

# **STOCK PRICE AND PERFORMANCE PREDICTION**

By

<b>ARJUN BATHLA</b>	<b>18BEC1236</b>
<b>V. NIMAL YUGHAN</b>	<b>18BEC1286</b>
<b>S. SABARISH</b>	<b>18BEC1280</b>

A project report submitted to

**Dr SATHIYA NARAYANAN S**

**SCHOOL OF ELECTRONICS ENGINEERING**

in partial fulfilment of the requirements for the course of

**CSE3505 – FOUNDATIONS OF DATA ANALYTICS**

in

**B. TECH. - ELECTRONICS AND COMMUNICATION  
ENGINEERING**



**VIT<sup>®</sup>**

**Vellore Institute of Technology**

(Deemed to be University under section 3 of UGC Act, 1956)

**Vandalur – Kelambakkam Road**

**Chennai – 600127**

**NOVEMBER 2020**

## **BONAFIDE CERTIFICATE**

Certified that this project report entitled “**Stock Price and Performance Prediction**” is a bonafide work of **Arjun Bathla (18BEC1236), V. Nimal Yughan (18BEC1286) and S. Sabarish (18BEC1280)** who carried out the project work under my supervision and guidance for **CSE3505 – Foundations of Data Analytics.**

**Dr SATHIYA NARAYANAN S**

Assistant Professor (Senior Grade 1)

School of Electronics Engineering (SENSE),

Vellore Institute of Technology (VIT Chennai)

Chennai – 600 127.

## **ABSTRACT**

Stock exchange is one of the main areas of investment in recent times. The risk involved in stock market trading is one reason why it was not that popular. The popularity of stock trading can be increased by developing an efficient algorithm which can predict stock prices and also help the user choose the best portfolio to maximize the returns. This algorithm should work in such a way that it minimizes the risk. In this project, we aim to build a model that predicts the returns on individual securities and portfolios, based on their past performances. This model can be used by investors and even by amateurs, to predict which security or portfolio will perform better, in order for them to invest in the securities respectively. In this model, we use covariance, linear regression for prediction, randomized weight distribution simulations, visualization of stock prices and other useful financial and analytical techniques.

## ACKNOWLEDGEMENT

First and foremost, we would like to express our gratitude to our project guide, **Dr. Sathiya Narayanan S**, School of Electronics Engineering, who made this entire course a smooth ride, considering the hard times of COVID-19. Not only did he help us throughout the project, but he also motivated us to build something practical and for the benefit of the general public.

We thank **Dr. A. Sivasubramanian**, Dean of School of Electronics Engineering, and **Dr. P. Vetrivelan**, Head of the Department, for the support they provided throughout the project.

We would also like to thank **NASSCOM**, for providing us the opportunity to study this exceptional course.

We thank our families, who made it easier for us to focus on the project, even with everything around us falling apart. Their love and support have made everything possible



**ARJUN BATHLA**



**V. NIMAL YUGHAN**



**S. SABARISH**

## TABLE OF CONTENTS

<b>SR. NO.</b>	<b>TITLE</b>	<b>PAGE NO.</b>
	ABSTRACT	3
	ACKNOWLEDGEMENT	4
<b>1</b>	<b>INTRODUCTION</b>	6
1.1	BACKGROUND AND MOTIVATION	6
1.2	PROBLEM STATEMENT AND OBJECTIVES	6
<b>2</b>	<b>IMPLEMENTATION</b>	8
2.1	IMPLEMENTED METHOD	8
2.2	ADVANTAGES	19
2.3	CHALLENGES FACED	20
2.4	CODE	21
<b>3</b>	<b>MAIN RESULTS AND INFERENCES</b>	27
3.1	MAIN RESULTS	27
3.2	INFERENCES	31
<b>4</b>	<b>CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK</b>	35
4.1	CONCLUSION	35
4.2	RECOMMENDATIONS FOR FUTURE WORK	35
	REFERENCES	37

# **CHAPTER 1**

## **INTRODUCTION**

[NOTE: Stock and security will be used interchangeably throughout the report]

### **1.1 BACKGROUND AND MOTIVATION**

The share market is currently one of the major places where investors invest considering the lower interest rates in other areas. But they also contain many uncertainties and risks. This makes it very difficult to guess the returns vs the risk. Either they have to be an expert or look out for financial advisors (who are experts in this field) to guide them. Plus, encouraging investors to invest in the primary market will improve industrial development and consequently the economy. The technology being widespread and the internet being easily available to most of the people, the historical data of securities can be easily accessed by a common man also. But the prediction and analysis of the data requires expertise knowledge. Thus, the main motivation for our project is to use various algorithms for finding the best securities or portfolios to invest in and predict the future price of the securities based on historical data.

### **1.2 PROBLEM STATEMENT AND OBJECTIVES**

In this project, we aim to build a model that analyses the returns of individual securities and gives the best individual securities or portfolios to invest in, based on the inputs given by the user. We take the securities that the user wishes to invest on as input. We are importing the dataset from <https://finance.yahoo.com/> . The adjusted closing price predominantly affects the market value of the stock. Hence, for both analysis and prediction we use only the adjusted closing price of each stock. The project uses logarithmic returns that are calculated using adjusted closing price

feature of the dataset. This project also visually represents the daily variations in the adjusted closing price of each stock for better understanding of the risks associated with the stock. We also aim to make our project user friendly, so that even a common man who has very limited knowledge of the stock market can use it to analyze and predict the best performing stocks, thereby encouraging more investments in the primary market.

## CHAPTER 2

### IMPLEMENTATION

The program for the price and performance prediction of stocks has been implemented in Python 3, since it allows a convenient way to read and gather data from the APIs of many websites with public information. It also makes manipulating datasets and plotting figures for visualization easier. Since it has a wide variety of libraries for machine learning, a good stock price prediction model can also be built.

The program is designed to be user-friendly and has been molded for the needs of the user, hence the outputs are based on the user inputs.

#### 2.1 IMPLEMENTED METHOD

##### LIBRARIES USED:

The program makes use of a number of libraries for different purposes.

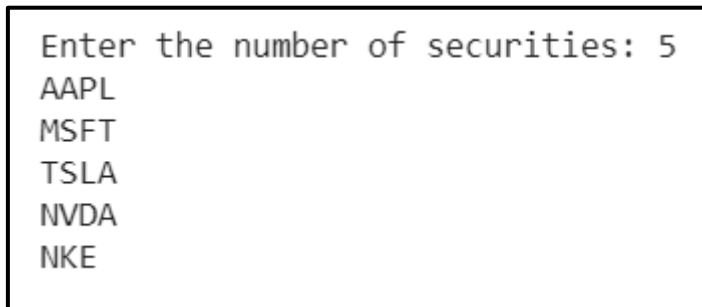
- The **NumPy** <sup>[2]</sup> library is used for mathematical operations such as logarithmic and exponential computations, as well as for manipulation of datasets and array conversions.
- The **pandas** <sup>[3]</sup> library is used to build a dataframe from the raw data obtained from Yahoo Finance.
- From **Scikit-learn** <sup>[4]</sup>, libraries for **LinearRegression**, **DecisionTreeRegressor** and **SVR** (support vector regression) are used to find the most accurate prediction method and then apply that for prediction.



