**Stock Market Analysis**

**Table of Contents**

**Page No.**

* Abstract 5
* Introductiom 5
* Overview
* Aim of the Project
* Technologies 5
* Analysis
* Feaability Study 15
* Machine Learning 27
* Tools 28
* Software Requirement 29
* Problems 29
* Code with Output 33
* Conclusion and Future Scope 43

**ABSTRACT**

Stock Market Analysis and Prediction is the project on technical analysis, visualization, and prediction using data provided by Google Finance. By looking at data from the stock market, particularly some giant technology stocks and others. Used pandas to get stock information, visualized different aspects of it, and finally looked at a few ways of analyzing the risk of a stock, based on its previous performance history. Predicted future stock prices through a Monte Carlo method!

**1.INTRODUCTION**

**1.1. OVERVIEW**

Stock Market Data Analysis is intended exclusively for analyzing the data of stock market for various organization. Using this type of data analysis any organization can get the stock related information in an easy way.

**1.2. AIM OF THE PROJECT**

The main aim of my project is to analyze the data of any organization in any format we require.

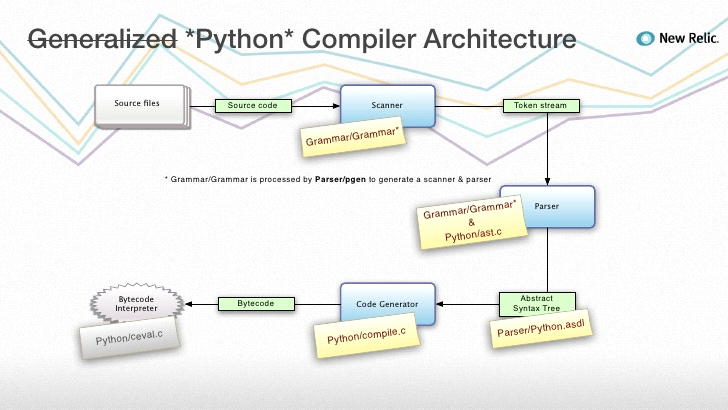
**2.TECHNOLOGIES**

**2.1. ABOUT PYTHON**

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace. It provides constructs that enable clear programming on both small and large scales.

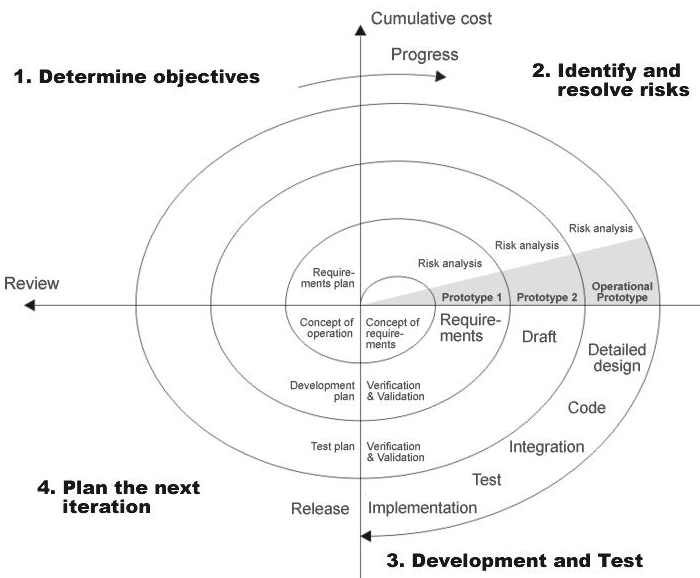
Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

**2.2 ARCHITECTURE OF PYTHON**



**ANALYSIS**

**PROCESS MODEL USED WITH JUSTIFICATION:**

**SDLC (Spiral Mo**

SDLC is nothing but Software Development Life Cycle. It is a standard which is used by software industry to develop good software.

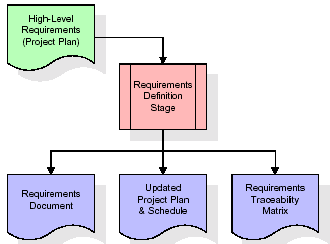
**Stages in SDLC:**

* Requirement Gathering
* Analysis
* Designing
* Coding
* Testing
* Maintenance

**Requirements Gatheringstage:**

The requirements gathering process takes as its input the goals identified in the high-level requirements section of the project plan. Each goal will be refined into a set of one or more requirements. These requirements define the major functions of the intended application.

