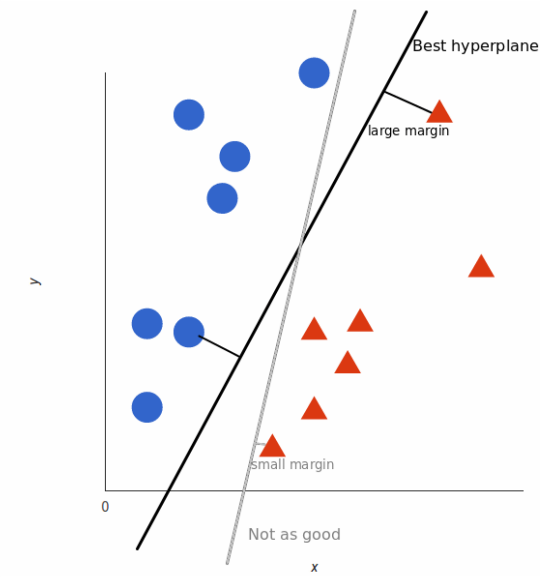


Figure 3.3

3.2 DEEP LEARNING

Deep learning basically means that, when algorithms exposed to different situations or patterns of data, these algorithms adapt. Uncovering or highlighting features in data that we never specifically programmed them to look for, and so we can say they learn on their own. This behaviour is what people are often describing it when they talk as AI.

3.2.1 Machine Learning Vs Deep Learning

Machine learning (ML). ML uses mathematical algorithms that allow machines to gain the ability to learn. Machine learning is a way of analysis used to solving problems through identification, classification or prediction. Algorithms learn from the data entered and then use this new knowledge to draw conclusions from new data.

Deep learning in concept is very similar to machine learning but uses different kind of algorithms. While machine learning uses regression algorithms and or decision trees, deep learning uses neural networks that can function very similarly to the biological neural connections of the human brain.

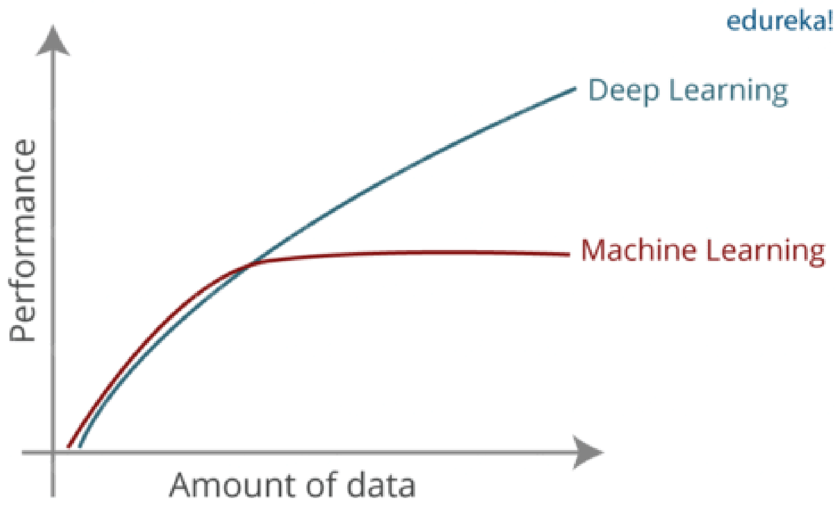


Figure 4: Machine learning Vs Deep Networks

3.2.2 How Deep Learning works

Deep learning is one of the few methods by which we can conquer the challenges of feature extraction. This is because deep learning models are capable of learning to focus on the right features by themselves with little guidance from the programmer. Basically, deep learning can mimic the way human brains functions i.e. it can learns from experience. As we know, the human brain is made up of billions and billions of neurons that allows us to do our everyday tasks. Even the brain of a one-year-old kid can solve complicated problems which are very difficult to solve even when we are using super-computers. For example:

- Recognizing the faces of their parents and different looking objects as well.
- distinguishing the difference between voices and can even recognize a particular person based on their voice.
- Draw assumption from facial gestures of other persons and many many more.

Our brain has sub-consciously trained itself to do tasks over the years. Deep learning uses the idea of artificial neurons that functions in a close manner as the biological neurons present in our brain. So, it can be said that Deep Learning is a subfield of **machine learning** concerned with algorithms motivated by the structure and function of the human brain called artificial neural networks.

Now, let’s take an example to gain a higher understanding of it. Suppose we want to produce a system that can be able to recognize faces of different people in a given image. If we solve this as a regular machine learning problem, we would define the facial features such as eyes, nose, ears etc. and then, the system would identify which are the major features and more important ones for each person on its own.

Now, deep learning takes this one step further. Deep learning automatically finds out the features which are major and important for classification because of constructed deep neural networks, whereas in case of Machine Learning we had to manually define what these features are.

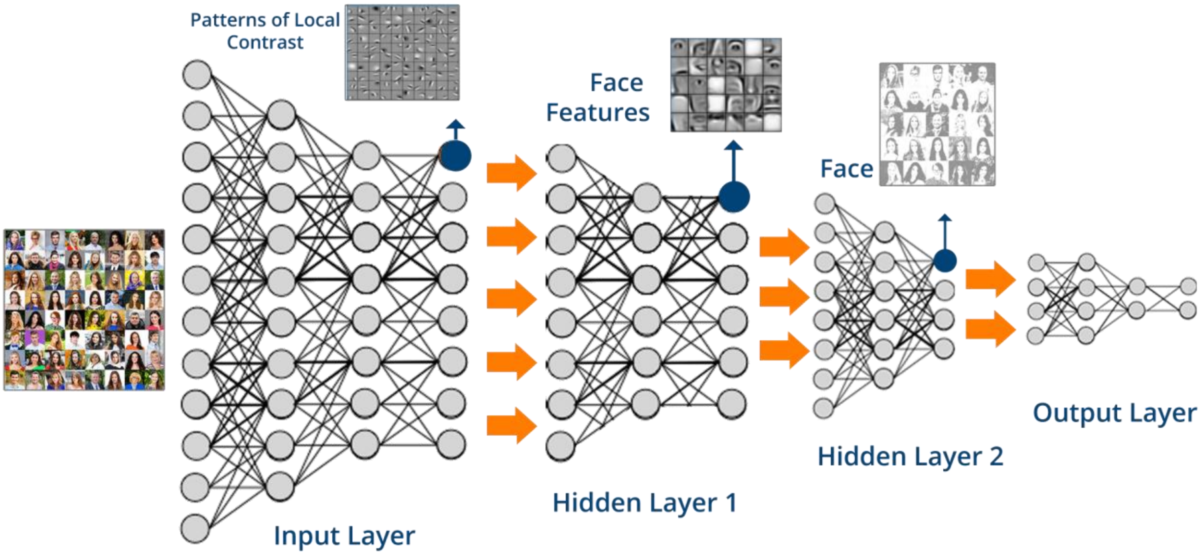


Figure 5: Face Recognition using Deep Networks

As previous in the image Deep Learning works as follows:

- At the lowest level, network focuses on patterns of local contrast as major.
- The next layer is then able to use those patterns of local contrast to focus on things that look like eyes, noses, and mouths.
- Finally, the top layer is able to put in those facial features to face templates.
- A deep neural network can compose more and more complex features in each of its successive layers on its own.

Nowadays deep learning is used by multiple companies for various reasons like how Facebook uses it to tag us in photos, or how YouTube uses it to recommend us videos.

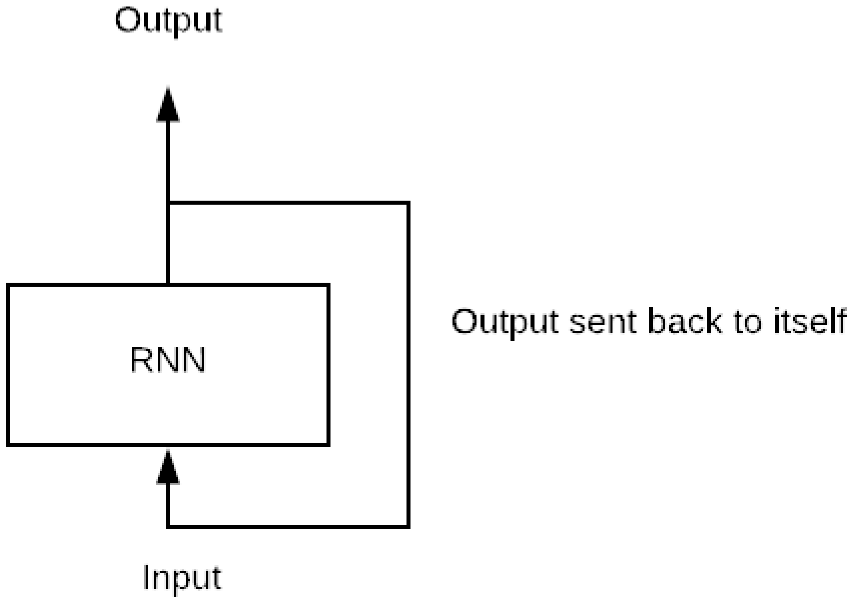
3.2.3 Recurrent Neural Network (RNN)

A Recurrent Neural Network (RNN) is a class of Artificial Neural Network in which the connection between different nodes forms a directed graph to give a temporal dynamic behavior. It helps to model sequential data that are derived from feedforward networks. It works similarly to human brains to deliver predictive results.

A recurrent neural network looks quite similar to a traditional neural network except that a memory-state is added to the neurons. The computation to include a memory is simple.

Imagine a simple model with only one neuron feeds by a batch of data. In a traditional neural net, the model produces the output by multiplying the input with the weight and the activation function. With an RNN, this output is sent back to itself number of time. We call timestep the amount of time the output becomes the input of the next matrice multiplication.

For instance, in the picture below, you can see the network is composed of one neuron. The network computes the matrices multiplication between the input and the weight and adds non-linearity with the activation function. It becomes the output at t-1. This output is the input of the second matrix multiplication.



3.2.4 Why do we need a Recurrent Neural Network (RNN)?

Recurrent Neural Network (RNN) allows you to model memory units to persist data and model short term dependencies. It is also used in time-series forecasting for the identification of data correlations and patterns. It also helps to produce predictive results for sequential data by delivering similar behavior as a human brain.

The metric applied is the loss. The higher the loss function, the dumber the model is. To improve the knowledge of the network, some optimization is required by adjusting the weights of the net. The stochastic gradient descent is the method employed to change the values of the weights in the rights direction. Once the adjustment is made, the network can use another batch of data to test its new knowledge.

The error, fortunately, is lower than before, yet not small enough. The optimization step is done iteratively until the error is minimized, i.e., no more information can be extracted.

3.2.5 Applications of RNN

RNN has multiple uses, especially when it comes to predicting the future. In the financial industry, RNN can be helpful in predicting stock prices or the sign of the stock market direction (i.e., positive or negative).

RNN is useful for an autonomous car as it can avoid a car accident by anticipating the trajectory of the vehicle.

RNN is widely used in text analysis, image captioning, sentiment analysis and machine translation. The machine can do the job with a higher level of accuracy.

3.2.6 Limitations of RNN

In theory, RNN is supposed to carry the information up to time . However, it is quite challenging to propagate all this information when the time step is too long. When a network has too many deep layers, it becomes untrainable. This problem is called: vanishing gradient problem. If you remember, the neural network updates the weight using the gradient descent algorithm. The gradients grow smaller when the network progress down to lower layers.

In conclusion, the gradients stay constant meaning there is no space for improvement. The model learns from a change in the gradient; this change affects the network's output. However, if the difference in the gradient is too small (i.e., the weights change a little), the network can't learn anything and so the output. Therefore, a network facing a vanishing gradient problem cannot converge toward a good solution.

3.2.7 Improvement LSTM

To overcome the potential issue of vanishing gradient faced by RNN, three researchers, Hochreiter, Schmidhuber and Bengio improved the RNN with an architecture called Long Short-Term Memory (LSTM). In brief, LSMT provides to the network relevant past information to more recent time. The machine uses a better architecture to select and carry information back to later time.

3.2.8 RNN in time series

Time series analysis refers to the analysis of change in the trend of the data over a period of time. Time series analysis has a variety of applications. One such application is the prediction of the future value of an item based on its past values.

Time series are dependent to previous time which means past values includes relevant information that the network can learn from. The idea behind time series prediction is to estimate the future value of a series, let's say, stock price, temperature, GDP and so on.

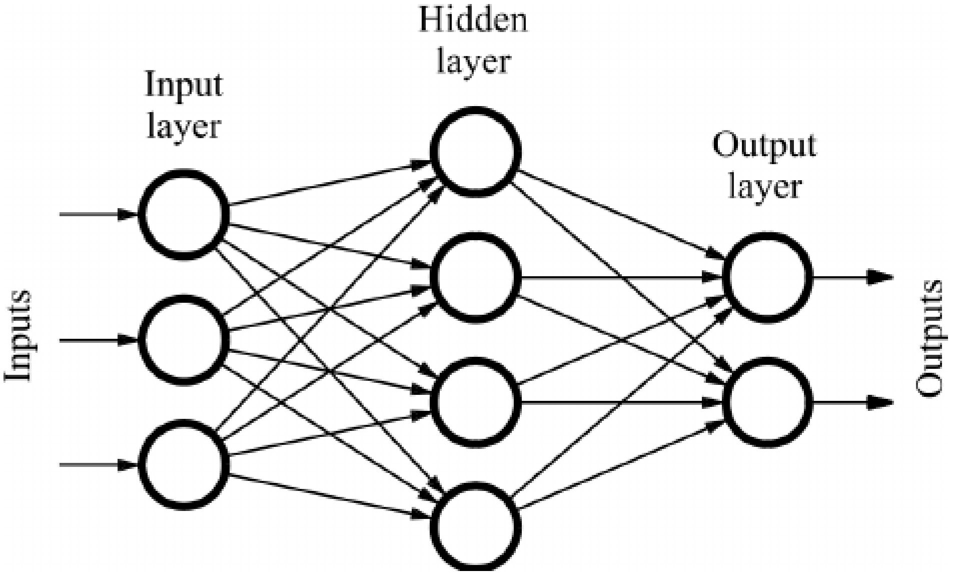
The data preparation for Keras RNN and time series can be a little bit tricky. First of all, the objective is to predict the next value of the series, meaning, you will use the past information to estimate the value at t + 1. The label is equal to the input sequence and shifted one period ahead. Secondly, the number of input is set to 1, i.e., one observation per time. Lastly, the time step is equal to the sequence of the numerical value. For instance, if you set the time step to 10, the input sequence will return ten consecutive times.