- From **pandas\_datareader**, the **data** module is used to obtain data from the API of Yahoo Finance [1]. Past data for any security for the required time period can be imported using the respective ticker for the security.
- **Matplotlib** [5] is also used for visualization of the past prices, as well as to draw inferences about the returns and risks.

#### ALGORITHM:

- The user is asked to give any number of securities as the input. These securities will undergo various functions for the determination of their performances. Figure 2.1 shows the tickers for the securities used for this implementation. These securities are Apple, Microsoft, Tesla, Nvidia and Nike.



```
Enter the number of securities: 5
AAPL
MSFT
TSLA
NVDA
NKE
```

Figure 2.1: Security tickers

- Next, using DataReader function from the data module, the past prices for these securities are obtained from <https://finance.yahoo.com/> using its API, which doesn't need to be given as a separate argument, since it comes predefined in the data module. The starting date for the past data is set to 01-01-2011. Figure 2.2 shows the values for the first five days for each security. 'High' refers to the highest price reached on a given day and 'Low' refers to the lowest. 'Open' refers to the opening price on a day and 'Close' refers to the closing price on a day. 'Volume' refers to the number

of shares traded on the day. ‘Adj Close’ refers to the adjusted closing prices after paying off all dividends.

AAPL:						
	High	Low	Open	Close	Volume	Adj Close
Date						
2010-12-31	11.552857	11.475357	11.533929	11.520000	193508000.0	9.954883
2011-01-03	11.795000	11.601429	11.630000	11.770357	445138400.0	10.171227
2011-01-04	11.875000	11.719643	11.872857	11.831786	309080800.0	10.224311
2011-01-05	11.940714	11.767858	11.769643	11.928572	255519600.0	10.307948
2011-01-06	11.973214	11.889286	11.954286	11.918928	300428800.0	10.299613
MSFT:						
	High	Low	Open	Close	Volume	Adj Close
Date						
2010-12-31	27.92	27.629999	27.799999	27.91	24752000.0	22.248495
2011-01-03	28.18	27.920000	28.049999	27.98	53443800.0	22.304296
2011-01-04	28.17	27.850000	27.940001	28.09	54405600.0	22.391983
2011-01-05	28.01	27.770000	27.900000	28.00	58998700.0	22.320236
2011-01-06	28.85	27.860001	28.040001	28.82	88026300.0	22.973900
TSLA:						
	High	Low	Open	Close	Volume	Adj Close
Date						
2010-12-31	5.45	5.300	5.314	5.326	7089500.0	5.326
2011-01-03	5.40	5.180	5.368	5.324	6415000.0	5.324
2011-01-04	5.39	5.204	5.332	5.334	5937000.0	5.334
2011-01-05	5.38	5.238	5.296	5.366	7233500.0	5.366
2011-01-06	5.60	5.362	5.366	5.576	10306000.0	5.576
NVDA:						
	High	Low	Open	Close	Volume	Adj Close
Date						
2010-12-31	15.42	14.980000	15.000000	15.40	9781300.0	14.163141
2011-01-03	15.97	15.500000	15.520000	15.82	20436200.0	14.549408
2011-01-04	15.92	15.420000	15.850000	15.77	16284600.0	14.503422
2011-01-05	17.00	15.900000	16.059999	16.98	35705400.0	15.616243
2011-01-06	19.34	17.370001	17.420000	19.33	87332800.0	17.777500
NKE:						
	High	Low	Open	Close	Volume	Adj Close
Date						
2010-12-31	21.459999	21.280001	21.350000	21.355000	5796000.0	16.570024
2011-01-03	21.645000	21.315001	21.457500	21.522499	8566400.0	16.699995
2011-01-04	21.437500	20.937500	21.400000	20.992500	13797600.0	16.288754
2011-01-05	21.207500	20.877501	20.912500	21.129999	11598800.0	16.395445

Figure 2.2: First five observations for each security

- Out of these different variables, the Adjusted Closing Price is the most useful when it comes to analyzing a stock and predicting its future prices. Thus, the ‘Adj. Close’ variables for each security are combined together into one single dataset called ‘sec\_data’, as shown in Figure 2.3. This dataset will be used for all analytical and prediction purposes in the program.

[ ] sec_data					
	AAPL	MSFT	TSLA	NVDA	NKE
2010-12-31	9.954883	22.248495	5.326000	14.163141	16.570024
2011-01-03	10.171227	22.304296	5.324000	14.549408	16.699995
2011-01-04	10.224311	22.391983	5.334000	14.503422	16.288754
2011-01-05	10.307948	22.320236	5.366000	15.616243	16.395445
2011-01-06	10.299613	22.973900	5.576000	17.777500	16.248020
...	...	...	...	...	...
2020-10-26	115.050003	210.080002	420.279999	525.650024	128.369995
2020-10-27	116.599998	213.250000	424.679993	535.869995	127.989998
2020-10-28	111.199997	202.679993	406.019989	505.079987	122.080002
2020-10-29	115.320000	204.720001	410.829987	520.960022	122.860001
2020-10-30	108.860001	202.470001	388.040009	501.359985	120.080002
2476 rows × 5 columns					

Figure 2.3: sec\_data – Adjusted closing prices for each security

The first major part of the program is to analyze a stock’s past performance, and based on that suggest the optimum security/portfolio to the user to invest into. This segment is named ‘STOCK PERFORMANCE’ in the .ipynb file attached in the folder.

- First, using the sec\_data dataset, the daily returns for each security are calculated. There are two methods for doing this – simple returns and

logarithmic returns. Simple returns can be defined as the fraction of price change in a stock price with respect to its price on the previous day. Mathematically,

$$\text{Simple return} = \frac{\text{Current Closing Price} - \text{Previous Closing Price}}{\text{Previous Closing Price}}$$

This formula is useful when the analysis operations involve multiplicative functions, so that a cumulative return for a number of days can be obtained. But since in this program, the returns will undergo additive operations, simple returns are not preferred, and hence logarithmic returns are used. When these returns undergo additive operations, it is equivalent to the multiplication of simple returns.

The logarithmic return <sup>[6]</sup> can be calculated simply by taking the natural log of the ratio of current closing price to previous closing price. Mathematically,

$$\text{Logarithmic return} = \ln\left(\frac{\text{Current Closing Price}}{\text{Previous Closing Price}}\right)$$

The logarithmic returns for each day and each security are calculated and stored in 'sec\_returns'. Figure 2.4 shows the logarithmic returns dataset.