Operational data areas and reference data areas, and define the initial data entities. Major functions include critical processes to be managed, as well as mission critical inputs, outputs and reports. A user class hierarchy is developed and associated with these major functions, data areas, and data entities. Each of these definitions is termed a Requirement. Requirements are identified by unique requirement identifiers and, at minimum, contain a requirement title andtextual description.



These requirements are fully described in the primary deliverables for this stage: the Requirements Document and the Requirements Traceability Matrix (RTM). The requirements document contains complete descriptions of each requirement, including diagrams and references to external documents as necessary. Note that detailed listings of database tables and fields are *not* included in the requirements document.

The title of each requirement is also placed into the first version of the RTM, along with the title of each goal from the project plan. The purpose of the RTM is to show that the product components developed during each stage of the software development lifecycle are formally connected to the components developed in prior stages.

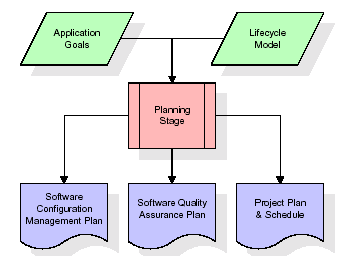
In the requirements stage, the RTM consists of a list of high-level requirements, or goals, by title, with a listing of associated requirements for each goal, listed by requirement title. In this hierarchical listing, the RTM shows that each requirement developed during this stage is formally linked to a specific product goal. In this format, each requirement can be traced to a specific product goal, hence the term *requirements traceability*.

The outputs of the requirements definition stage include the requirements document, the RTM, and an updated project plan.

* Feasibility study is all about identification of problems in a project.
* No. of staff required to handle a project is represented as Team Formation, in this case only modules are individual tasks will be assigned to employees who are working for that project.
* Project Specifications are all about representing of various possible inputs submitting to the server and corresponding outputs along with reports maintained by administrator

**Analysis Stage:**

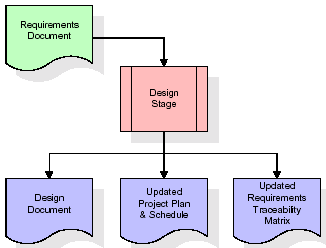
The planning stage establishes a bird's eye view of the intended software product, and uses this to establish the basic project structure, evaluate feasibility and risks associated with the project, and describe appropriate management and technical approaches.



The most critical section of the project plan is a listing of high-level product requirements, also referred to as goals. All of the software product requirements to be developed during the requirements definition stage flow from one or more of these goals. The minimum information for each goal consists of a title and textual description, although additional information and references to external documents may be included. The outputs of the project planning stage are the configuration management plan, the quality assurance plan, and the project plan and schedule, with a detailed listing of scheduled activities for the upcoming Requirements stage, and high level estimates of effort for the out stages.

**Designing Stage:**

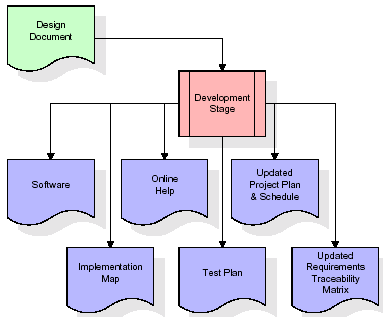
The design stage takes as its initial input the requirements identified in the approved requirements document. For each requirement, a set of one or more design elements will be produced as a result of interviews, workshops, and/or prototype efforts. Design elements describe the desired software features in detail, and generally include functional hierarchy diagrams, screen layout diagrams, tables of business rules, business process diagrams, pseudo code, and a complete entity-relationship diagram with a full data dictionary. These design elements are intended to describe the software in sufficient detail that skilled programmers may develop the software with minimal additional input.



When the design document is finalized and accepted, the RTM is updated to show that each design element is formally associated with a specific requirement. The outputs of the design stage are the design document, an updated RTM, and an updated project plan.

**Development (Coding) Stage:**

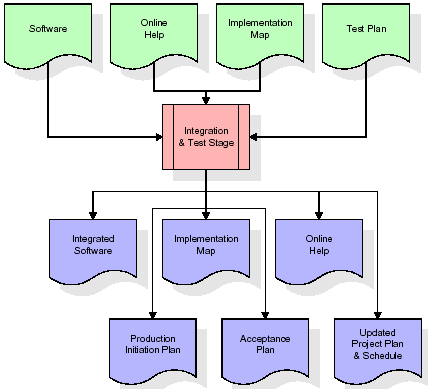
The development stage takes as its primary input the design elements described in the approved design document. For each design element, a set of one or more software artifacts will be produced. Software artifacts include but are not limited to menus, dialogs, and data management forms, data reporting formats, and specialized procedures and functions. Appropriate test cases will be developed for each set of functionally related software artifacts, and an online help system will be developed to guide users in their interactions with the software.