Future stock price prediction is probably the best example of such an application.

3.2.9 What is the difference between LSTM, RNN and sequence to sequence?

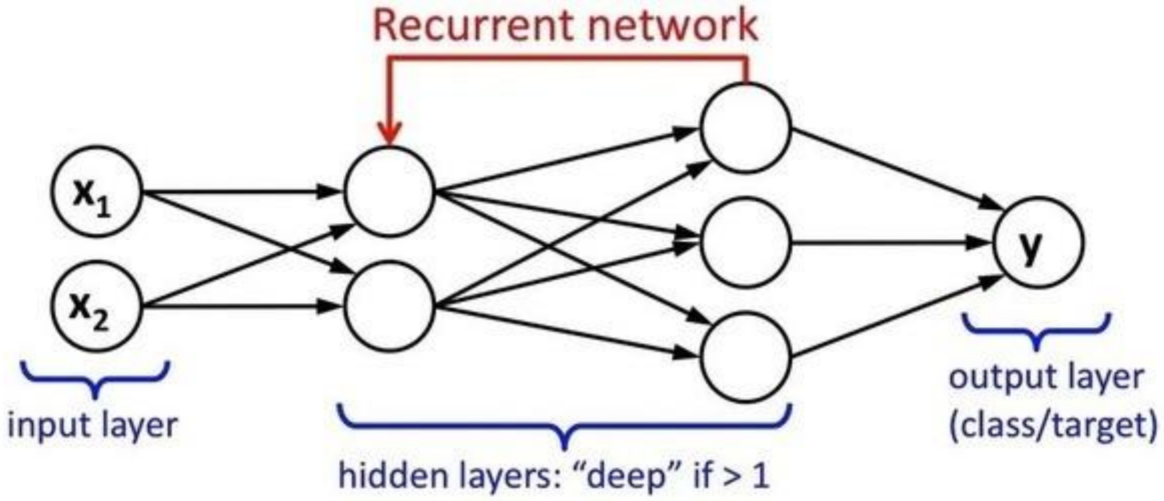
First, sequence-to-sequence is a problem setting, where your input is a sequence and your output is also a sequence. Typical examples of sequence-to-sequence problems are machine translation, question answering, generating natural language description of videos, automatic summarization, etc. LSTM and RNN are models, that is, they can be trained using some data and then be used to predict on new data. You can have sequence-to-sequence models that use LSTMs/RNNs as modules inside them, where a “sequence-to-sequence model” is just a model that works for sequence to sequence tasks.

RNNs is a class of neural networks that is not feed-forward. A feed-forward neural network looks something like this:



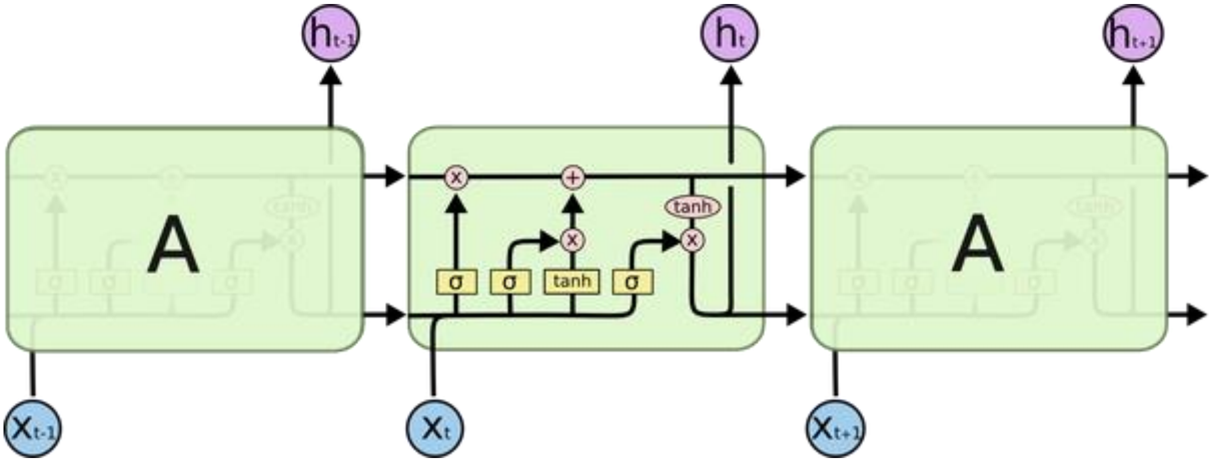
As the name suggests, you only feed the input in the forward direction, that is, the output from one unit is only fed to units further ahead.

On the other hand, an RNN looks something like the following:



That is, there are loops in the network graph, and the output of one unit may go back to one of the already visited units.

Finally, LSTMs are a special type of RNNs, where you connect these units in a specific way, to avoid some problems that arise in regular RNNs [vanishing and exploding gradient].



Here, A is the unit that takes an input X, outputs a vector h, and also some other vectors that are fed back to A [shown by right arrows from one A to another]. Note that all the A’s here refer to a single object in memory — they are drawn in this unrolled way for explanation. What’s really happening is that the outputs that come out from the right of A are fed back from the left to the same A module.

There are many ways in which you can use LSTMs for sequence-to-sequence tasks. As the above diagram shows, you can feed in the input sequence to an LSTM and get a sequence of h vectors. You can then process these h vectors further.

3.2.10 Sequence Classification with LSTM Recurrent Neural Networks in Python with Keras

Sequence classification is a predictive modeling problem where you have some sequence of inputs over space or time and the task is to predict a category for the sequence.

What makes this problem difficult is that the sequences can vary in length, be comprised of a very large vocabulary of input symbols and may require the model to learn the long-term context or dependencies between symbols in the input sequence.

Dense layer

is the regular deeply connected neural network layer. It is most common and frequently used layer. Dense layer does the below operation on the input and return the output.

output = activation(dot(input, kernel) + bias)

where

input : represent the input data

kernel : represent the weight data

dot : represent numpy dot product of all input and its corresponding weights

bias : represent a biased value used in machine learning to optimize the model

activation : represent the activation function which is relu in our model.

3.3 TENSORFLOW

3.3.1 What is tensorflow ?

We wanted TensorFlow is an open source library for numerical computation and large-scale machine learning. TensorFlow bundles together a slew of machine learning and deep learning (aka neural networking) models and algorithms and makes them useful by way of a common metaphor. It uses Python to provide a convenient front-end API for building applications with the framework, while executing those applications in high-performance C++.

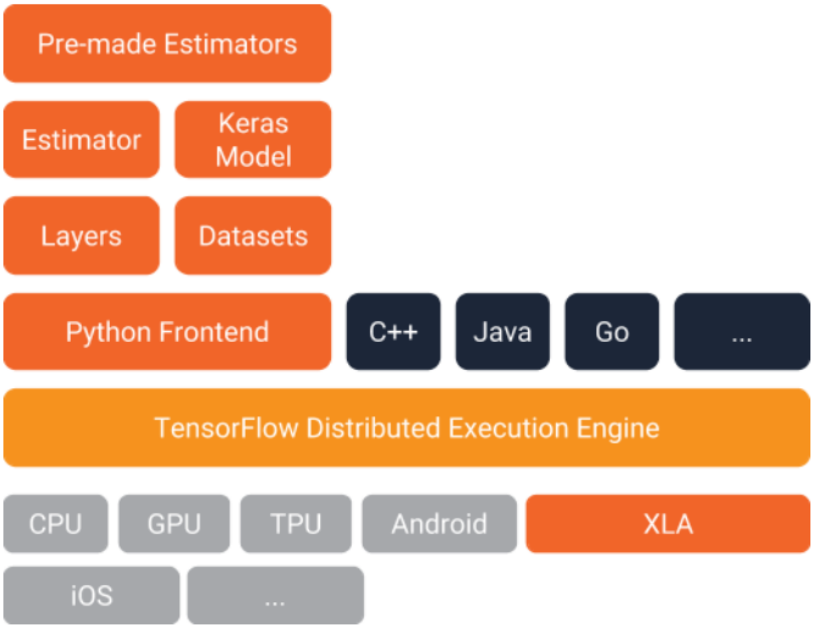
TensorFlow can train and run deep neural networks for handwritten digit classification, image recognition, word embeddings, recurrent neural networks, sequence-to-sequence models for machine translation, natural language processing, and PDE (partial differential equation) based simulations

TensorFlow is a dataflow programming framework at its core. It makes use of a variety of optimization approaches to make calculating mathematical expressions easier and faster.

Some of the key features of TensorFlow are :

- Works quickly with multi-dimensional arrays and mathematical equations.
- Deep neural networks and machine learning ideas are well supported.
- Computing on both GPUs and CPUs, where the same code may run on both architectures
- High computational scalability between devices and large data sets

TENSORFLOW GENERAL ARCHTICTURE



The TensorFlow runtime is a library that runs on any platform. The system architecture that allows for this scale combination to be adjustable. TensorFlow programming features like as the computation graph, operations, and sessions are familiar to us.

To comprehend TensorFlow architecture, some words must first be comprehended. TensorFlow Servable, Servable Streams, TensorFlow Models, and Loade are the terminologies used.

3.3.2 What is tensorflow used for?

Tensorflow is used for multiple thing so we are going to break them apart to make it easy to understand.

3.3.2.1 Voice/Sound Recognition

Sound-based applications are one of TensorFlow's most well-known applications. Neural networks can recognise audio signals if they are fed the right data. These can be anything:

- Voice recognition – mostly used in IoT, Automotive, Security and UX/UI
- Voice search – mostly used in Telecoms, Handset Manufacturers
- Sentiment Analysis – mostly used in CRM
- Flaw Detection (engine noise) – mostly used in Automotive and Aviation

3.3.2.2 Text based applications

Language Detection is one of the most popular uses of text based applications.

- We all know Google Translate, which supports over 100 languages translating from one to another. The evolved versions can be used for many cases like translating jargon legalese in contracts into plain language.

- Text Summarization

3.3.2.3 Time series

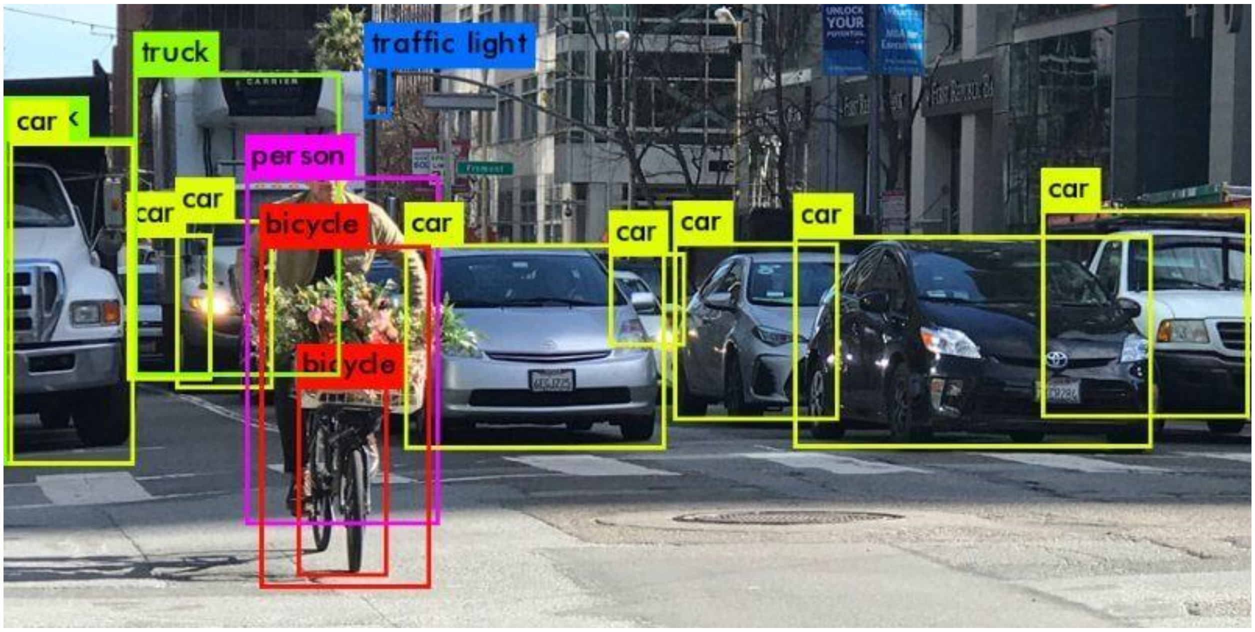
Recommendation is the most popular use case for Time Series. You've definitely heard of companies like Amazon, Google, Facebook, and Netflix that monitor client behaviour and compare it to the activity of millions of other users to figure out what the customer would want to buy or watch. These suggestions are becoming increasingly intelligent; for example, they may propose you particular items as gifts (not for yourself).