	AAPL	MSFT	TSLA	NVDA	NKE
2020-10-26	0.000087	-0.028854	-0.000832	-0.033596	-0.012541
2020-10-27	0.013382	0.014977	0.010415	0.019256	-0.002965
2020-10-28	-0.047419	-0.050837	-0.044934	-0.059175	-0.047276
2020-10-29	0.036381	0.010015	0.011777	0.030956	0.006369
2020-10-30	-0.057648	-0.011051	-0.057071	-0.038349	-0.022887

Figure 2.4: sec\_returns – Daily log returns

- For the perusal of a professional stock analyst, the past prices for each security are plotted. These plots consist of various patterns using which a professional can predict the future performance of a stock. Since this plot is of little use to an amateur, data analysis techniques are performed to make it easier for the common man to compare stocks. Figure 2.5 shows the plot of the past stock prices for each security. For a better understanding, the prices have been normalized to 100, i.e. the prices for each stock have been divided by the first available price for the respective stock and then multiplied by 100.

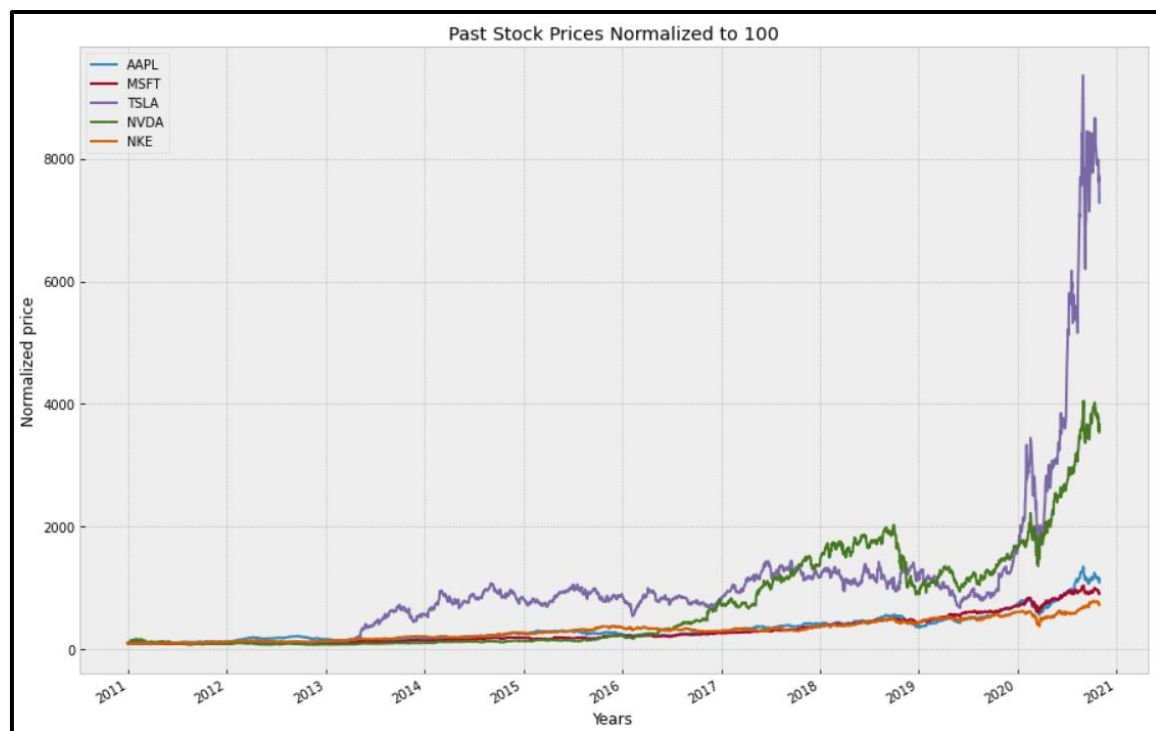


Figure 2.5: Normalized past stock prices

- Using the sec\_returns dataset, the average annual return and risk for each security are calculated. The stock market is open for around 250 days in a year. Thus, the average annual return is simply the mean of all returns for

a security, multiplied by 250. The average annual risk is nothing but the standard deviation of all returns for a security multiplied by the square root of 250. All these values are stored in a dataset.

- To compare performances of securities and find the security with the best past performance, the main aim is to maximize the return and minimize the risk. Thus, return-to-risk ratios (RRRs) are calculated for each security. This is simply the average annual return divided by the average annual risk. The security with the highest return-to-risk ratio, if invested in, is the most likely to give good returns at the lowest risks. The program outputs the details of this security.
- When investing in the stock market, the majority of people prefer to put their money on a collection of stocks, instead of a single stock. This group of stocks is called a portfolio. The advantage of investing in a portfolio is that even if you lose money on one security, you might gain on another security, hence the risk is minimized. To minimize the risk to a greater extent, the portfolios are built such that the securities in the portfolio have the least correlation. This implies that if one stock goes down, the chances of the other stock going down are low. Thus, next we focus on optimizing a portfolio.
- First, a list of all possible combinations for a portfolio is made. This includes portfolios with two, three or even more securities. Figure 2.6 shows a list of possible portfolios for the securities analyzed in this implementation.
- For each portfolio in the list, 1000 iterations of randomized weight distribution are performed. In each iteration, random weights are generated for each security in the portfolio, and their sum is 1. These are essentially the ratios of investment amount in each security.
- For each weight combination, the estimated annual return is calculated by computing the dot product of the weights with the respective average annual returns for the securities in the portfolio.

- For each weight combination, the estimated annual risk is calculated by computing the square root of the matrix multiplication of (weights, covariance matrix of daily returns of securities in the portfolio, weights). This is a measure of risk for the portfolio, and is equivalent to the standard deviation of the combination of securities in the portfolio.

```
[('AAPL', 'MSFT'),
 ('AAPL', 'TSLA'),
 ('AAPL', 'NVDA'),
 ('AAPL', 'NKE'),
 ('MSFT', 'TSLA'),
 ('MSFT', 'NVDA'),
 ('MSFT', 'NKE'),
 ('TSLA', 'NVDA'),
 ('TSLA', 'NKE'),
 ('NVDA', 'NKE'),
 ('AAPL', 'MSFT', 'TSLA'),
 ('AAPL', 'MSFT', 'NVDA'),
 ('AAPL', 'MSFT', 'NKE'),
 ('AAPL', 'TSLA', 'NVDA'),
 ('AAPL', 'TSLA', 'NKE'),
 ('AAPL', 'NVDA', 'NKE'),
 ('MSFT', 'TSLA', 'NVDA'),
 ('MSFT', 'TSLA', 'NKE'),
 ('MSFT', 'NVDA', 'NKE'),
 ('TSLA', 'NVDA', 'NKE'),
 ('AAPL', 'MSFT', 'TSLA', 'NVDA'),
 ('AAPL', 'MSFT', 'TSLA', 'NKE'),
 ('AAPL', 'MSFT', 'NVDA', 'NKE'),
 ('AAPL', 'TSLA', 'NVDA', 'NKE'),
 ('MSFT', 'TSLA', 'NVDA', 'NKE'),
 ('AAPL', 'MSFT', 'TSLA', 'NVDA', 'NKE')]
```

Figure 2.6: All possible portfolios

- The return-to-risk ratios are calculated for each iteration, and the details of the iteration with the highest RRR are recorded and displayed. These include the security combination for the highest RRR, the ratios of investment in each security, and the estimated annual return.
- For each portfolio, the returns vs risks for all the 1000 iterations are plotted for inferential purposes.

