**Risk and Returns : The Sharpe Ratio**

**Prepared by:**

**P. Prathap (N130950)**

**T. Siva (N140191)**

**V. Jaya Naga Sri (N140205)**

**A. Saraswathi (N140280)**

**Guided By:**

**Mr. Sravan Kumar**

**Faculty, Dept. of CSE**

**RGUKT- IIIT, Nuzvid.**

**Rajiv Gandhi University of Knowledge Technologies**

**APIIIT-Nuzvid**

**Department of Computer Science and Engineering**

****

**CERTIFICATE OF PROJECT COMPLETION**

This is to certify that the project work entitled **“Risk and Returns : The sharpe Ratio”** is a bonfide work by **N130950, N140191, N140205, and N140280** submitted in partial fulfillment of the requirement for the award of marks of Mini Project in the **department of CSE** under my guidance during the academic year 2018-2019. This project in my opinion, is worthy of consideration for the award of marks in accordance with the Department and University regulations.

**Date:**

**Place:**

**R.Upendar Rao**

**Head of the Department**

**Department of Computer Science and Engineering**

**Supervisor**

**Mr. Sravan Kumar**

**Faculty**

**Department of CSE**

**November 2018**

**DECLARATION**

We certify that

a) The work contained in the project report is original and has been done by ourselves under the general supervision of our supervisor.

b) The work has not been submitted to any other institute for any degree or diploma.

c) We have followed the guidelines provided by the institute in writing the thesis.

d) We have conformed to the norms and guidelines gives in the Ethical Code of Conduct of the Institute.

e) Whenever we have used material (data, theoretical analysis and text) from other sources, we have given due credit to them by citing in the text of the thesis and giving their details in the reference.

f) Whenever we have quoted written material from other sources, we have put them under quotations marks and given due credit to them by citing them and giving their details in the references.

**P. Prathap (N130950)**

**T. Siva (N140191)**

**V. Jaya Naga Sri (N140205)**

**A. Saraswathi (N140280)**

**Table of Contents**

1. **Abstract………………………………………………………1**

**2. Introduction…………………………………………………1**

2.1 Purpose

2.2 Scope

**3. Problem Definition ……………………………………….2**

3.1 Problem Definition

**4. System Analysis…………………………………………….2**

4.1 Data Flow Diagrams

**5. Software Requirement Specification…………………4**

5.1 Definition of SRS

5.2 Requirement Analysis

5.3 Requirement Specification

**6. Document Design……………………………………………7**

6.1 System Design

**7. About Software………………………………………………7**

7.1 Over View of Python

7.2 Over View of Jupiter Notebook

**8. Algorithm…….………………………………………………9**

8.1 Algorithm

**9. Coding………………………………………………………….10**

**10. Testing………………………………………………………..12**

10.1 A/B Testing

**11. Screens……………………………………………………….13**

**12. Conclusion…………………………………………………..14**

**13. Bibliography………………………………………………..15**

**1.ABSTRACT**

When you assess whether to invest in an asset, you want to look not only at how much money you could make but also at how much risk you are taking. We use The Sharpe Ratio, developed by Nobel Prize winner William Sharpe and relate this return to the risk of the investment. An investment may make sense if we expect it to return more money than it costs. But returns are only part of the story because they are risky - there may be a range of possible outcomes. Investors can use this project to determine whether to buy an asset or not. Using this they can avoid loss of investment.

**2. INTRODUCTION**

**2.1 Purpose:**

When you assess whether to invest in an asset, you want to look not only at how much money you could make but also at how much risk you are taking. We use The Sharpe Ratio, developed by Nobel Prize winner William Sharpe and relate this return to the risk of the investment. An investment may make sense if we expect it to return more money than it costs.

**2.2 Scope:**

* Investors can use this project to determine whether to buy an asset or not. Using this they can avoid loss of investment.

**3. PROBLEM DEFINITION**

**3.1 Problem Definition**

An investment may make sense if we expect it to return more money than it costs. But returns are only part of the story because they are risky - there may be a range of possible outcomes. How does one compare different investments that may deliver similar results on average, but exhibit different levels of risks?