3.3.2.4 Video detection

Video data is also handled by TensorFlow neural networks. This is mostly utilised in Motion Detection and Real-Time Thread Detection in Gaming, Security, Airports, and UX/UI. Universities have recently begun to work on large-scale video classification datasets, such as YouTube-8M, in order to speed up research on large-scale video understanding and representation learning. noisy data modeling, transfer learning, and domain adaptation approaches for video.

3.3.2.5 Image recognition

video. Object recognition algorithms based on TensorFlow classify and recognise arbitrary things in larger photos. This is commonly used in engineering applications to recognise shapes for modelling (3D space building from 2D photos) and by social media platforms for photo tagging (Facebook's Deep Face). The technology can learn to identify a tree by studying thousands of photographs of trees, for example.



3.3.3 Data flow graphs

Advantages of data flow diagram:

- Parallelism
- Distributed execution
- compilation
- portability

Data flow graphs are used to describe computation in TensorFlow. Each edge of the graph represents a multi-dimensional data set (tensor) on which the operations are conducted, and each node represents an instance of a mathematical operation (such as addition, division, or multiplication).

TensorFlow manages computational graphs in which each node represents the instantiation of an operation with zero or more inputs and outputs.

Normal edges transmit data structures (tensors) where the output of one operation might become the input of another, and special edges, which are used to govern dependency between two nodes to define the order of operations where one node waits for another to finish.

3.3.4 Visualizing the Computational Graph with TensorBoard

TensorBoard is a data flow graph analysis visualisation tool. This can be beneficial in learning more about machine learning models.

TensorBoard allows you to see several forms of statistics on the parameters as well as details about the various parts of the computational graph in general. A deep neural network with a large number of nodes is not uncommon. TensorBoard gives developers access to each node and how the computation is carried out during the TensorFlow runtime.

The graph parameter will be passed from the session object created in the training program.

3.3.5 machine learning with tensorflow

TensorFlow will be used to demonstrate a machine learning use case. The first example will be a kNN-based data classification technique, while the second will be a linear regression algorithm.

3.3.5.1 KNN

k-Nearest Neighbors is the first algorithm (kNN). It's a supervised learning algorithm that classifies data against training using distance metrics like Euclidean distance. It's one of the most basic algorithms, but it's still very effective in classifying data.

- Advantages of this algorithm include:

When the training model is large enough, it gives excellent accuracy, isn't sensitive to outliers, and we don't need to make any data assumptions.

- The disadvantages of this algorithm include:

New classified data must be uploaded to all initial training instances, which is computationally intensive and requires a lot of memory.

3.3.5.2 Regression analysis

A linear relationship between two variables is sought via the linear regression procedure. We're trying to estimate the parameters of the function y = Wx + b if the dependent variable is y and the independent variable is x.

In the realm of applied sciences, linear regression is a commonly used approach. This approach provides for the implementation of two essential machine learning concepts: the cost function and the gradient descent method for locating the function's minimum.

A machine learning system that uses this method must predict values of y as a function of x, with values W and b determined using a linear regression algorithm, which are truly unknowns and are determined throughout the training process. The cost function is selected, and where the gradient descent is the optimization approach used to obtain a local minimum of the cost function, the mean square error is commonly used

Although the gradient descent approach only finds a local function minimum, it can be used to find a global minimum by selecting a new start point at random after finding a local minimum and repeating the procedure multiple times. If the function's number of minima is limited and there are a large number of attempts, Then there's a good possibility that the global minimum will be discovered at some point.

3.3.6 XOR

XOR is an exclusive or (exclusive disjunction) logical operation that outputs true only when inputs differ.

3.3.6.1 How to build a neural network on Tensorflow for XOR

- For a demonstration, utilise the XOR function with two inputs and one output.
- To create a mathematical representation of a neural network, use tensors.
- Python is a basic and widely used programming language.
- make the code as easy to understand as feasible
- There are several ways to train the neural network.

3.4 KERAS

Keras is one of python's libraries to implement deep learning, it is a high level neural networks API , written in phython and it was developed with a focus on enabling fast experimentation.

Keras is:

- Simple : But not in a basic way. Keras relieves developer cognitive load, allowing you to focus on the most important aspects of the problem.

- Flexible : Keras follows the notion of incremental complexity disclosure: simple workflows should be rapid and easy, while arbitrarily complicated workflows should be accessible via a clear interface.

- Powerful : Keras provides industry-leading performance and scalability, and it is utilised by NASA, YouTube, and Waymo, among others. Allows for easy and fast prototyping (through user friendliness, modularity, and extensibility). Three sorts of inputs are accepted by Keras models: Like Scikit-Learn and many other Python-based packages, NumPy arrays. If your data fits in memory, this is a good alternative. Dataset objects in TensorFlow. This is a high-performance alternative that is better suited to datasets that are too large to store in memory and are streamed from disc or a distributed filesystem. Python generators that produce a large number of files Runs seamlessly on CPU and GPU.

3.4.1 Why choose Keras

3.4.1.1 Keras prioritizes developer experience

Keras is a human-centric API, not a machine-centric one. Keras adheres to best practises for lowering cognitive load, such as providing consistent and straightforward APIs, limiting the number of user activities necessary for common use cases, and providing clear and responsive feedback in the event of a user error.

Keras is simple to learn and use as a result of this. You are more productive as a Keras user, letting you to explore more things.

Keras ranked as #1 for deep learning both among primary frameworks and among all frameworks used.

3.4.1.2 Keras has broad adoption in the industry and the research community

Keras has a large user base in both the industry and the research community, with over 400,000 users as of early 2021. Keras has higher acceptance in every vertical than any other deep learning solution, thanks to its partnership with TensorFlow 2

Keras-powered features are already in use at Netflix, Uber, Yelp, Instacart, Zocdoc, Square, and a slew of other companies.

3.4.1.3 Keras makes it easy to turn models into products

our Keras models can be easily deployed across a greater range of platforms than any other deep learning API:

- On server via a Python runtime or a Node.js runtime
- On server via TFX / TF Serving
- In the browser via TF.js
- On Android or iOS via TF Lite or Apple's CoreML
- On Raspberry Pi, on Edge TPU, or another embedded system

3.4.1.4 Keras has strong multi-GPU & distributed training support

Keras is scalable. Using the TensorFlow DistributionStrategy API, which is supported natively by Keras, you easily can run your models on large GPU clusters (up to thousands of devices) or an entire TPU pod, representing over one exaFLOPs of computing power.

Keras also has native support for mixed-precision training on the latest NVIDIA GPUs as well as on TPUs, which can offer up to 2x speedup for training and inference.

For more information, see our guide to multi-GPU & distributed training.

3.4.1.5 Keras is at the nexus of a large ecosystem

Like you, we know firsthand that building and training a model is only one slice of a machine learning workflow. Keras is built for the real world, and in the real world, a successful model begins with data collection and ends with production deployment.

Keras is at the center of a wide ecosystem of tightly-connected projects that together cover every step of the machine learning workflow, in particular:

- Rapid model prototyping with AutoKeras
- Scalable model training in on GCP via TF Cloud
- Hyperparameter tuning with Keras Tuner
- Extra layers, losses, metrics, callbacks... via TensorFlow Addons
- InSference model quantization & pruning with the TF Model Optimization Toolkit
- Model deployment on mobile or on an embedded with TF Lite
- Model deployment in the browser via TF.js
- ...and many more.

3.4.2 Types of models

3.4.2.1 Sequential model

For most deep learning networks that you can build , the sequential model is likely what you will use , it allow you to easily stack sequential layers (and even recurrent layers) of network in order from input to output.

When to use the Sequential model

A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.

A Sequential model is not appropriate when:

- Your model has multiple inputs or multiple outputs
- Any of your layers has multiple inputs or multiple outputs
- You need to do layer sharing
- You want non-linear topology (e.g. a residual connection, a multi-branch model)

3.4.2.2 Functional API

The Keras functional API is a way to create models that are more flexible than the tf.keras.Sequential API. The functional API can handle models with non-linear topology, shared layers, and even multiple inputs or outputs.

The main idea is that a deep learning model is usually a directed acyclic graph (DAG) of layers. So the functional API is a way to build graphs of layers.

4 MESSUREMENTS

4.1 METRICS

Mean Squared Error:

Mean Squared Error (MSE) is an estimator measures the average of squares of the errors. The term error here indicates the difference between the actual value and the estimated value. The equation below indicates the mathematical equation for calculating the mean squared error.

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \widehat{Y_\imath})^2$$

The Y_i here represents the vector of the actual values and, the vector values is denoted as $\widehat{Y_\imath}$. MSE is most likely used when we have vectors with that in fact continuous values. We will be using this metric for comparing the results of the Numerical Analysis models.

4.2 ACCURACY

The Accuracy of a binary classifier is a statistical measure of how well effectively the classifier is able to predict the correct values. It is the percentage of true results amid the total number of samples. It is represented by the below equation.

Accuracy = True Positives / Total number of predictions

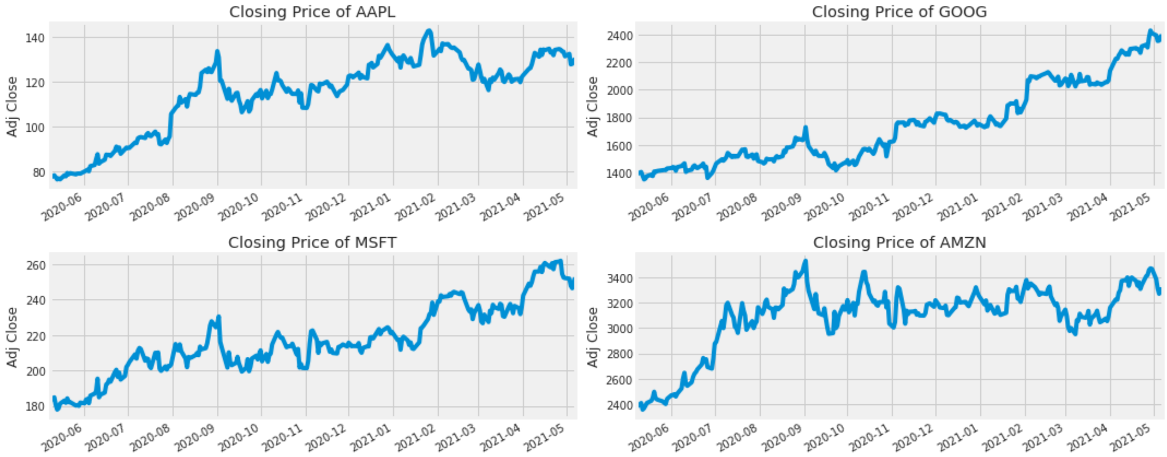
Accuracy is a metric most likely used for classification problems. we converted the problem of predicting exact stock prices to predicting whether the price goes up or goes down, we can use the accuracy as the metric to make an evaluation of the model.

5 STOCK MARKET DATA ANALYSIS

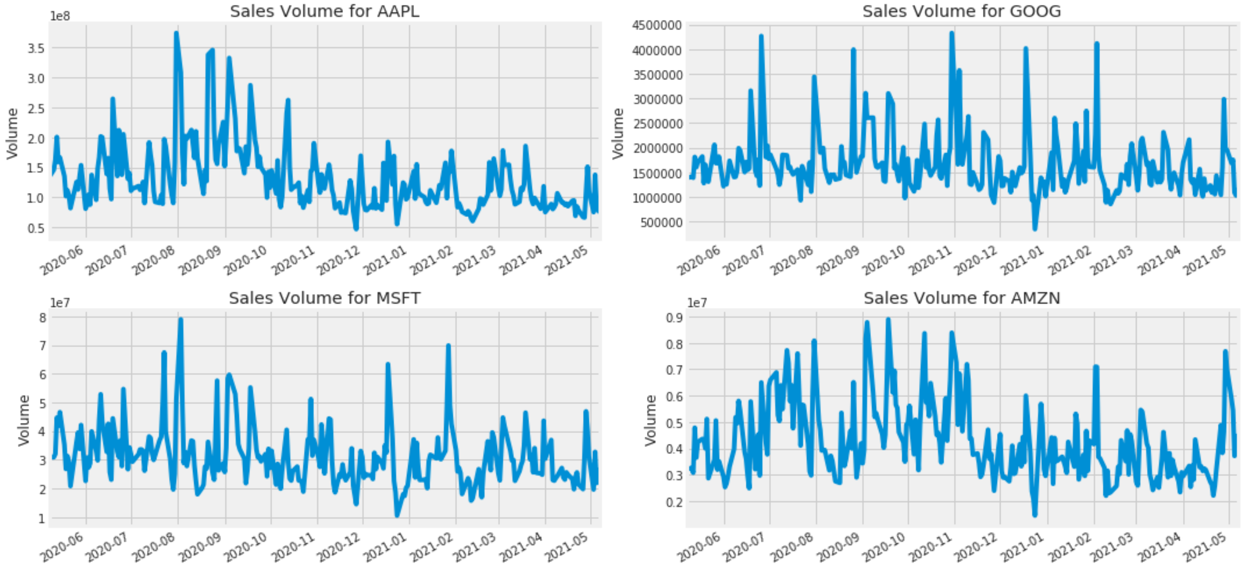
In this section we will be looking at data from the stock market, some technology stocks in particular. We will use pandas to get stock information, and we’re going to visualize different aspects of it, and last but not least we will take look at a few ways of analyzing the risk of a stock, based on its previous performance and history. We will also be predicting future stock prices through something called Long ShortTerm Memory (LSTM) method!

5.1 WHAT WAS THE CHANGE IN PRICE OF THE STOCK OVERTIME?

In this section talk about how to handle requesting stock information with pandas, and how to analyze basic attributes of a stock.