- There is a special provision if the user is willing to take a high risk, since a high risk also implies a high return rate. All the possible portfolios are coded with numbers starting from 1, which appear at the top of the return vs risk plots. The user is asked to input the respective portfolio code along with the amount of risk they are willing to take, for which they can refer the return vs risk plots, since there are no specific units for risk. For the user-provided inputs, the maximum annual profit is estimated from the 1000 iterations, and the respective ratios for investment in the portfolio are displayed. All this information is extracted from the datasets formed while randomized weight distribution.

Using these ratios, the user can invest in the securities respectively, in order to maximize their profit within the risk they are willing to take.

The second major part of the program is to predict the future prices for a given security. This segment is named ‘STOCK PRICE PREDICTION’ in the .ipynb file attached in the folder.

- First, the user is asked to input the security for which they would like to predict the future stock prices, along with the number of days in the future prediction. Let this number be ‘days’.  
In this implementation, the stock prices of Microsoft (MSFT) will be predicted for the next 250 days. Since the most common time interval for a possible trend in data such as stock prices is a year, and there are 250 days in a “market year”, 250 days prediction will give a somewhat better fit.
- A dataset is formed with two columns, one for the past prices of the security and another with the same values but shifted up by the number of days in prediction, as shown in Figure 2.7.



	Past Prices	Shifted Prices
2010-12-31	22.248495	21.129261
2011-01-03	22.304296	21.292927
2011-01-04	22.391983	21.243826
2011-01-05	22.320236	21.906679
2011-01-06	22.973900	22.422224
...	...	...
2020-10-26	210.080002	NaN
2020-10-27	213.250000	NaN
2020-10-28	202.679993	NaN
2020-10-29	204.720001	NaN
2020-10-30	202.470001	NaN

2476 rows × 2 columns

Figure 2.7: Past and shifted past prices

- Next, two different arrays are obtained from this dataset. One is the feature array (feature dataset for training), which contains the past prices excluding the last ‘days’ days, 250 in this case. The other array is the target array (target dataset for training), which contains the shifted past prices excluding the NaN values. Clearly, the size of these two arrays will be the same. Basically, the price on a given day is being mapped to the price ‘days’ days later. The relationship between these two variables will determine the future estimate for the stock prices. Figure 2.8 and 2.9 show the feature and target arrays respectively.

```
array([[ 22.2484951 ],
       [ 22.30429649],
       [ 22.39198303],
       ...,
       [141.75616455],
       [142.10223389],
       [142.92289734]])
```

Figure 2.8: Feature array

```
array([ 21.12926102, 21.29292679, 21.24382591, ..., 202.67999268,
        204.72000122, 202.47000122])
```

Figure 2.9: Target array

- Now that the feature and target datasets are ready, the prediction model will be trained based on them. The feature and target arrays are both randomly split into training and testing datasets, with testing datasets of size 30%.
- In this program, we compare three different methods of regression – linear regression, decision tree regression and support vector regression. Thus, three different prediction models are trained using the obtained training datasets.
- The coefficients of determination,  $R^2$ , also referred to as confidence intervals [\[7\]](#), are computed for each model. The maximum possible value for this coefficient is 1, which implies the best fit. In practical cases, the value is almost never 1, hence a higher value implies a better fit.
- It was found that linear regression gives the best fit, i.e. the highest confidence interval. Thus, the prediction is performed using linear regression.
- Linear regression computes a linear equation for the best possible fit, minimizing the residual squares. Thus, appropriate values for  $\beta_0$  and  $\beta_1$  are obtained.

- The prices in the last 'days' days are fed to the linear regression model, and based on the values of  $\beta_0$  and  $\beta_1$ , the future values are predicted. Let the input to the model be  $X$  and the respective future prices by  $Y$ . Then  $Y$  is calculated with the simple linear equation:

$$Y = \beta_0 + \beta_1 X$$

- The predicted future prices are also plotted for the convenience of the user.

## 2.2 ADVANTAGES

This model for stock price and performance prediction has various advantages, including but not limited to the following.

- For an amateur who has no prior information about stock trading and trends in a stock, it can be very hard to begin with trading. Moreover, if they invest wrongly, they might face huge losses and debts. Thus, this model provides them with something of an entry into stock trading.
- For someone who does have prior information about trends in stock prices, this program provides a platform for visualization of the prices, for them to observe and infer from.
- Since most of the already existing stock analysis models give importance only to the returns of a security, they overlook its risk which is a big determining factor in the performance of a stock. This model gives equal importance to both returns and risks, and analyzes the stocks based on their respective ratios.
- Investment in a single security holds the highest risks. When investing in a portfolio of different securities, this risk can be minimized. This model

provides an analysis of every possible portfolio that can be built from the provided securities. The portfolio that has the best performance is suggested for investment, along with the ratios for investment in each security in the portfolio.

- Not all investors look for minimal risk. Some big investors are willing to take big risks to maximize their returns. This program can determine the maximum return a portfolio can provide, given a certain risk parameter. It also suggests the ratios for investment in each security in the portfolio, so as to obtain the maximum return.
- For the prediction of the prices of a given security, different regression models are trained, and only the model with the best fit, i.e. the model with the highest confidence interval is selected to predict the future prices of the security, based on its past prices.

## **2.3 CHALLENGES FACED**

In the implementation of this program, a few challenges were faced as well, which took a lot of brainstorming to get around.

- Since the returns of a security normally refer to simple returns, they were used in the project initially. But they did not provide the desired results, since additive operations like mean could not be effectively used on simple returns. Thus, we had to drop the idea, and after a lot of research, come up with logarithmic returns.
- In visualizing the past prices, it was hard to analyze the securities since some of them were comparatively newer and started with lower prices. Thus, we normalized all the prices to 100. This gave a better understanding of the relative performance of a stock in the past 10 years.

- Initially, the stocks were compared based on their returns alone. But we observed that the stocks with the highest returns also showed the biggest dips in prices whenever they encountered a phase of constant loss. Thus, we also included risks as a major determining factor of a stock's performance.
- For analyzing the performances of a portfolio, initially, the securities in each portfolio were given equal weights, i.e. equal ratios of investments. But this model was clearly ineffective in finding the best performing portfolio. Thus, we randomized the weight distribution, and for a more robust model, ran 1000 iterations of the same on each portfolio.
- When the portfolio-and-weight combination with the highest RRR was obtained, we found that the return was not as high as one would want. Thus, we added another functionality to the program that computes the estimated annual return for any given risk value.