**4. SYSTEM ANALYSIS**

System Analysis is first stage according to System Development Life Cycle model. This System Analysis is a process that starts with the analyst.

Analysis is a detailed study of the various operations performed by a system and their relationships within and outside the system. One aspect of analysis is defining the boundaries of the system and determining whether or not a candidate should consider other related systems.

**4.1 DATAFLOW DIAGRAMS:**

A graphical tool used to describe and analyze the moment of data through a system manual or automated including the process, stores of data, and delays in the system. Data Flow Diagrams are the central tool and the basis from which other components are developed. The transformation of data from input to output, through processes, may be described logically and independently of the physical components associated with the system. The DFD is also known as a data flow graph or a bubble chart.

**DFD Symbols:**

**Dataflow:**

Data move in a specific direction from an origin to a Destination.

**Process:**

People, procedures, or devices that use or produce (Transform) Data. The physical component is not identified.

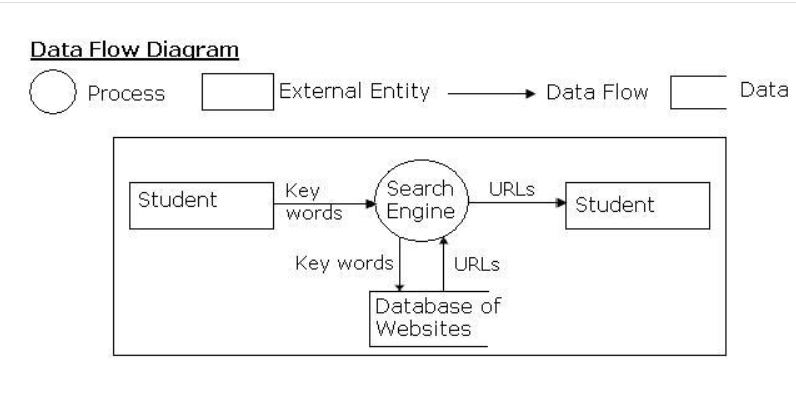
**Source:**

External sources or destination of data, which may be People , programs, organizations or other entities.

**Data Store:**

Here data are stored or referenced by a process in the System

DATAFLOW DIAGRAM



**5. SOFTWARE REQUIREMENT SPECIFICATION**

**5.1 Definition of SRS:**

The SRS is the means of translating the ideas of the minds of client into a formal document .and also it fully describes what the software will do and how it will be expected to perform.

**5.2 Requirement Analysis:**

This stage is to obtain a clear picture of the needs and requirements of the end-user and also the organization. Analysis involves interaction between the clients and the analysis. Usually analysts research a problem from any questions asked and reading existing documents. The analysts have to uncover the real needs of the user even if they don’t know them clearly.

* The information domain of the problem must be represented and understood.
* The functions that the software is to perform must be defined.
* The behavior of the software as a consequence of external events must be defined.
* The analysis process must move from essential information to implementation detail.

**5.3 Requirement Specification:**

**Specification Principles:**

Software Requirements Specification plays an important role in creating quality software solutions. Specification is basically a representation process. Requirements are represented in a manner that ultimately leads to successful software implementation.

Requirements may be specified in a variety of ways. However, there are some guidelines worth following: -

* Representation format and content should be relevant to the problem
* Information contained within the specification should be nested
* Diagrams and other notational forms should be restricted in number and consistent in use.
* Representations should be revisable.

**Software Requirements Specifications:**

The software requirements specification is produced at the culmination of the analysis task. The function and performance allocated to the software as a part of system engineering are refined by establishing a complete information description, a detailed functional and behavioral description, and indication of performance requirements and design constraints, appropriate validation criteria and other data pertinent to requirements.

**External Interface Requirements:**

**User Interfaces:**

The user interface for the software shall be compatible to any browser such as Internet Explorer, Mozilla or Chrome by which user can access to the system. The user interface shall be implemented using Anaconda 5.3(includes 1,400+packages for ML).

**Hardware Interfaces:**

Since the application uses heavy data set, we should have high hardware configured system.