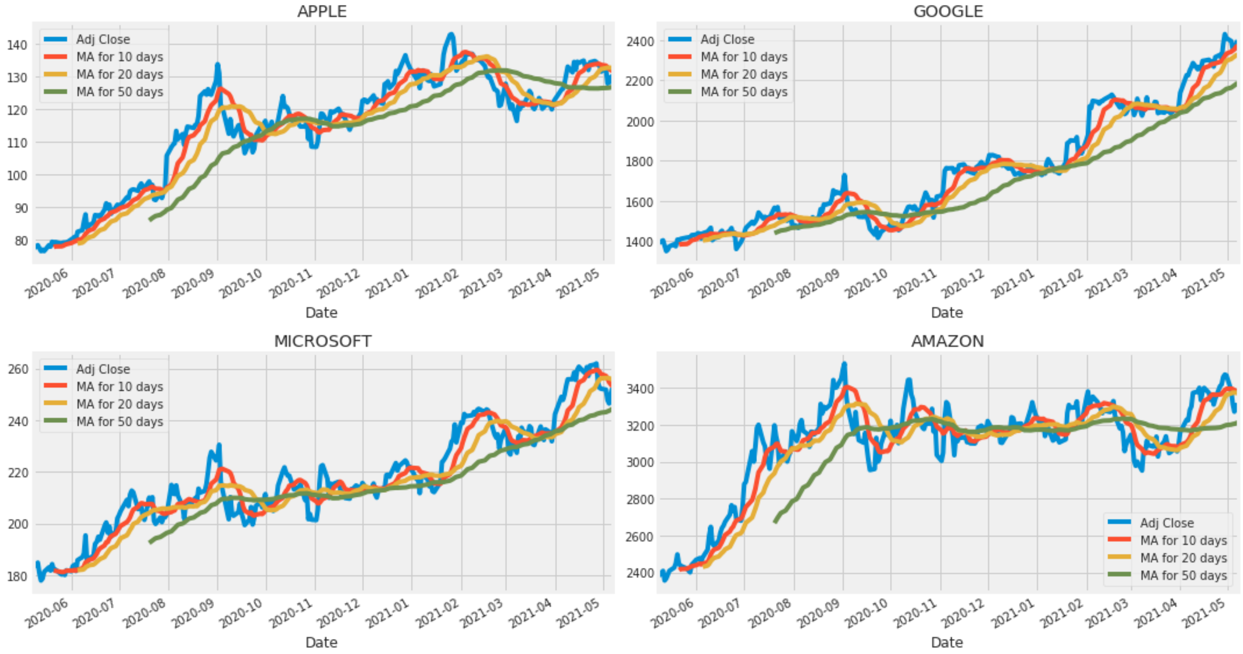


The previous graphs are the historical View of the closing price of the stocks.



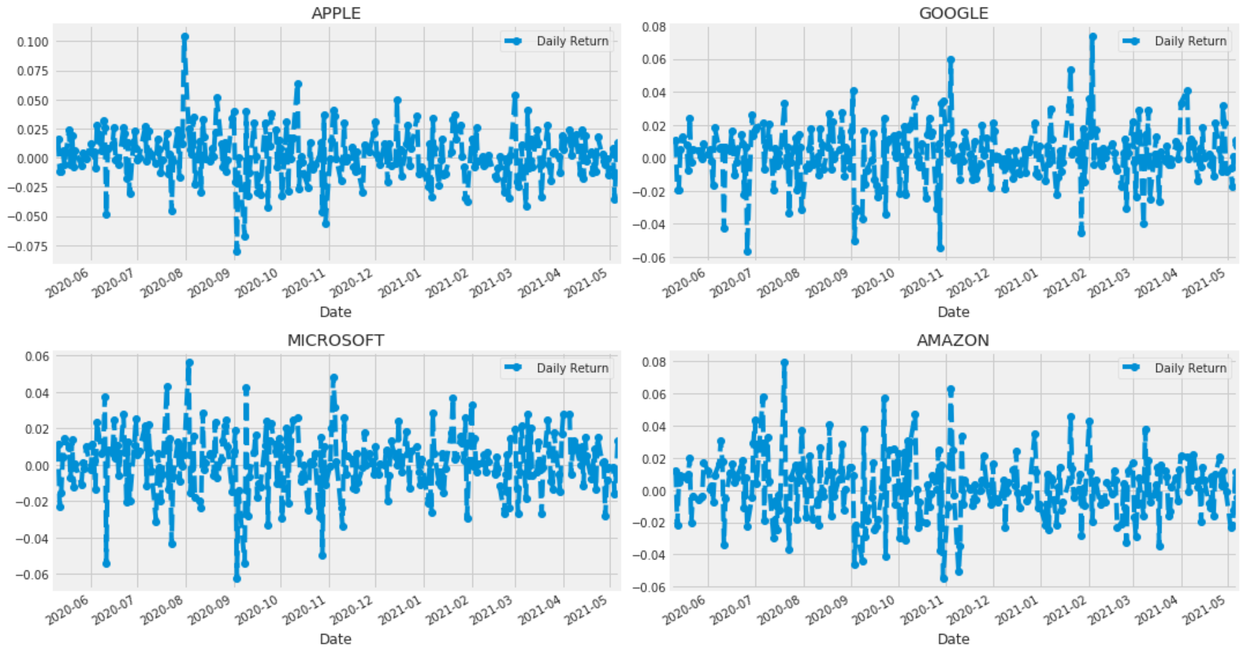
The previous graphs are the volume traded each day of each stock.

5.2 WHAT WAS THE MOVING AVERAGE OF THE VARIOUS STOCKS?

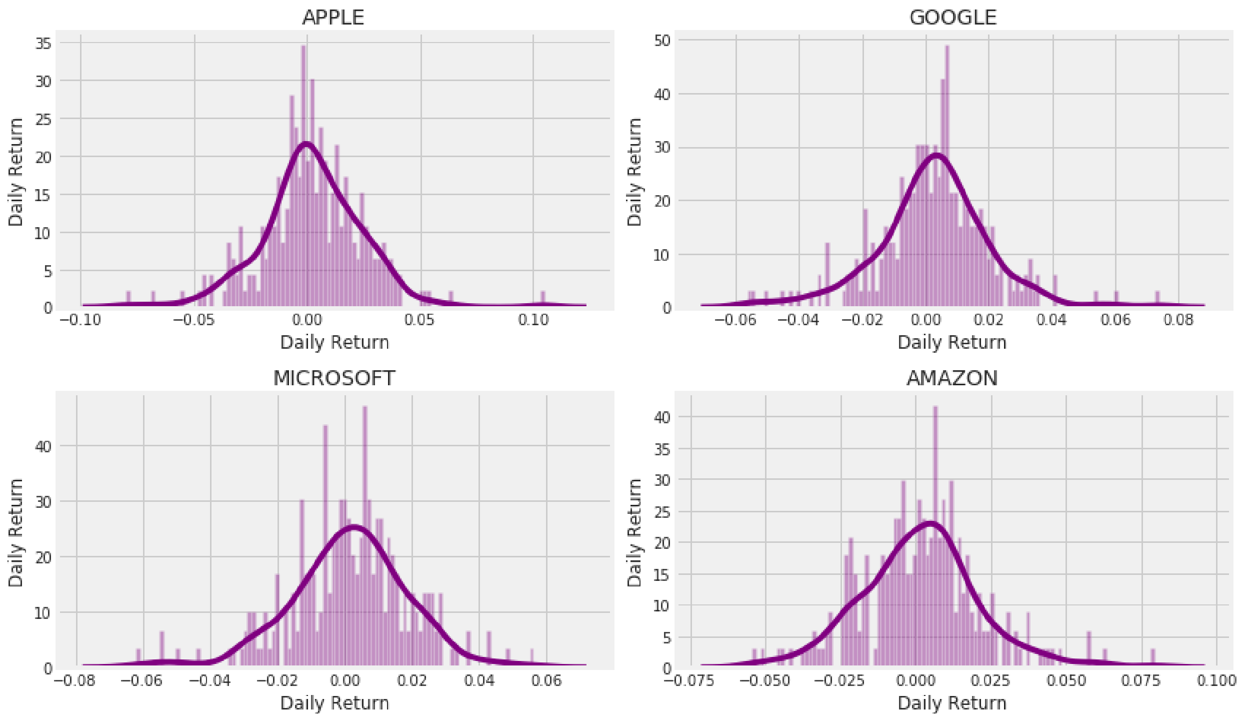


5.3 WHAT WAS THE DAILY RETURN OF THE STOCK ON AVERAGE?

Now that we've done some of the baseline analysis, let's go ahead and dive a little bit deeper. We are now will be going to analyze the risk taken of each stock. To be able to do so we'll be taking a a closer look at the daily changes of the stock, and not only its absolute value.

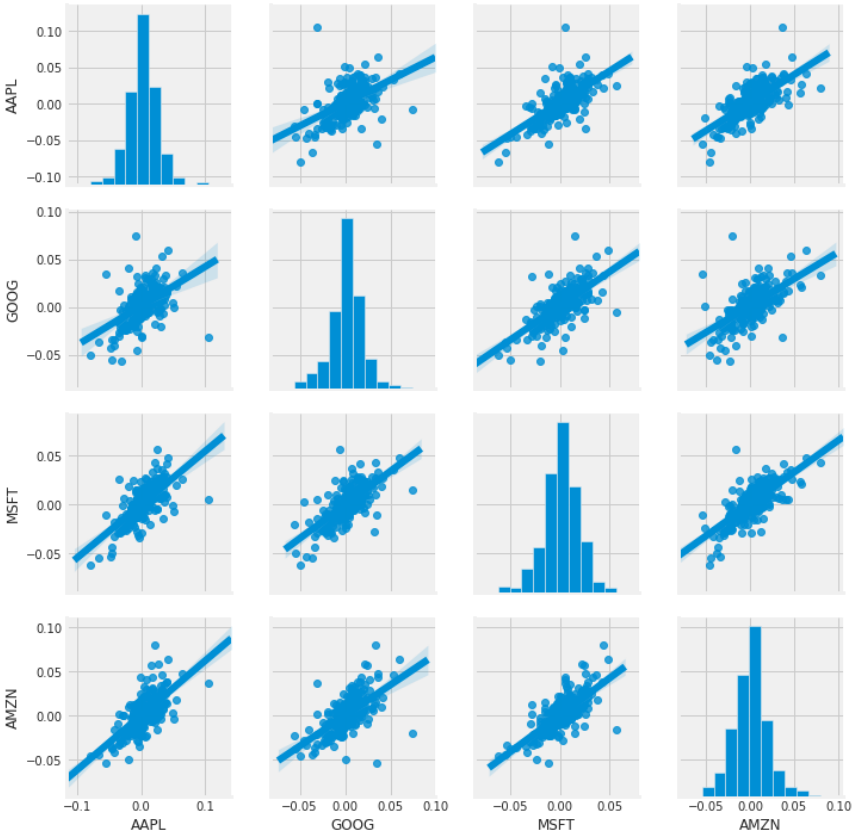


But if we want to gain a higher understanding, we will be taking a look at their histogram.

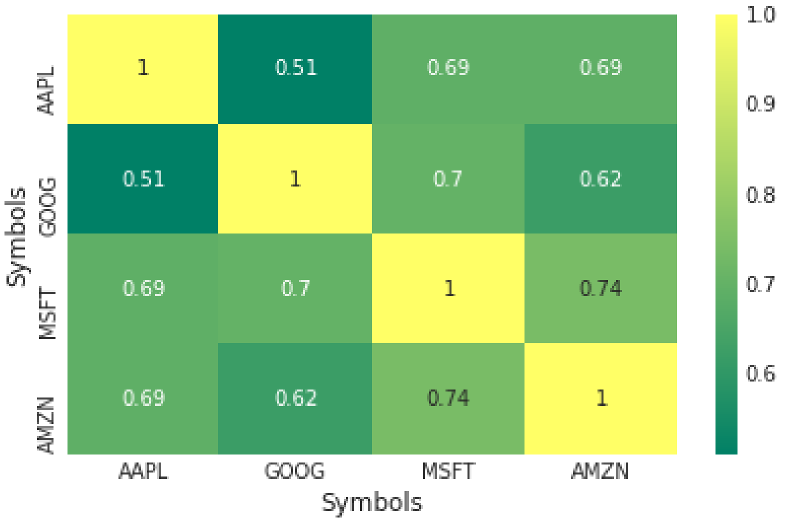


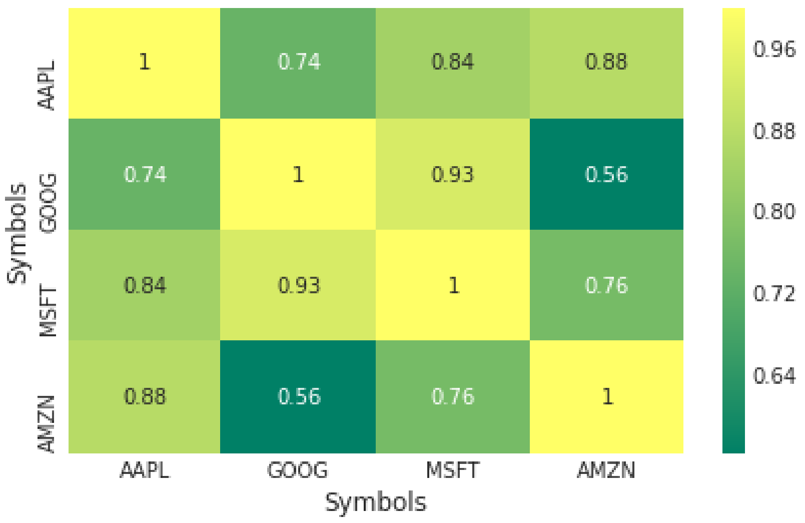
5.4 WHAT WAS THE CORRELATION BETWEEN DIFFERENT STOCKS CLOSING PRICES?