In the prediction segment of the program, no major challenges were faced as such.

## 2.4 CODE

Following are the code segments used for the implementation of the program.

```
# Importing the required libraries

import numpy as np
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.tree import DecisionTreeRegressor
from sklearn.svm import SVR
from sklearn.model_selection import train_test_split
```

```

from pandas_datareader import data
import matplotlib.pyplot as plt
plt.style.use('bmh')

# Input the securities

n = int(input('Enter the number of securities: '))
tickers = []
for i in range(n):
    tickers.append(input())

# Combining adjusted closing prices since 2011 into one
dataframe 'sec_data' (current date = 30/10/2020)

sec_data = pd.DataFrame()

for t in tickers:
    x = data.DataReader(t, data_source='yahoo', start='2011-1-
1')
    print('\n'+t+':')
    print(x.head())
    sec_data[t] = x['Adj Close']

print(sec_data)

# STOCK PERFORMANCE SEGMENT

# Computing daily log returns 'sec_returns'

sec_returns = np.log(sec_data / sec_data.shift(1))
print(sec_returns.tail())

# Plotting the daily prices of all securities

(sec_data/sec_data.iloc[0]*100).plot(figsize=(15,10))
plt.title('Past Stock Prices Normalized to 100')
plt.xlabel('Years')
plt.ylabel('Normalized price')
plt.show()

# Computing average annual returns and risks, and their ratios

avg_ret=[]
avg_risk=[]
ret_risk_ratio=[]

```

```

for t in tickers:
    mn=(sec_returns[t].mean()*250)
    sd=(sec_returns[t].std()*250**0.5)
    avg_ret.append(mn)
    avg_risk.append(sd)
    ret_risk_ratio.append(mn/sd)

dict = {'Average Annual Return':avg_ret,'Average Annual Risk':
avg_risk,'Return-to-Risk Ratio':ret_risk_ratio}
mean_returns = pd.DataFrame(dict,index=tickers)
print(mean_returns)

# The security with highest ratio

max_ret_risk_ratio = mean_returns['Return-to-
Risk Ratio'].max()

print('Details of the security with the highest return-to-
risk ratio:')
print(mean_returns.loc[mean_returns['Return-to-
Risk Ratio']==max_ret_risk_ratio])

# All possible portfolios

from itertools import combinations

portfolios = []

for r in range(2, n+1):
    combinations_object = combinations(tickers, r)
    portfolios += combinations_object

print(portfolios)

# 1000 iterations of randomized weight distribution and
plotting return vs risk for each portfolio

port_returns=[]
port_risks=[]
port_weights=[]
mean_ret_t = mean_returns.transpose()
i=0
max_rrr=0
max_rrr_index=-1

```

```

for p in portfolios:
    port=list(p)
    port_dat=mean_ret_t[port]
    l=len(port)
    ann_ret=port_dat.loc['Average Annual Return',:]
    ann_risk=port_dat.loc['Average Annual Risk',:]
    i+=1
    reti=[]
    riski=[]
    wts=[]
    for x in range (1000):
        weights = np.random.random(l)
        weights /= np.sum(weights)
        ret=np.dot(ann_ret,weights)
        ris=np.sqrt(np.dot(weights.T,np.dot(sec_returns[port].
cov() * 250, weights)))
        reti.append(ret)
        riski.append(ris)
        wts.append(weights)
        if ret/ris > max_rrr:
            max_rrr = ret/ris
            max_rrr_index = i
            max_rrr_weights = weights
            max_rrr_ret=ret
    port_returns.append(reti)
    port_risks.append(riski)
    port_weights.append(wts)

    port_df = pd.DataFrame({'Return': reti, 'Risk': riski})
    port_df.plot(x='Risk', y='Return', kind='scatter', figsize
=(10, 6));
    plt.xlabel('Expected Risk')
    plt.ylabel('Expected Return')
    plt.title('CODE = '+str(i)+'      :      '+str(port))

# Portfolio with highest RRR

print('The best choice for a portfolio:')
print(portfolios[max_rrr_index-1])
print('\nThe best ratios of investment in each security (in or
der):')
print(max_rrr_weights)
print('\nThe estimated maximum annual return-to-
risk ratio for this portfolio with the suggested weights:')
print(round(max_rrr, 5))

```



```

print('\nReturn:\t',str(round(np.exp(max_rrr_ret)*100-
100,3)), '%\t', 'Risk:\t',str(round(max_rrr_ret/max_rrr,5)))

# Highest return for a given risk

code=int(input('Enter the code for your preferred portfolio (r
efer the plots above): '))
max_risk = float(input('Enter the maximum risk you are willing
to take (refer the plots above): '))

max_return = np.array(port_returns[code])[tuple(np.where(np.ar
ray(port_risks[code])<=max_risk))].max()
max_return_index = port_returns[code].index(max_return)
print('\nEstimated maximum annual profit with the risk:')
print(str(round(np.exp(max_return)*100-100,3)), '%')
print('\nRespective weights for the securities:')
print(port_weights[code][max_return_index])

# STOCK PRICE PREDICTION SEGMENT

# Dataframe with past and shifted past prices

pred_sec=input('Enter the ticker for the security that you wan
t to predict the prices of: ')
days = int(input('Enter the number of future days for predicti
on: '))
pred_prices = sec_data[[pred_sec]]
pred_prices.columns = ['Past Prices']
pred_prices['Shifted Prices'] = pred_prices[['Past Prices']].s
hift(-days)
print(pred_prices)

# Preparing feature array

X = np.array(pred_prices.drop(['Shifted Prices'],1))[:-days]
print(X)

# Preparing target array

y = np.array(pred_prices['Shifted Prices'])[:-days]
print(y)

# Splitting into training and testing datasets, computing
confidence intervals for different methods of regression

```

```

x_train, x_test, y_train, y_test = train_test_split(X, y, test
_size=0.3)

lin_reg = LinearRegression()
lin_reg.fit(x_train, y_train)
lin_reg_r2 = lin_reg.score(x_test, y_test)
print("Linear regression confidence interval:\t\t", lin_reg_r2
)

dec_tree_reg = DecisionTreeRegressor()
dec_tree_reg.fit(x_train, y_train)
dec_tree_r2 = dec_tree_reg.score(x_test, y_test)
print("Decision tree regressor confidence interval:\t", dec_tr
ee_r2)

svr = SVR()
svr.fit(x_train, y_train)
svr_r2 = svr.score(x_test, y_test)
print("Support vector regression confidence interval:\t", svr_
r2)

# Past prices in the last 'days' days

last_days_prices = np.array(pred_prices.drop(['Shifted Prices'
],1))[-days:]
print(last_days_prices)

# Predicting future prices for 'days' days

predicted_prices = lin_reg.predict(last_days_prices)
print(predicted_prices)

# Plotting the predicted prices

plt.figure(figsize=(15,10))
plt.plot(predicted_prices)
plt.xlabel('Future days')
plt.ylabel('Predicted price (in USD)')
plt.title('Predicted Stock Prices')
plt.show()

```

## **CHAPTER 3**

### **MAIN RESULTS AND INFERENCES**

#### **3.1 MAIN RESULTS**

The result of each individual stage of the project depends on the input given by the user initially. The full dataset will not be imported to the code from Yahoo finance website as it takes a lot of time to pre-process it and it is not required to get details of all the securities as only securities corresponding to user input is needed.