**Software Interfaces:**

We use Anaconda 5.3 it includes 1,400+packages for machine learning. We give a csv file which consists of all information about the stocks (stocks dataset). We load the data using panda’s read\_ csv file.

***Software Quality Attributes:***

Stock Prices are adaptable to any changes. So we should maintain updated stock prices.

**6.DOCUMENT DESIGN**

**6.1 SYSTEM DESIGN:**

**Hardware Requirements:**

* I3 processor
* 4GB RAM
* 60GB to 80GB Hard disk space

**6.2 Software Requirements:**

**Language :** Python 3.7

**Software :** Anaconda 5.3(includes 1,400+packages for ML)

**Operating System:** Operating System (Windows, Linux, Mac)

**7.ABOUT SOFTWARE**

**7.1 OVERVIEW OF PYTHON:**

Python is a multi-paradigm, general-purpose, interpreted, high-level programming language. Python allows programmers to use different programming styles to create simple or complex programs, get quicker results and write code almost as if speaking in a human language. Some of the popular systems and applications that have employed Python during development include Google Search, YouTube, Bit Torrent, Google App Engine, Eve Online, Maya and iRobot machines

Python’s initial development was spearheaded by Guido van Rossum in the late 1980s. Today, it is developed by the Python Software Foundation. Because Python is a multi-paradigm language, Python programmers can accomplish their tasks using different styles of programming: object oriented, imperative, functional or reflective. Python can be used in Web development, numeric programming, game development, serial port access and more.

There are two attributes that make development time in Python faster than in other programming languages:

1. Python is an interpreted language, which precludes the need to compile code before executing a program because Python does the compilation in the background. Because Python is a high-level programming language, it abstracts many sophisticated details from the programming code. Python focuses so much on this abstraction that its code can be understood by most novice programmers.
2. Python code tends to be shorter than comparable codes. Although Python offers fast development times, it lags slightly in terms of execution time. Compared to fully compiling languages like C and C++, Python programs execute slower. Of course, with the processing speeds of computers these days, the speed differences are usually only observed in benchmarking tests, not in real-world operations. In most cases, Python is already included in Linux distributions and Mac OS X machines.

**7.2 OVERVIEW OF JUPYTER NOTEBOOK:**

A Jupyter Notebook is like a digital trapper keeper filled with everything an individual or organization needs to analyze data. This is a key feature of Jupyter. Each Notebook has not only the data contained within it, but also the necessary software libraries needed to view the data and code contained therein. Each Notebook is completely self-contained; anyone who accesses a Notebook can run it and view the data as intended. If the author so chooses, they can package Jupyter Notebooks in a way in which viewers don’t even have to download anything at all. The data can be hosted on a server and rendered out as HTML. What this means for the original author(s) of a Jupyter Notebook is that the code or data within a Notebook is perfectly preserved and ready to be viewed by anyone who has access. Users can even fork the Notebooks themselves, so they can adjust the findings within the Notebook.

Jupyter has been developed from the ground up to be open source and completely flexible. It started as a project called IPython and was developed by Fernando Pérez. Jupyter became a spin off project from IPython that supported three core languages at first: Julia, Python, and R (which, combined, gives the name Jupyter). Now, Jupyter Notebooks can support programming language kernel’s like Ruby, Haskell, C#, Perl, PHP, JavaScript, and purportedly 50 other languages. This makes Jupyter something of a wunderkind in the software engineering world. That it can support so many languages makes it a powerful tool for disseminating data. As mentioned before Jupyter Notebooks are distributed with all the necessary code and software libraries. The end user does not need to install anything on their computer to view the Notebook.

**8.Algorithm**

**8.1 Algorithm**

**Step 1:** compute the percentage change in value from one day to the next of S&P. (For both S&P 500 and stocks).

**Step 2:** compute the relative performance of stocks vs. the S&P 500 benchmark.(stock\_returns - sp\_returns for each day).