Below the relationships on daily returns between all the stocks can be seen. If we take a quick glance it would show an intriguing correlation between Google and Amazon daily returns.



Also, when we do a correlation plot, to get some actual numerical values for the correlation between the stocks' daily return values. By making a comparison between the closing prices, we would see an intriguing relationship between Microsoft and Apple.

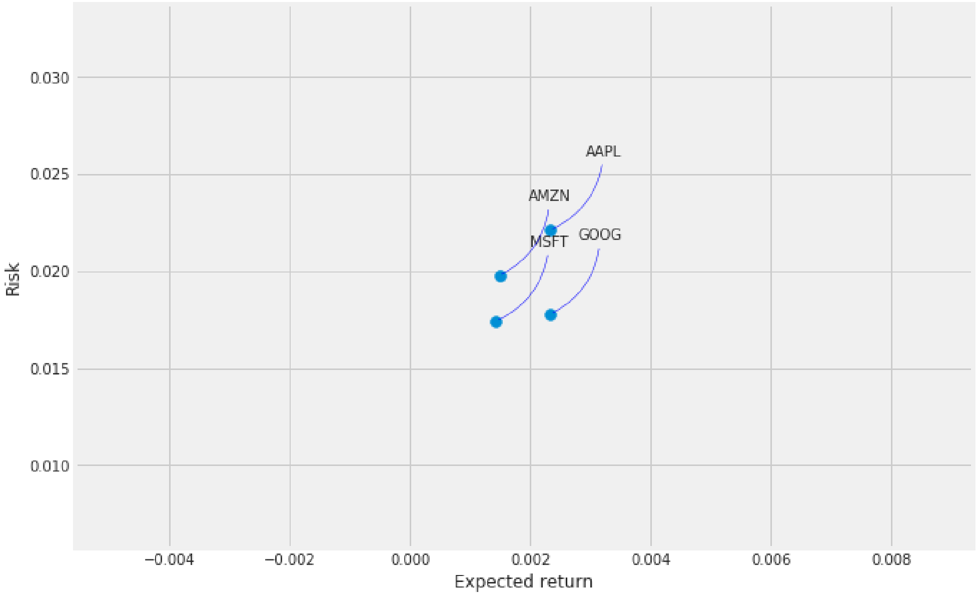




Just like we thought in our PairPlot we can see here numerically and visually that Microsoft and Amazon did have the strongest correlation of daily stock return. It's also intriguing to see that not some, but all the technology companies are positively correlated.

5.5 HOW MUCH VALUE DO WE PUT AT RISK BY INVESTING IN A PARTICULAR STOCK?

There are a lot of ways to quantify risk, and one of the most basic ways is by using the information we have collected on daily percentage returns is by making a comparison between the expected return and the standard deviation of the daily returns.

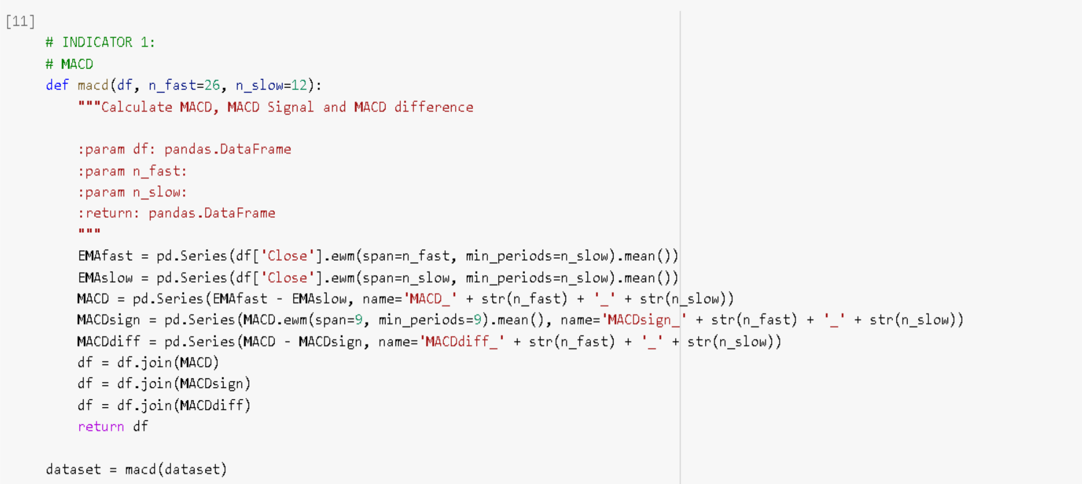


6 EXPRIMENTAL

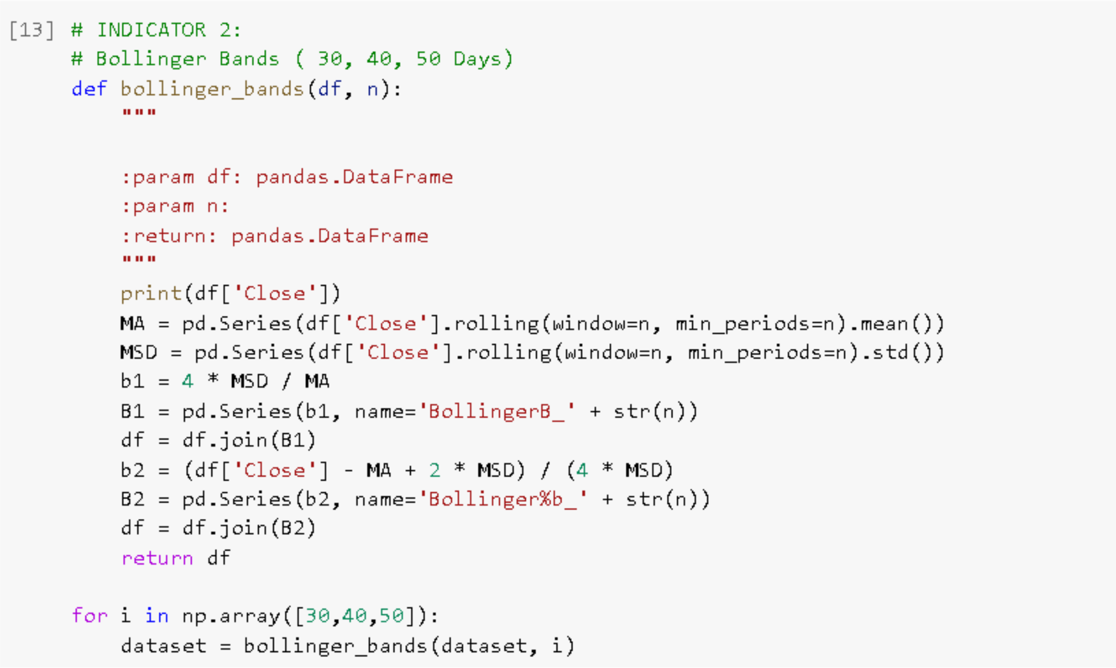
6.1 OSCILLATORS

First from data analysis section we understood which company we would invest in right now with low risk for reward ratio. so we have chosen apple stock at the beginning as well as

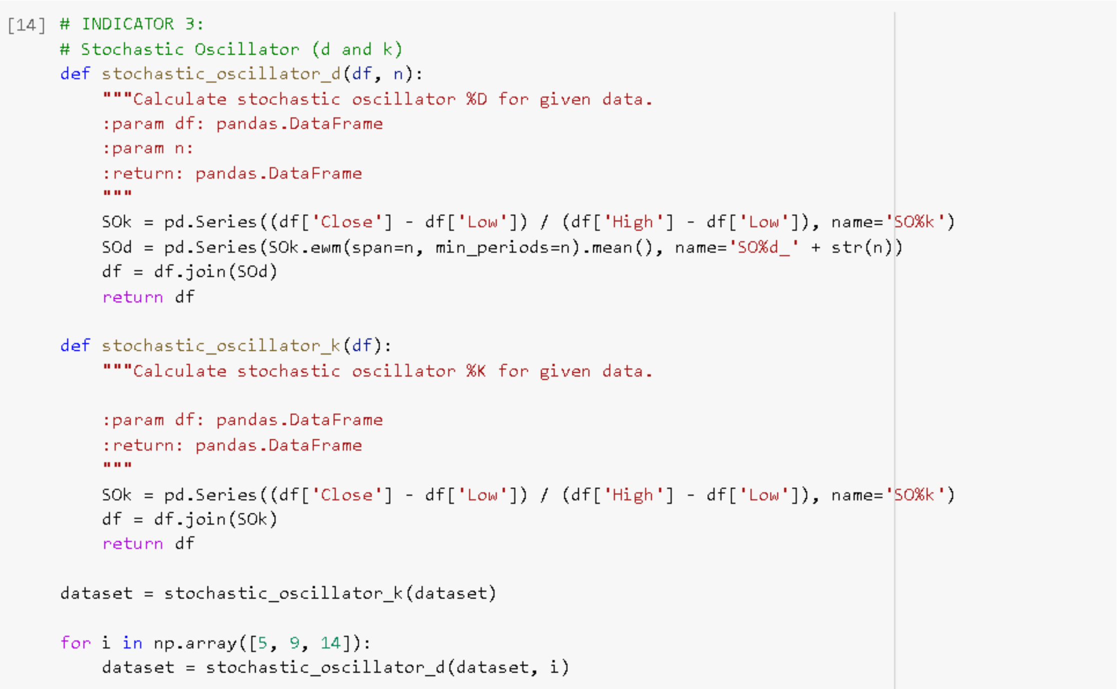
Microsoft, Tesla and google but we choose apple for experiment. We had an issue in getting the daily data That we needed and it was one of our limitations specially indicators and oscillators data. So, we made functions out of the formulas that made the indicators and oscillators itself:



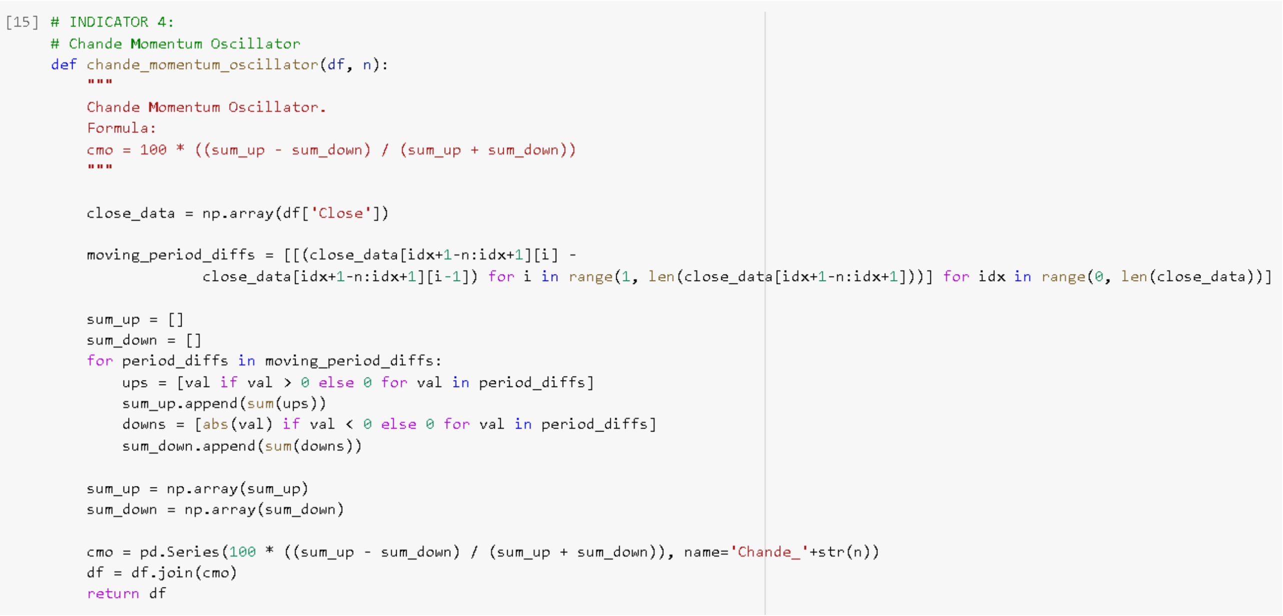
```
[11]
     # INDICATOR 1:
     # MACD
     def macd(df, n_fast=26, n_slow=12):
         """Calculate MACD, MACD Signal and MACD difference

         :param df: pandas.DataFrame
         :param n_fast:
         :param n_slow:
         :return: pandas.DataFrame
         """
         EMAfast = pd.Series(df['Close'].ewm(span=n_fast, min_periods=n_slow).mean())
         EMAslow = pd.Series(df['Close'].ewm(span=n_slow, min_periods=n_slow).mean())
         MACD = pd.Series(EMAfast - EMAslow, name='MACD_' + str(n_fast) + '_' + str(n_slow))
         MACDsign = pd.Series(MACD.ewm(span=9, min_periods=9).mean(), name='MACDsign_' + str(n_fast) + '_' + str(n_slow))
         MACDdiff = pd.Series(MACD - MACDsign, name='MACDdiff_' + str(n_fast) + '_' + str(n_slow))
         df = df.join(MACD)
         df = df.join(MACDsign)
         df = df.join(MACDdiff)
         return df

     dataset = macd(dataset)
```