The user inputs the securities they want to invest in. In this implementation of the program, the securities are Apple, Microsoft, Tesla, Nvidia and Nike.

After performing the necessary data pre-processing as explained in Section 2.1, we obtain four main results:

- 1] The best security to invest in to get the maximum return to risk ratio when the user wants to invest in only one security. Figure 3.1 shows the dataframe that contains the average annual returns, average annual risks and the return-to-risk ratios (RRR) for each security. Figure 3.2 shows the security with the highest return to risk ratio. Microsoft, in this case, has the highest RRR. Thus, if a user is going to invest only in one security, Microsoft is their best option.

	Average Annual Return	Average Annual Risk	Return-to-Risk Ratio
AAPL	0.241616	0.283882	0.851114
MSFT	0.223062	0.256446	0.869821
TSLA	0.433183	0.547222	0.791603
NVDA	0.360271	0.420000	0.857788
NKE	0.200057	0.260493	0.767993

Figure 3.1: Average annual returns, risks and their ratios

Details of the security with the highest return-to-risk ratio:			
	Average Annual Return	Average Annual Risk	Return-to-Risk Ratio
MSFT	0.223062	0.256446	0.869821

Figure 3.2: Security with the highest RRR

2] The best portfolio to choose (group of securities) which gives the maximum return for a minimum risk and along with it produces the ratio of investments in each security in the suggested portfolio and the corresponding return percentage and risk accompanied with it. Figure 3.3 shows the best choice for a portfolio. In this case, it is a portfolio with all five securities, to be invested in such a manner that the investment ratios for Apple : Microsoft : Tesla : Nvidia : Nike is the equivalent to 0.21 : 0.21 : 0.16 : 0.18 : 0.24. An investment in this fashion carries the highest return when compared to the risk it carries.

The best choice for a portfolio: ( 'AAPL', 'MSFT', 'TSLA', 'NVDA', 'NKE' )	
The best ratios of investment in each security (in order): [0.21053441 0.20858937 0.16478011 0.17684972 0.23924639]	
The estimated maximum annual return-to-risk ratio for this portfolio with the suggested weights: 1.14157	
Return: 32.36 %	Risk: 0.24559

Figure 3.3: Portfolio-weight-combination with the highest RRR

3] If the user decides to choose his own portfolio after going through the plots of expected return vs expected risk, then what is the best ratio of investments to maximize the return for the specified risk he/she is willing to take. For the given code of portfolio and maximum risk a person is willing to take the best investments ratio and maximum profit is shown in Figure 3.4, where code 26 refers to the portfolio containing all five securities. Figure 3.5 shows the respective return vs risk plot for the portfolio.

```
Enter the code for your preferred portfolio (refer the plots above): 26
Enter the maximum risk you are willing to take (refer the plots above): 0.3

Estimated maximum annual profit with the risk:
38.971 %

Respective weights for the securities:
[0.14272592 0.16392723 0.29019217 0.32261191 0.08054277]
```

Figure 3.4: Weight combination for the highest return for the input portfolio and risk

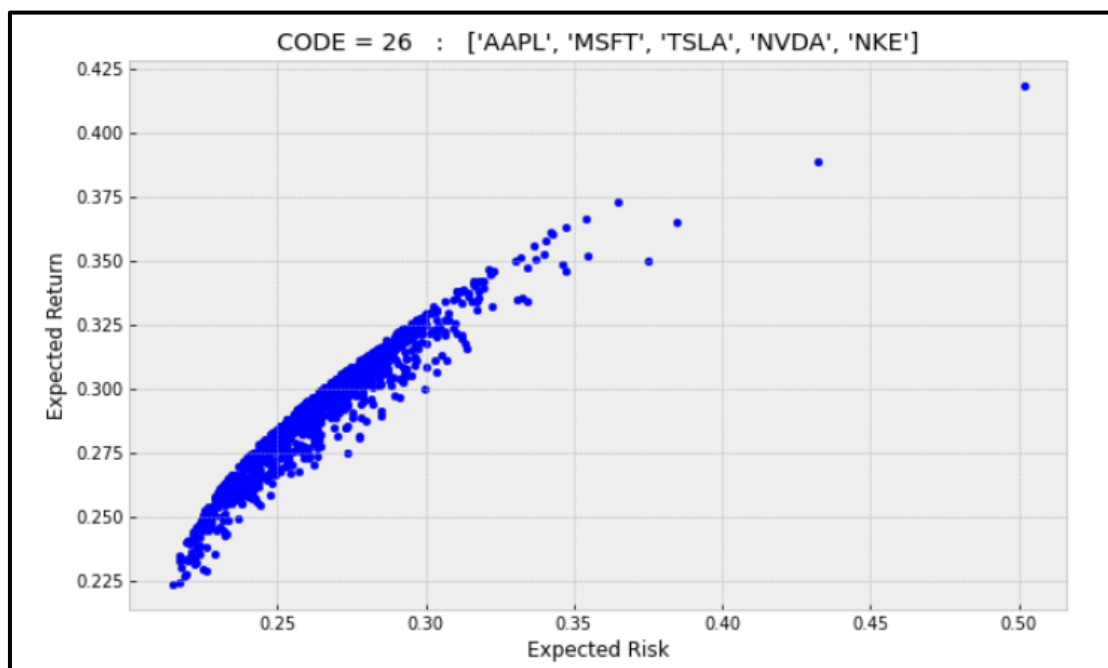


Figure 3.5: Return vs risk for the desired portfolio

This code is the same as that of the suggested portfolio, and one main inference we can take away from this example is that the suggested portfolio had an estimated annual return percentage of 32.36% and corresponding risk of 0.24559. When the risk is increased to 0.3, the return also increases to 38.971%. This proves the fact that risk increases as returns increases.

4] In the prediction part of the project, we are able to predict the adjusted close price for the given security for the specified number of future days using linear regression. We opted for simple linear regression as it was able to give a better fit, i.e. a higher confidence interval than the decision tree regression and support vector regression on the test data. Figure 3.6 shows the coefficients of determination,  $R^2$  (also called confidence intervals) for all three methods of regression.