**Step 3:** compute the Average difference in daily returns Stocks vs S&P 500.

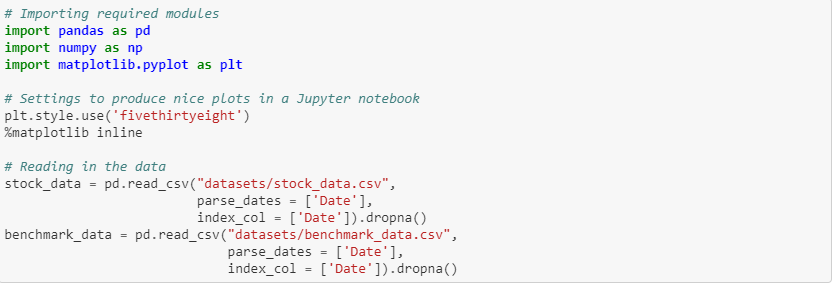
**Step 4:** compute Standard Deviation of the Return Difference.

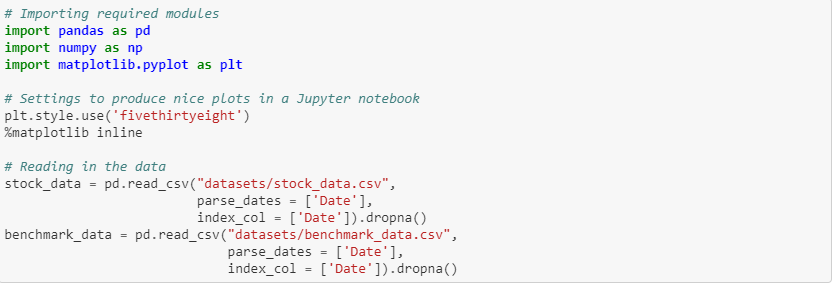
**Step 5:** compute the ratio of Average difference and Standard deviation. This is Sharpe ratio of *daily returns*.

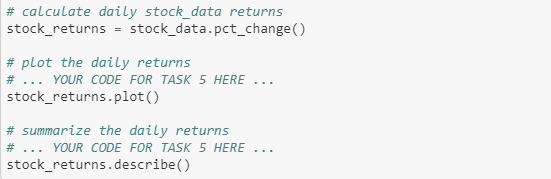
**Step 6:** compute *Annualized Sharpe Ration* by multiplying *Daily Sharpe Ratio* with *sqrt of no.of trading days.*

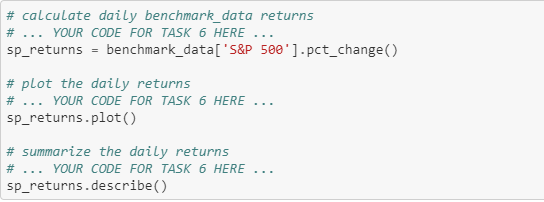
*(trading days=252 (*5 days, 52 weeks, minus a few holidays*))*

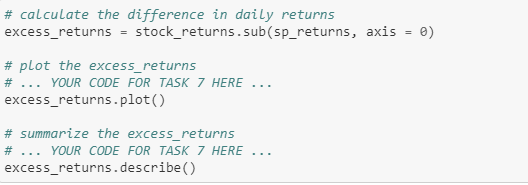
**9. CODING**

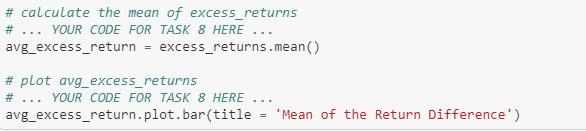


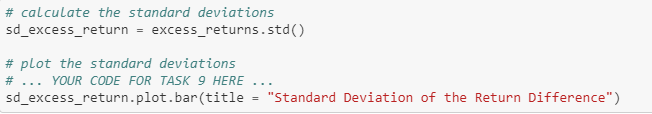


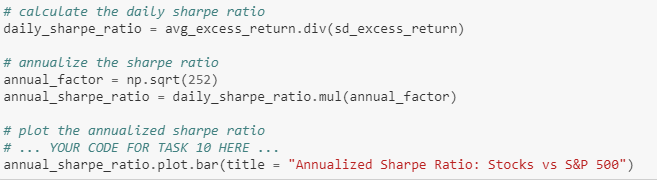












**10. TESTING**

Software Testing is a critical element of software quality assurance and represents the ultimate review of specification, design and coding, Testing presents an interesting anomaly for the software engineer.