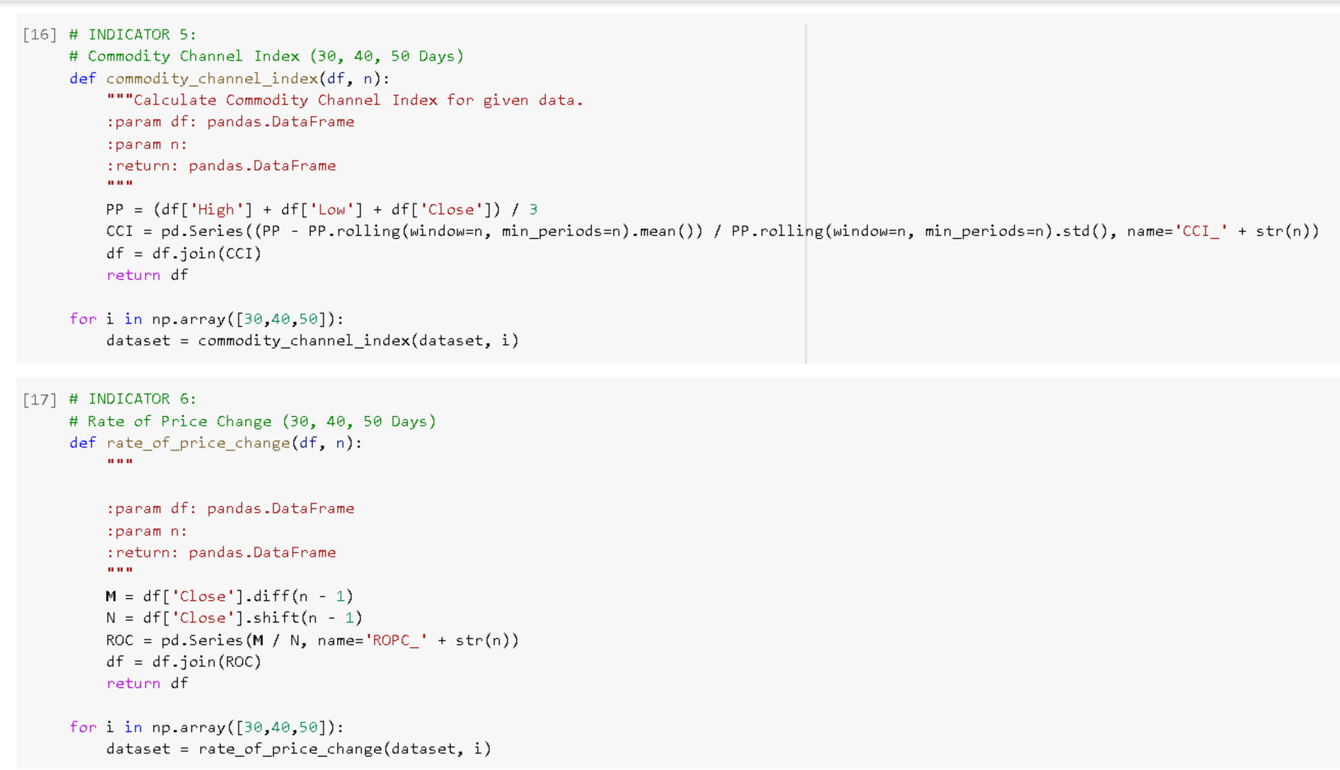
```
[13] # INDICATOR 2:
     # Bollinger Bands ( 30, 40, 50 Days)
     def bollinger_bands(df, n):
         """


         :param df: pandas.DataFrame
         :param n:
         :return: pandas.DataFrame
         """
         print(df['Close'])
         MA = pd.Series(df['Close'].rolling(window=n, min_periods=n).mean())
         MSD = pd.Series(df['Close'].rolling(window=n, min_periods=n).std())
         b1 = 4 * MSD / MA
         B1 = pd.Series(b1, name='BollingerB_' + str(n))
         df = df.join(B1)
         b2 = (df['Close'] - MA + 2 * MSD) / (4 * MSD)
         B2 = pd.Series(b2, name='Bollinger%b_' + str(n))
         df = df.join(B2)
         return df

     for i in np.array([30,40,50]):
         dataset = bollinger_bands(dataset, i)
```



```
[14] # INDICATOR 3:
     # Stochastic Oscillator (d and k)
     def stochastic_oscillator_d(df, n):
         """Calculate stochastic oscillator %D for given data.
         :param df: pandas.DataFrame
         :param n:
         :return: pandas.DataFrame
         """
         SOk = pd.Series((df['Close'] - df['Low']) / (df['High'] - df['Low']), name='SO%k')
         SOd = pd.Series(SOk.ewm(span=n, min_periods=n).mean(), name='SO%d_' + str(n))
         df = df.join(SOd)
         return df

     def stochastic_oscillator_k(df):
         """Calculate stochastic oscillator %K for given data.

         :param df: pandas.DataFrame
         :return: pandas.DataFrame
         """
         SOk = pd.Series((df['Close'] - df['Low']) / (df['High'] - df['Low']), name='SO%k')
         df = df.join(SOk)
         return df

     dataset = stochastic_oscillator_k(dataset)

     for i in np.array([5, 9, 14]):
         dataset = stochastic_oscillator_d(dataset, i)
```

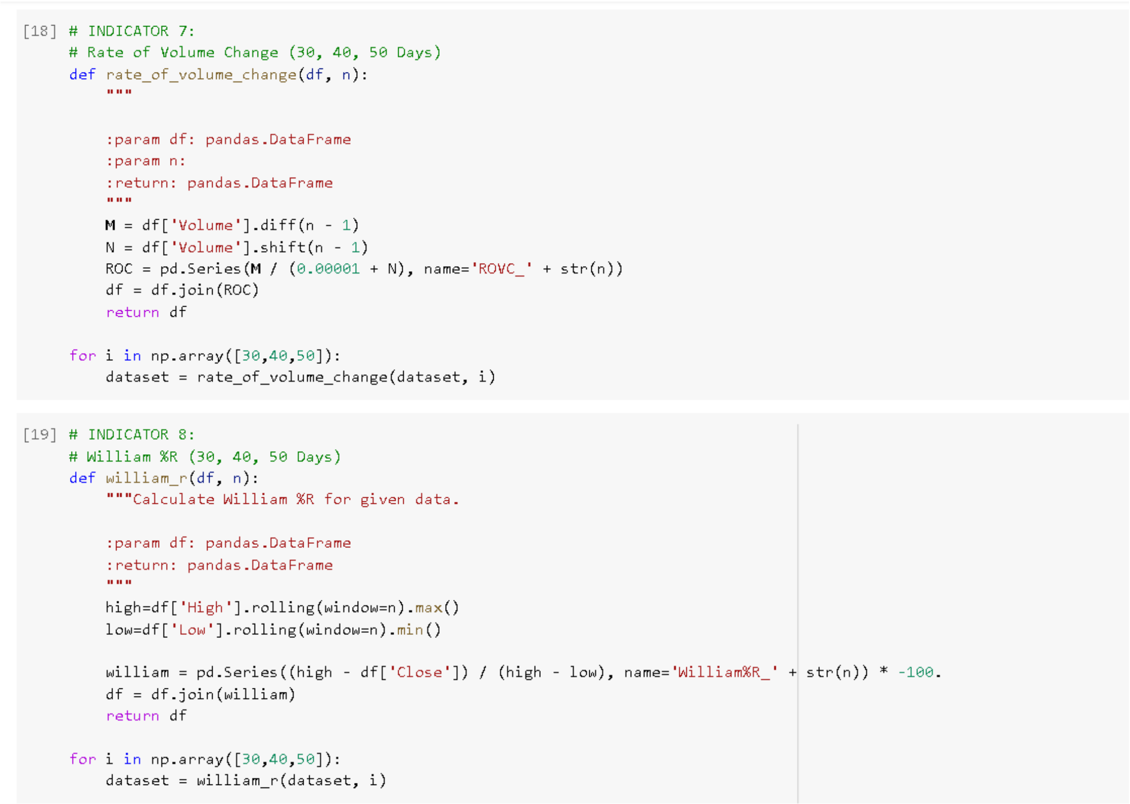


```
[15] # INDICATOR 4:
     # Chande Momentum Oscillator
     def chande_momentum_oscillator(df, n):
         """
         Chande Momentum Oscillator.
         Formula:
         cmo = 100 * ((sum_up - sum_down) / (sum_up + sum_down))
         """

         close_data = np.array(df['Close'])

         moving_period_diffs = [[(close_data[idx+1-n:idx+1][i] -
                          close_data[idx+1-n:idx+1][i-1]) for i in range(1, len(close_data[idx+1-n:idx+1]))] for idx in range(0, len(close_data))]

         sum_up = []
         sum_down = []
         for period_diffs in moving_period_diffs:
             ups = [val if val > 0 else 0 for val in period_diffs]
             sum_up.append(sum(ups))
             downs = [abs(val) if val < 0 else 0 for val in period_diffs]
             sum_down.append(sum(downs))

         sum_up = np.array(sum_up)
         sum_down = np.array(sum_down)

         cmo = pd.Series(100 * ((sum_up - sum_down) / (sum_up + sum_down)), name='Chande_'+str(n))
         df = df.join(cmo)
         return df
```



```
[16] # INDICATOR 5:
     # Commodity Channel Index (30, 40, 50 Days)
     def commodity_channel_index(df, n):
         """Calculate Commodity Channel Index for given data.
         :param df: pandas.DataFrame
         :param n:
         :return: pandas.DataFrame
         """
         PP = (df['High'] + df['Low'] + df['Close']) / 3
         CCI = pd.Series((PP - PP.rolling(window=n, min_periods=n).mean()) / PP.rolling(window=n, min_periods=n).std(), name='CCI_' + str(n))
         df = df.join(CCI)
         return df

     for i in np.array([30,40,50]):
         dataset = commodity_channel_index(dataset, i)
```

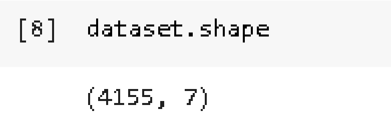
```
[17] # INDICATOR 6:
     # Rate of Price Change (30, 40, 50 Days)
     def rate_of_price_change(df, n):
         """

         :param df: pandas.DataFrame
         :param n:
         :return: pandas.DataFrame
         """
         M = df['Close'].diff(n - 1)
         N = df['Close'].shift(n - 1)
         ROC = pd.Series(M / N, name='ROPC_' + str(n))
         df = df.join(ROC)
         return df

     for i in np.array([30,40,50]):
         dataset = rate_of_price_change(dataset, i)
```



```
[18] # INDICATOR 7:
     # Rate of Volume Change (30, 40, 50 Days)
     def rate_of_volume_change(df, n):
         """

         :param df: pandas.DataFrame
         :param n:
         :return: pandas.DataFrame
         """
         M = df['Volume'].diff(n - 1)
         N = df['Volume'].shift(n - 1)
         ROC = pd.Series(M / (0.00001 + N), name='ROVC_' + str(n))
         df = df.join(ROC)
         return df

     for i in np.array([30,40,50]):
         dataset = rate_of_volume_change(dataset, i)
```

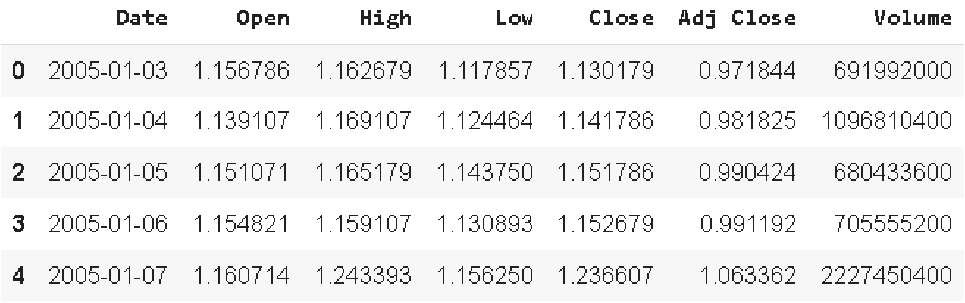
```
[19] # INDICATOR 8:
     # William %R (30, 40, 50 Days)
     def william_r(df, n):
         """Calculate William %R for given data.

         :param df: pandas.DataFrame
         :return: pandas.DataFrame
         """
         high=df['High'].rolling(window=n).max()
         low=df['Low'].rolling(window=n).min()

         william = pd.Series((high - df['Close']) / (high - low), name='William%R_' + str(n)) * -100.
         df = df.join(william)
         return df

     for i in np.array([30,40,50]):
         dataset = william_r(dataset, i)
```

Like that we turned the data from the basic form which is like



```
[8]  dataset.shape

     (4155, 7)
```

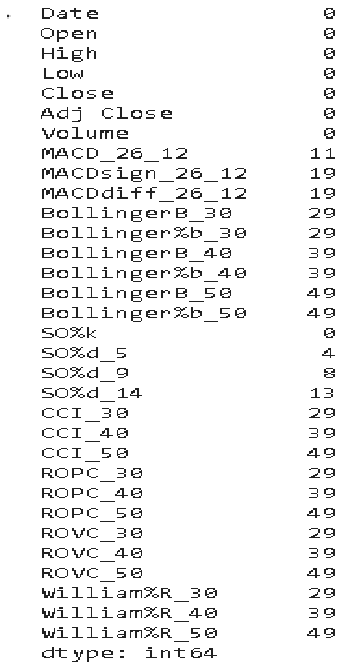
`4155 record and 7 columns`



	Date	Open	High	Low	Close	Adj Close	Volume
0	2005-01-03	1.156786	1.162679	1.117857	1.130179	0.971844	691992000
1	2005-01-04	1.139107	1.169107	1.124464	1.141786	0.981825	1096810400
2	2005-01-05	1.151071	1.165179	1.143750	1.151786	0.990424	680433600
3	2005-01-06	1.154821	1.159107	1.130893	1.152679	0.991192	705555200
4	2005-01-07	1.160714	1.243393	1.156250	1.236607	1.063362	2227450400

Figure 2 implementation Data frame for apple stock since 3/1/2005 till 2021.