Linear regression confidence interval:	0.973499568226818
Decision tree regressor confidence interval:	0.971609482880668
Support vector regression confidence interval:	0.9583107657033514

Figure 3.6: Confidence intervals for different methods of regression

As observed, linear regression has the highest confidence interval, i.e the best fit, and an accuracy of 97.35%. Thus, the prediction of future stock prices is performed with the linear regression model. Figure 3.7 shows the plot of the predicted stock prices for the next 250 days, i.e. an entire “market year”. Since trends can be observed annually in the prices of a stock, this prediction is highly likely to be accurate.

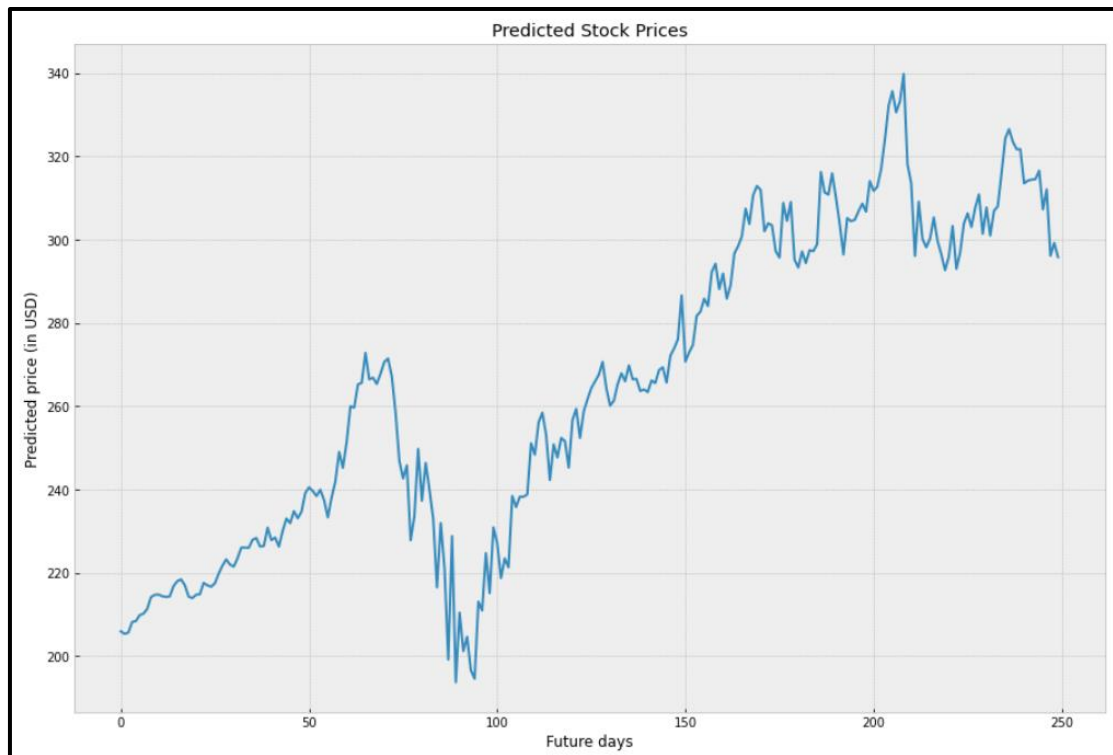


Figure 3.7: Plot of predicted prices for Microsoft (in USD)

### 3.2 INFERENCES

Certain inferences can be drawn from the obtained outputs.

- The stock price and performance prediction can be performed only by using the adjusted close feature. The adjusted closing price is in general terms the closing price of the stock after all the distributions to be done are deduced from it. Other features like opening and closing price do not play any significant role in determining the future prices of the stock and also when analyzing historical data when predicting the performance of stocks. This can be related to the general statement that whatever happens during the day on the stock price does not let us generalize or make further analysis on the

company's stock as it keeps varying unexpectedly. This is why adjusted close price is used as the main feature for further analysis.

- Through the adjusted close price plot shown in Figure 2.5, we can infer how the closing price of each security is varying over time. For the given choice of security, we can infer that the returns of TESLA are varying more than any other security. This can be explained from the plot that when compared to its initial value in 2011, the current closing price is around 80 times higher than what it was back then. Due to this high varying close price we can come to an assumption that TESLA is less likely to be invested in as it is unstable as compared to the others.
- The log returns are preferred over the raw returns as the plot of raw returns is not easy to interpret and it looks noisy. Log returns are preferred when doing analysis on time series data as log transformation converts the given random data to a normal distribution. Another reason why log returns are preferred is because it can be added whereas raw returns are supposed to be compounded.
- When the investor is interested in investing in only one security, we can find the security which gives the maximum return to risk ratio. Average annual return is found using the mean of the log transformed adjusted close price for the 250 days the market is open and risk is given by the standard deviation of this variable. We can infer that for the given set of securities Microsoft (MSFT) had the maximum return to risk ratio, so it will be the first choice if dealing with single security.
- When the user plans to invest in more than one security we try to find the best portfolio which maximizes the return to risk ratio by assigning random



weights for each security and repeating the same for 1000 iterations for each portfolio to get a more robust suggestion. For the given securities we find that the best choice of portfolio is ('AAPL', 'MSFT', 'TSLA', 'NVDA', 'NKE') and the ratio of investments is also given. From this we can infer that the best portfolio is the one in which individual securities are not correlated a lot, that is the return of each security should vary less together so that even if one security fails, it would not drag the other securities down along with it.

- When the user himself chooses the portfolio and also gives the maximum risk he is willing to take, we can infer from the expected return vs risk plot, the point which has the maximum return for the least risk and inform the user about how much return he might get and also how much he should invest in each.
- For a portfolio with a smaller number of securities the plot looks like Figure 3.8.