**Testing Principles:**

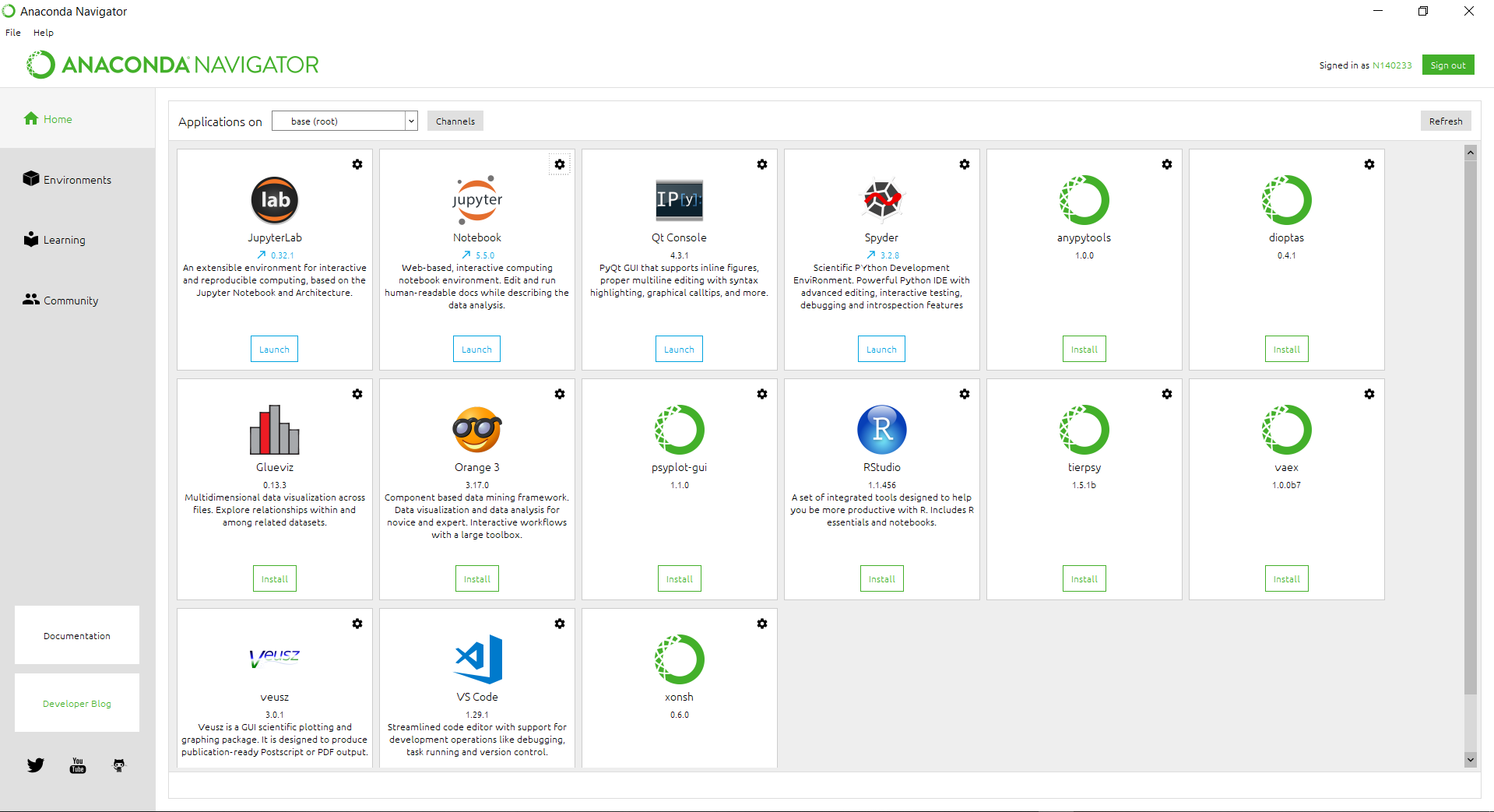
* All tests should be traceable to end user requirements
* Tests should be planned long before testing begins.
* Testing should begin on a small scale and progress towards testing in large.
* Exhaustive testing is not possible.
* To be most effective testing should be conducted by an independent third party.