To be like that:



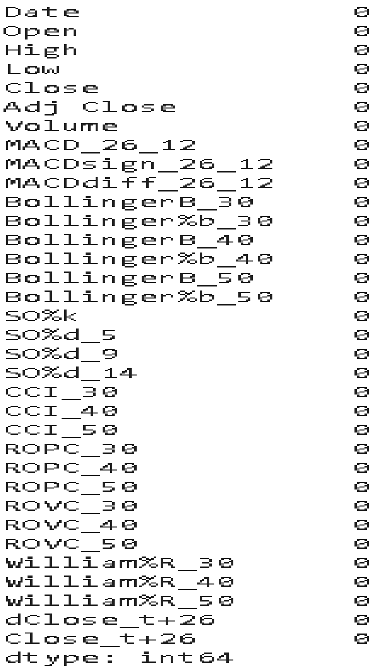
```
Date                 0
Open                 0
High                 0
Low                  0
Close                0
Adj Close            0
Volume               0
MACD_26_12          11
MACDsign_26_12      19
MACDdiff_26_12      19
BollingerB_30       29
Bollinger%b_30      29
BollingerB_40       39
Bollinger%b_40      39
BollingerB_50       49
Bollinger%b_50      49
SO%k                 0
SO%d_5               4
SO%d_9               8
SO%d_14             13
CCI_30              29
CCI_40              39
CCI_50              49
ROPC_30             29
ROPC_40             39
ROPC_50             49
ROVC_30             29
ROVC_40             39
ROVC_50             49
William%R_30        29
William%R_40        39
William%R_50        49
dtype: int64
```

Figure 2 implementation

As there were a lot of data were missing out of functions out put We didn’t want to delete these records as some were in the middle of data so we can’t do it, or we will change the whole data itself logically and specially in figures.

So we took the average for nearest 2 records before and 2 records after the record that has NAN value

hard coding not functionally.



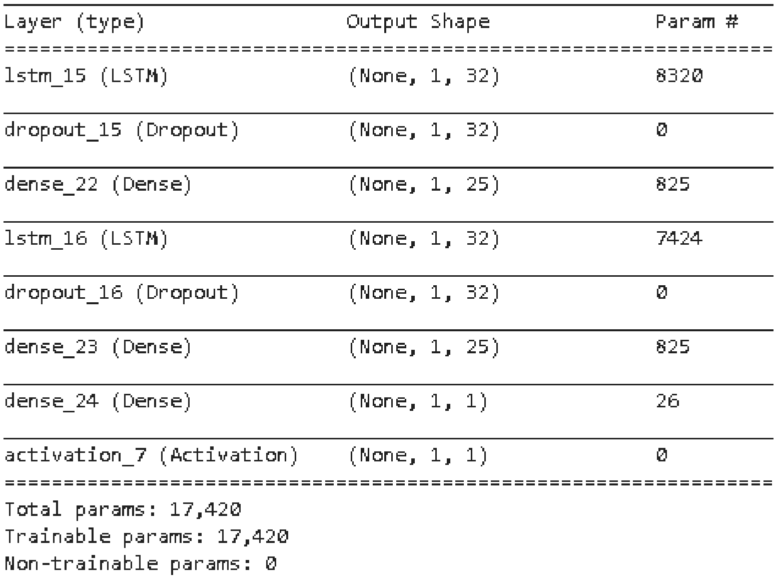
```
Date                  0
Open                  0
High                  0
Low                   0
Close                 0
Adj Close             0
Volume                0
MACD_26_12            0
MACDsign_26_12        0
MACDdiff_26_12        0
BollingerB_30         0
Bollinger%b_30        0
BollingerB_40         0
Bollinger%b_40        0
BollingerB_50         0
Bollinger%b_50        0
SO%k                  0
SO%d_5                0
SO%d_9                0
SO%d_14               0
CCI_30                0
CCI_40                0
CCI_50                0
ROPC_30               0
ROPC_40               0
ROPC_50               0
ROVC_30               0
ROVC_40               0
ROVC_50               0
william%R_30          0
william%R_40          0
william%R_50          0
dClose_t+26           0
Close_t+26            0
dtype: int64
```

Figure 2 implementation.

6.2 BUILDING A MODEL STAGE

First we had a problem to choose the best indicators data that are compatible with each other from technical analysts perspective. As we found out the SMA and EMA which we talked about in section 3.2.7 page 36 These indicator’s data weren’t compatible with other indicator’s data. Even these indicators are used in most of the indicators formulas.

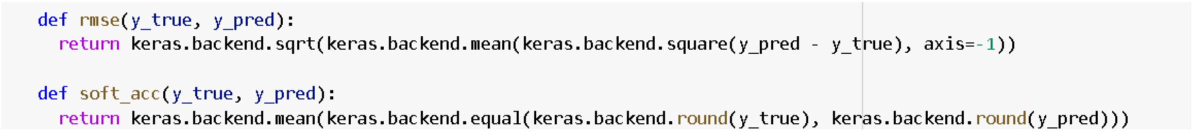
Then using tensor flow / keras to build models. we have built many models that could be related to regression problems and used stock market data to test them. then we tested RNN, ANN and LSTM by making a simple model LSTM had the highest result as its structure suitable for our problem discussed in section 2.2 page 16 using the columns that I got earlier as features in the simple model as shown:



```
Layer (type)                 Output Shape              Param #   
=================================================================
lstm_15 (LSTM)               (None, 1, 32)             8320      
_________________________________________________________________
dropout_15 (Dropout)         (None, 1, 32)             0         
_________________________________________________________________
dense_22 (Dense)             (None, 1, 25)             825       
_________________________________________________________________
lstm_16 (LSTM)               (None, 1, 32)             7424      
_________________________________________________________________
dropout_16 (Dropout)         (None, 1, 32)             0         
_________________________________________________________________
dense_23 (Dense)             (None, 1, 25)             825       
_________________________________________________________________
dense_24 (Dense)             (None, 1, 1)              26        
_________________________________________________________________
activation_7 (Activation)    (None, 1, 1)              0         
=================================================================
Total params: 17,420
Trainable params: 17,420
Non-trainable params: 0
```

Figure 3 implementation .

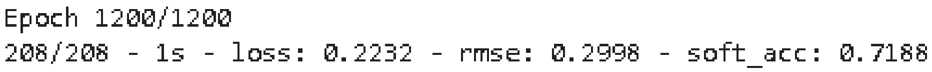
And for each LSTM layer we’ve put dropout layer to overcome over fitting. Our evaluation was using accuracy function and RMSE residual mean squared error:



```
def rmse(y_true, y_pred):
  return keras.backend.sqrt(keras.backend.mean(keras.backend.square(y_pred - y_true), axis=-1))

def soft_acc(y_true, y_pred):
  return keras.backend.mean(keras.backend.equal(keras.backend.round(y_true), keras.backend.round(y_pred)))
```

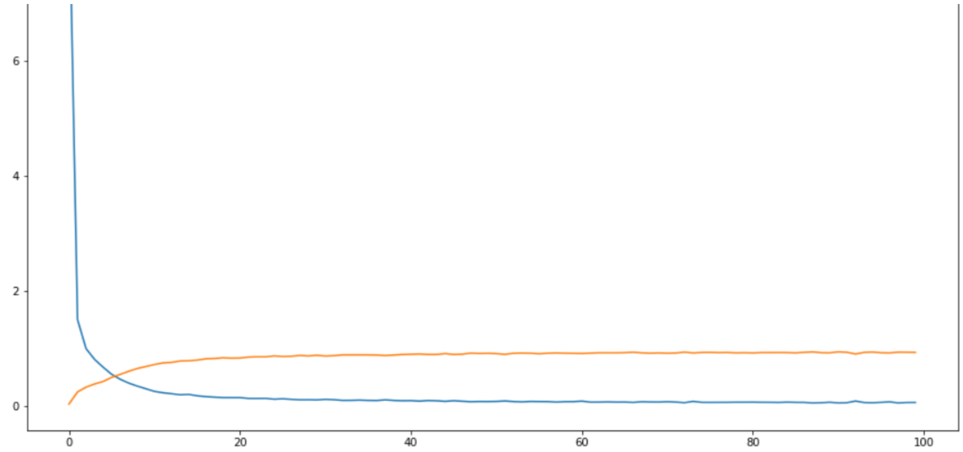
Then after using our concept (the data from different indicators) on the same model it turned to be like that

Then:



```
Epoch 1200/1200
208/208 - 1s - loss: 0.2232 - rmse: 0.2998 - soft_acc: 0.7188
```

and for the graph between: loss functions which is MSE (lower line) and soft accuracy (upper line) in 100 epochs



7 LINKING FRONT-END TO BACK-END USING FLASK

7.1 TERMINOLOGY

7.1.1 Web Framework

Web Application Framework, or simply Web Framework, is a set of libraries and modules that allow web application developers to construct applications without worrying about low-level issues like protocols and thread management.

7.1.2 Flask

Flask is a Python-based web application framework. It was created by Armin Ronacher, who is the leader of Pocco, an international association of Python enthusiasts. The Werkzeug WSGI toolkit and the Jinja2 template engine are the foundations of Flask. Both are Pocco projects.

7.1.3 WSGI

The Web Server Gateway Interface (WSGI) has become the standard for developing Python web applications. It is a specification for a universal interface between the web server and web applications.

7.1.4 Werkzeug

It's a WSGI toolkit that handles requests, response objects, and other utility functions. This makes it possible to construct a web framework on top of it. Werkzeug is one of the foundations of the Flask framework.

7.1.5 Jinja2

Jinja2 is a popular Python templating engine. To render dynamic web pages, a web templating system combines a template with a specific data source.

Flask is also called micro framework. It aims to keep the core of an application simple yet extensible. Flask does not have built-in abstraction layer for database handling, nor does it have form a validation support. Instead, Flask supports the extensions to add such functionality to the application.

7.2 PREREQUISITE

Python 2.6 or higher is usually required for installation of Flask. Although Flask and its dependencies work well with Python 3 (Python 3.3 onwards), many Flask extensions do not support it properly. Hence, it is recommended that Flask should be installed on Python 2.7.

7.3 APPLICATION

FIRST WE Install **virtualenv** for development environment

virtualenv is a virtual Python environment builder. It helps a user to create multiple Python environments side-by-side. Thereby, it can avoid compatibility issues between the different versions of the libraries.

Importing **flask** module in the project is mandatory. An object of **Flask** class is our **WSGI** application.

Flask constructor takes the name of **current module (__name__)** as argument.

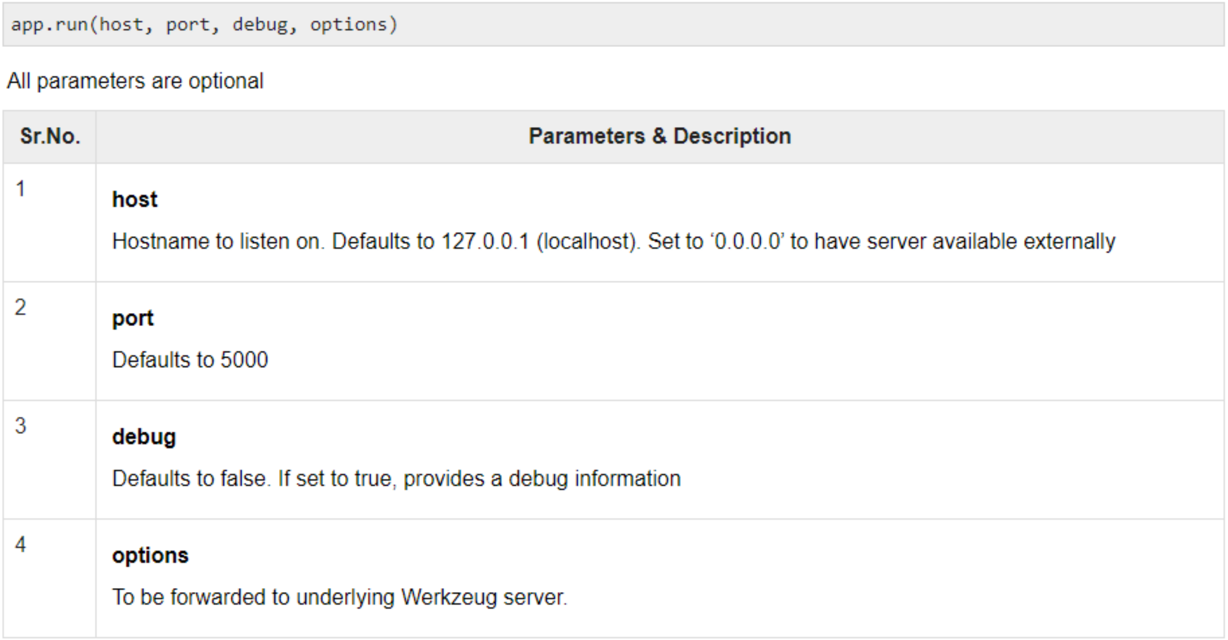
The **route()** function of the Flask class is a decorator, which tells the application which URL should call the associated function.

app.route(rule, options)

The **rule** parameter represents URL binding with the function.

The **options** is a list of parameters to be forwarded to the underlying Rule object.

Finally the **run()** method of Flask class runs the application on the local development server.



```
app.run(host, port, debug, options)
```

All parameters are optional

Sr.No.	Parameters & Description
1	**host** Hostname to listen on. Defaults to 127.0.0.1 (localhost). Set to ‘0.0.0.0’ to have server available externally
2	**port** Defaults to 5000
3	**debug** Defaults to false. If set to true, provides a debug information
4	**options** To be forwarded to underlying Werkzeug server.