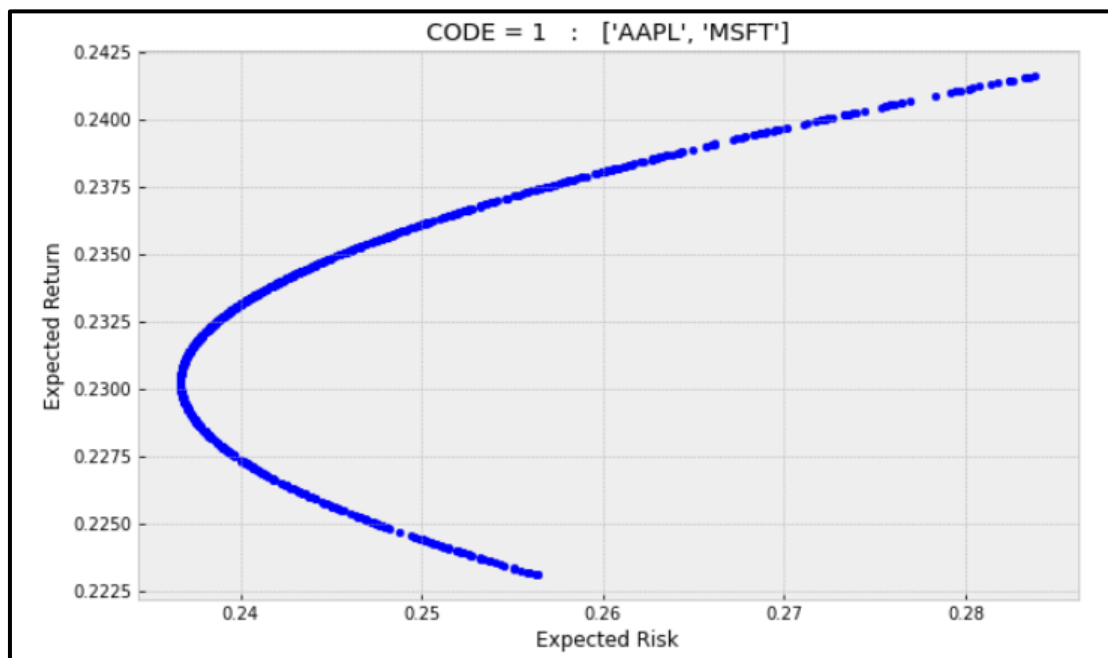


Figure 3.8: Return vs risk for portfolio with 2 securities

There exists a point such that we get the maximum profit for a minimal risk.  
As the number of securities increases the plot looks like Figure 3.9.

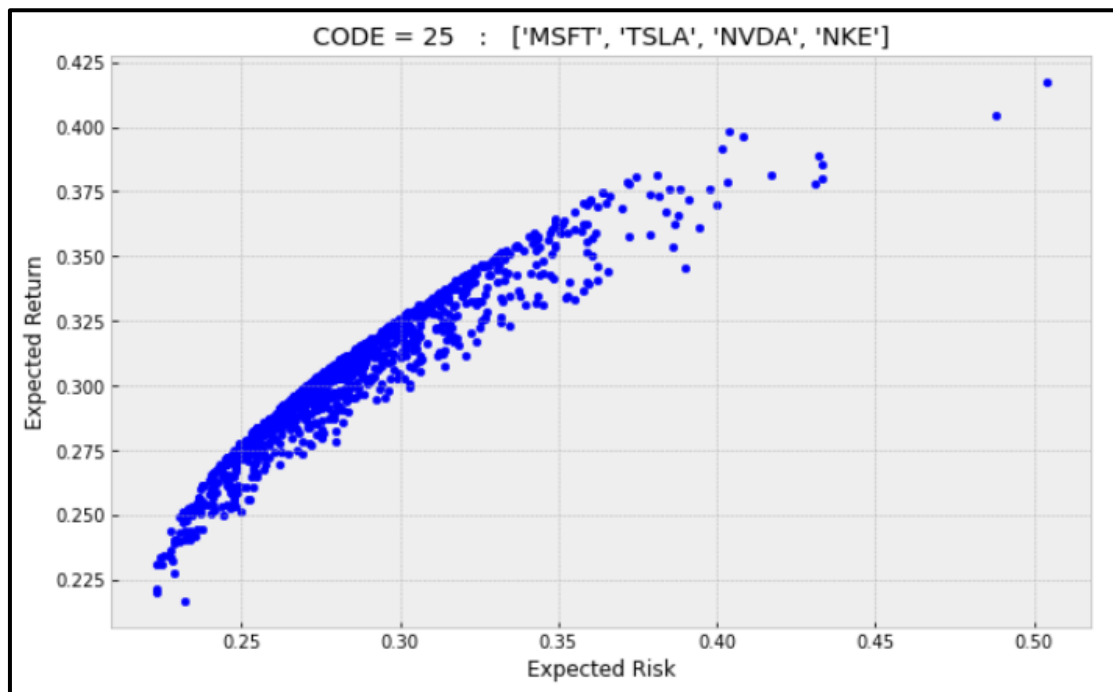


Figure 3.9: Return vs risk for portfolio with 4 securities

The points become scattered and it becomes obvious that for a large number of securities, the risk increases almost proportionally with returns. We also infer that the curve with the tangent that gives the highest return for the lowest risk also disappears.

The plots become scattered because the degrees of freedom increase as the number of securities in the portfolio increases.

- From the test accuracy of above 97% when predicting the adjusted close price for the future number of days, we can infer that linear regression is the best algorithm for this application.

## **CHAPTER 4**

### **CONCLUSION AND RECOMMENDATIONS FOR FUTURE WORK**

#### **4.1 CONCLUSION**

We have derived a program using Linear Regression to predict and Randomized Weight Distribution Simulations to analyze the best stock/securities which are traded in the stock market based on the adjusted closing prices of the traded stock (Historical data). But there are other factors which influence the prices of securities like political and economic conditions, which can be considered for further analysis. This acts as only a basic guide to the common man to provide an idea as to which securities are profitable based on the Historical data.

#### **4.2 RECOMMENDATIONS FOR FUTURE WORK**

The adjusted close value is predicted for future days only by using the historical data of the adjusted close price as the prediction feature used as the predictor variable is composed of only the shifted adjusted close price. A simple linear regression may not always be the perfect choice as the number of predictor variables increases, a better and more robust prediction can be made. In order to do this, we can include features like GDP growth and employment records, as well as import-export records of a country to predict adjusted close price.

Along with it, we can include various features like continuous ups and downs in the adjusted close price, how much trade happens on that day, and integrating social media analytics with stock price and performance prediction will play a key role as we will have knowledge of what is being liked by people currently and what will be the possible growth of a particular security in the future by performing sentiment analysis on tweets, financial news and other open-ended comments or remarks. Since stocks are basically about working with numbers, deep neural network architecture can be used as our main model for stock price prediction as it can uncover even more information compared to traditional machine learning algorithms. LSTM based RNN's can also be used to simultaneously predict various stock prices.

## REFERENCES

- [1] <https://www.learndatasci.com/tutorials/python-finance-part-yahoo-finance-api-pandas-matplotlib/>
- [2] <https://numpy.org/doc/stable/user/quickstart.html>
- [3] [https://pandas.pydata.org/docs/user\\_guide/index.html](https://pandas.pydata.org/docs/user_guide/index.html)
- [4] [https://scikit-learn.org/stable/user\\_guide.html](https://scikit-learn.org/stable/user_guide.html)
- [5] <https://matplotlib.org/tutorials/introductory/usage.html#sphx-glr-tutorials-introductory-usage-py>
- [6] <https://quantivity.wordpress.com/2011/02/21/why-log-returns/>
- [7] <https://www.statisticshowto.com/probability-and-statistics/coefficient-of-determination-r-squared/>