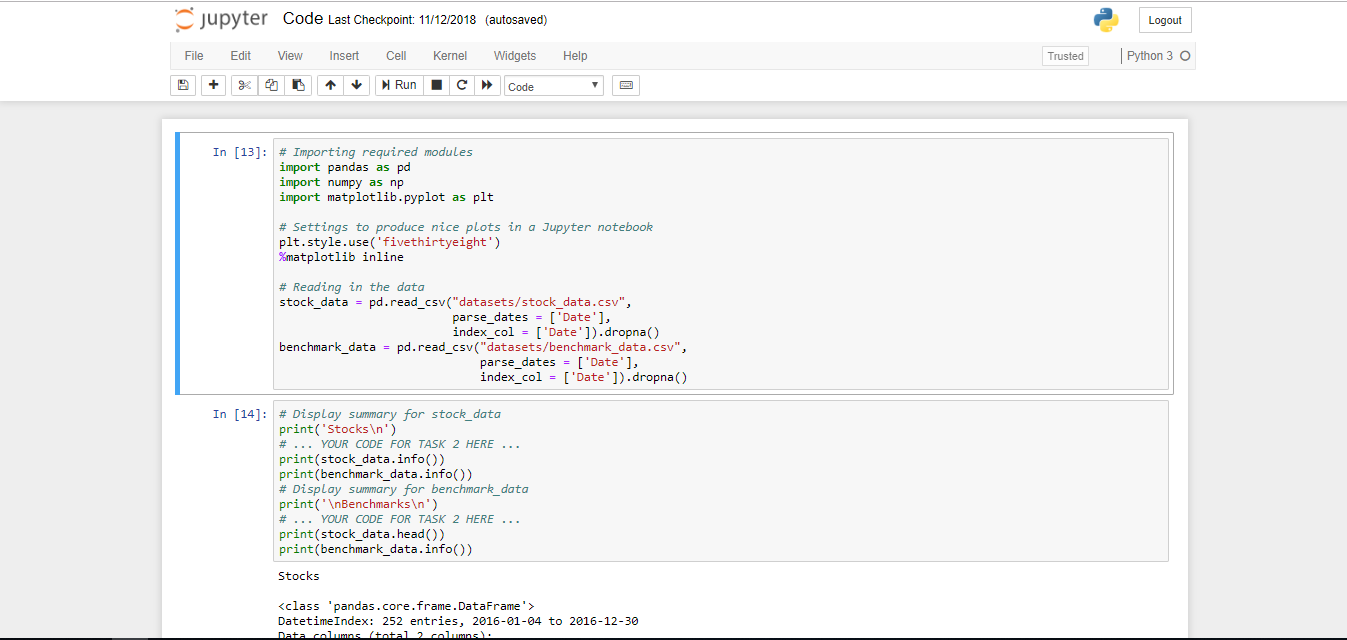
**10.1 A/B Testing:**

A/B testing is a well-known method to check what pleases your customers and identify user-experience improvements. In short, you separate your users in two groups, one sees the same old page, the second sees the experimental page, you gather statistics, and after a while you decide whether the user interface improvement is worthwhile. This approach is great as it does not leave your user interface decisions to random, rather it helps you build arguments for every change.

Do you have time to test on a small group, sit-back, analyze the results and launch on a large scale? Then you should use A/B testing. A good example is emailing: you can send two different e-mails with two different catch phrases to two samples, you see which one actually convert more, and then choose the right email to send to everyone. You have to decide constantly on what to show? You should use machine learning, to learn from the past and give each customer a unique experience.

**11.SCREENS**





**12. CONCLUSION**

* Even though it is classifying, we are not taking market trend into consideration. So, we will incorporate market trends for better accuracy.
* We are Planning to make it accessible via web interface.

**13. BIBLIOGRAPHY**

1.datacamp.com