The desired Python script is executed from Python shell.

Python file_name.py

A message in Python shell informs you that

*** Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)**

Open the above URL (localhost:5000) in the browser. The output will be displayed on it.

7.4 APPLICATION FLASK

A Flask application is started by calling the **run()** method. while the application is under development, it should be restarted manually for each change in the code. To avoid it enable **debug support**. The server will then reload itself if the code changes. It will also provide a useful debugger to track the errors if any, in the application.

The **Debug** mode is enabled by setting the **debug** property of the application object to **True** before running or passing the debug parameter to the **run()** method.

app.debug = True

app.run()

app.run(debug = True)

7.4.1 Routing

Modern web frameworks use the routing technique to help a user remember application URLs. It is useful to access the desired page directly without having to navigate from the home page.

The **route()** decorator in Flask is used to bind URL to a function.

7.4.2 Variable Rules

It is possible to build a URL dynamically by adding variable parts to the rule parameter.

This variable part is marked as **<variable-name>.** It is passed as a keyword argument to the function with which the rule is associated.

The **revision()** function takes up the floating point number as argument.

The URL rules of **Flask** are based on **Werkzeug's** routing module.

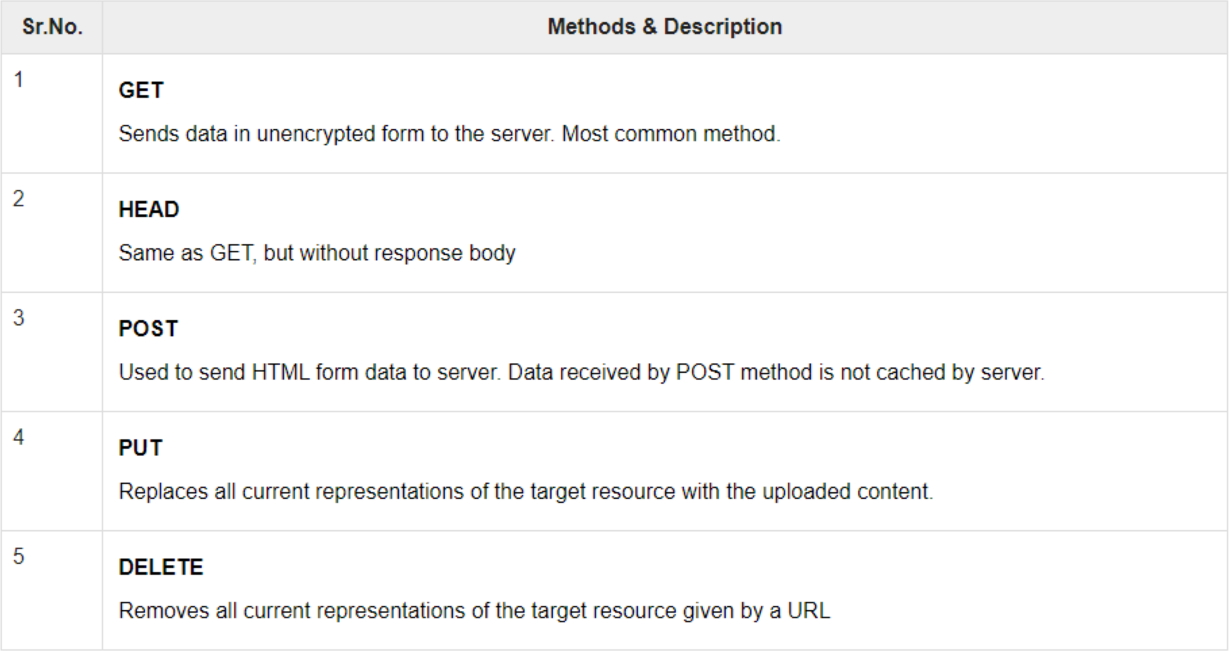
This ensures that the URLs formed are unique and based on precedents laid down by **Apache**.

7.4.3 URL Building

The **url_for()** function is very useful for dynamically building a URL for a specific function. The function accepts the name of a function as first argument, and one or more keyword arguments, each corresponding to the variable part of URL.

7.4.4 HTTP methods

Http protocol is the foundation of data communication in world wide web. Different methods of data retrieval from specified URL are defined in this protocol.



Sr.No.	Methods & Description
1	**GET** Sends data in unencrypted form to the server. Most common method.
2	**HEAD** Same as GET, but without response body
3	**POST** Used to send HTML form data to server. Data received by POST method is not cached by server.
4	**PUT** Replaces all current representations of the target resource with the uploaded content.
5	**DELETE** Removes all current representations of the target resource given by a URL

By default, the Flask route responds to the **GET** requests. However, this preference can be altered by providing methods argument to **route()** decorator.

7.4.5 Templates

It is possible to return the output of a function bound to a certain URL in the form of HTML.

Flask will try to find the HTML file in the **templates** folder, in the same folder in which this script is present.

The term **'web templating system'** refers to designing an HTML script in which the variable data can be inserted dynamically. A web template system comprises of a template engine, some kind of data source and a template processor.

Flask uses **Jinja2** template engine. A web template contains **HTML** syntax interspersed placeholders for variables and expressions which are replaced values when the template is rendered.

The Jinja2 template engine uses the following delimiters for escaping from HTML.

- ✓ {% ... %} for Statements

- ✓ {{ ... }} for Expressions to print to the template output
- ✓ {# ... #} for Comments not included in the template output
- ✓ # ... ## for Line Statements

7.4.6 Static Files

A web application often requires a **static** file such as a **javascript** file or a **CSS** file supporting the display of a web page.

Usually, **the web server** is configured to serve them for us, but during the development, these files are served from static folder in our package or next to our module and it will be available at /static on the application.

A special endpoint **'static'** is used to generate URL for static files.

7.4.7 Request Object

The data from a client's web page is sent to the server as a global request object. In order to process the request data, it should be imported from the Flask module.

Important attributes of request object are listed below −

- ✓ **Form** − It is a dictionary object containing key and value pairs of form parameters and their values.
- ✓ **args** − parsed contents of query string which is part of URL after question mark (?).
- ✓ **Cookies** − dictionary object holding Cookie names and values.
- ✓ **files** − data pertaining to uploaded file.

method − current request method.

7.4.8 Sending Form Data to Template

We have already seen that the http method can be specified in URL rule. The Form data received by the triggered function can collect it in the form of a dictionary object and forward it to a template to render it on a corresponding web page.

7.4.9 Cookies

A cookie is stored on a client's computer in the form of a text file. Its purpose is to remember and track data pertaining to a client's usage for better visitor experience and site statistics.

A Request object contains a cookie's attribute. It is a dictionary object of all the cookie variables and their corresponding values, a client has transmitted. In addition to it, a cookie also stores its expiry time, path and domain name of the site.

In Flask, cookies are set on response object. Use **make_response()** function to get response object from return value of a view function. After that, use the **set_cookie()** function of response object to store a cookie.

Reading back a cookie is easy. The **get()** method of **request.cookies** attribute is used to read a cookie.

7.4.10 Sessions

Session data is stored on client. Session is the time interval when a client logs into a server and logs out of it. The data, which is needed to be held across this session, is stored in the client browser.

A session with each client is assigned a **Session ID**. The Session data is stored on top of cookies and the server signs them cryptographically. For this encryption, a Flask application needs a defined **SECRET_KEY**.

Session object is also a dictionary object containing key-value pairs of session variables and associated values.

To release a session variable use **pop()** method.

7.5 REDIRECT & ERRORS

Flask class has a **redirect()** function. It returns a response object and redirects the user to another target location with specified status code.

Prototype of **redirect()** function is as below –

Flask.redirect(location, statuscode, response)

- ✓ **location** parameter is the URL where response should be redirected.
- ✓ **statuscode** sent to browser's header, defaults to 302.
- ✓ **response** parameter is used to instantiate response.

The following status codes are standardized –

- ✓ HTTP_300_MULTIPLE_CHOICES
- ✓ HTTP_301_MOVED_PERMANENTLY
- ✓ HTTP_302_FOUND
- ✓ HTTP_303_SEE_OTHER
- ✓ HTTP_304_NOT_MODIFIED
- ✓ HTTP_305_USE_PROXY
- ✓ HTTP_306_RESERVED
- ✓ HTTP_307_TEMPORARY_REDIRECT

The default status code is 302, which is for '**found'**.

Flask class has **abort()** function with an error code.

Flask.abort(code)

The **Code** parameter takes one of following values –

- ✓ 400 – for Bad Request
- ✓ 401 – for Unauthenticated
- ✓ 403 – for Forbidden
- ✓ 404 – for Not Found
- ✓ 406 – for Not Acceptable
- ✓ 415 – for Unsupported Media Type
- ✓ 429 – Too Many Requests

7.5.1 Message Flashing

A good GUI based application provides feedback to a user about the interaction.

Flashing system of Flask framework makes it possible to create a message in one view and render it in a view function called **next**.

A Flask module contains **flash()** method. It passes a message to the next request, which generally is a template.

flash(message, category)

message parameter is the actual message to be flashed.

category parameter is optional. It can be either 'error', 'info' or 'warning'.

In order to remove message from session, template calls **get_flashed_messages().**

get_flashed_messages(with_categories, category_filter)

Both parameters are optional.

with_categories is a tuple if received messages are having category. **category_filter** is useful to display only specific messages.

7.5.2 File Uploading

It needs an HTML form with its enctype attribute set to 'multipart/form-data', posting the file to a URL. The URL handler fetches file from **request.files[]** object and saves it to the desired location.

Each uploaded file is first saved in a temporary location on the server, before it is actually saved to its final location. Name of destination file can be hard-coded or can be obtained from filename property of **request.files[file]** object. However, it is recommended to obtain a secure version of it using the **secure_filename()** function.

7.5.3 Extensions

Some important extensions:

- ✓ **Flask Mail** − provides SMTP interface to Flask application
- ✓ **Flask WTF** − adds rendering and validation of WTForms
- ✓ **Flask SQLAlchemy** − adds SQLAlchemy support to Flask application
- ✓ **Flask Sijax** − Interface for Sijax - Python/jQuery library that makes AJAX easy to use in web applications

7.5.4 Deployment

7.5.4.1 Externally Visible Server

A Flask application on the development server is accessible only on the computer on which the development environment is set up. This is a default behavior, because in debugging mode, a user can execute arbitrary code on the computer.

If **debug** is disabled, the development server on local computer can be made available to the users on network by setting the host name as **‘0.0.0.0’.**

app.run(host = ’0.0.0.0’)

your operating system listens to all public IPs.

7.5.4.2 Deployment

To switch over from a development environment to a full-fledged production environment, an application needs to be deployed on a real web server. Depending upon what you have, there are different options available to deploy a Flask web application.

For small application, you can consider deploying it on any of the following hosted platforms, all of which offer free plan for small application.

- ✓ **Heroku**
- ✓ **dotcloud**
- ✓ **webfaction**

Flask application can be deployed on these cloud platforms. In addition, it is possible to deploy Flask app on Google cloud platform. Localtunnel service allows you to share your application on localhost without messing with DNS and firewall settings.

8 CONCLUSION

8.1 GENRAL COMMENTS

We successfully were able to predict some stocks of the stock market and see whither they had corelation or not using our generated data, and the reason we used generated data and not real time data for test was the inability to gain real time data and we simply didn’t have enough time.

How our model would have behaved with real time data?

We treated the data we generated as real time data. So, it is safe to say that the same methods that we used we used will be able to predict real time stocks prices.

Also our models can easily works with different time intervals as well. We can choose a day intervals, week intervals, even a month intervals. Just by changing the way we process the data we input to a different time intervals we will be able to train a network to predict the prices of stock on multiple time gaps.

However our neural network will not be able to predict sudden spick or change of prices in the stock prices. For example: when a company or their competitors announce their new product. Those kinds of incidents can make the stock price skyrocket or make lose a considerable amount of value.

8.2 LIMITATIONS

As we mentioned before one of the limitations we faced was collecting enough data for the model, and we had to make the model work on data that we generated.

Another limitation we ran into was the computing power. Training a complicated model like ours that uses Recurrent neural networks. We would need powerful GPU in order to speed up the training process. Unfortunately we did not have that much of a computing power and we were not able to make further research with our models.

8.3 FUTURE WORK

We want to test the our methods with more data as the keep on coming. We want to make sure that the results we got it is not a fluke and not just a random event that happened as result of the time period we tested in. We have to test our model with even more data as time goes on and make sure that our model can make a generalization.

And also we want incorporate more libraries into our model. There was a library named plotly that would have made the graphs and plots on our website more interactable. We want to add more features to our website as well, like more stock more stock market analysis and so on and so forth.

REFERENCES

[1] W. J. Wilder. New Concepts in Technical Trading Systems. Trend Research, 1

edition, 1978.

[2] Deep Learning with Python book by François Chollet.

[3] W. J. Wilder. New Concepts in Technical Trading Systems. Trend Research, 1

edition, 1978.

[4] https://www.investopedia.com/terms/m/macd.asp

[5] Machine Learning For Dummies Book by John Mueller and Luca Massaron.

[6] I. Goodfellow, Y. Bengio, and A. Courville. Deep Learning. MIT Press, 2016.

http://www.deeplearningbook.org.

[7] https://www.tutorialspoint.com/flask/flask_quick_guide.html

[8] Stock Price Prediction using Deep Learning by Abhinav Tipirisetty.

[9] Stock market trend prediction using text mining approach by Pegah falinouss.

[10] https://www.mathworks.com/discovery/machine-learning.html#how-it-works

[11] D. Srinivasan, A. Liew, and C. Chang. A neural network short-term load forecaster. Electric Power Systems Research.