The RTM will be updated to show that each developed artifact is linked to a specific design element, and that each developed artifact has one or more corresponding test case items. At this point, the RTM is in its final configuration. The outputs of the development stage include a fully functional set of software that satisfies the requirements and design elements previously documented, an online help system that describes the operation of the software, an implementation map that identifies the primary code entry points for all major system functions, a test plan that describes the test cases to be used to validate the correctness and completeness of the software, an updated RTM, and an updated project plan.

**Integration & Testing Stage:**

During the integration and test stage, the software artifacts, online help, and test data are migrated from the development environment to a separate test environment. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite confirms a robust and complete migration capability. During this stage, reference data is finalized for production use and production users are identified and linked to their appropriate roles. The final reference data (or links to reference data source files) and production user list are compiled into the Production Initiation Plan.

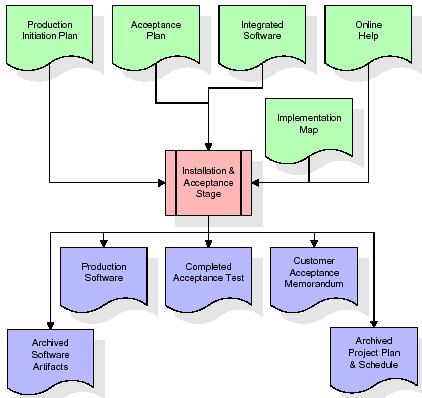


The outputs of the integration and test stage include an integrated set of software, an online help system, an implementation map, a production initiation plan that describes reference data and production users, an acceptance plan which contains the final suite of test cases, and an updated project plan.

* **Installation & Acceptance Test:**

During the installation and acceptance stage, the software artifacts, online help, and initial production data are loaded onto the production server. At this point, all test cases are run to verify the correctness and completeness of the software. Successful execution of the test suite is a prerequisite to acceptance of the software by the customer.

After customer personnel have verified that the initial production data load is correct and the test suite has been executed with satisfactory results, the customer formally accepts the delivery of the software.



The primary outputs of the installation and acceptance stage include a production application, a completed acceptance test suite, and a memorandum of customer acceptance of the software. Finally, the PDR enters the last of the actual labor data into the project schedule and locks the project as a permanent project record. At this point the PDR "locks" the project by archiving all software items, the implementation map, the source code, and the documentation for future reference.

**Maintenance:**

Outer rectangle represents maintenance of a project, Maintenance team will start with requirement study, understanding of documentation later employees will be assigned work and they will undergo training on that particular assigned category.

For this life cycle there is no end, it will be continued so on like an umbrella (no ending point to umbrella sticks).

**SYSTEM ARCHITECTURE**

**Architecture flow:**

Below architecture diagram represents mainly flow of requests from users to database through servers. In this scenario overall system is designed in three tires separately using three layers called presentation layer, business logic layer and data link layer. This project was developed using 3-tire architecture

**FEASIBILITY STUDY**

**SYSTEM STUDY**

**Feasibility study:**

Feasibility study is the initial design stage of any project, which brings together the elements of knowledge that indicate if a project is possible or not. All projects are feasible if they have unlimited resources and infinite time. But the development of the software is plagued by the scarcity of resources and difficult delivery rate. It is necessary and prudent to evaluate the feasibility of the project at the earlier times.

**Economic Feasibility:**

The procedure is to determine the benefits and savings that are expected from the candidate system and compare with the cost. It benefits outweigh cost then the decision is to make design and implement the system. Otherwise further justification or alternatives in the proposed system that have to make if it having a change of the system lifecycle. For my project iam not expecting any feasibility costs spent on this project because here I am using open source environments.

**Technical Feasibility:**

Technical feasibility centres on the existing system and to what extent it support the proposed addition if the budget is serious constraint, then the project is judged not feasible. The technical feasible is important role in my project because I am using sentiment analysis.

Technical feasibility study is the complete study of the project in terms of input, processes, output, fields, programs and procedures. It is a very effective tool for long term planning and trouble shooting. The technical feasibility study should most essentially support the financial information of an organization.

**Operational Feasibility:**

Operational feasibility refers to the measure of solving problems with the help of a new proposed system. It helps in taking advantage of the opportunities and fulfils the requirements as identified during the development of the project. It takes care that the management and the users support the project.

**What can Python do?**

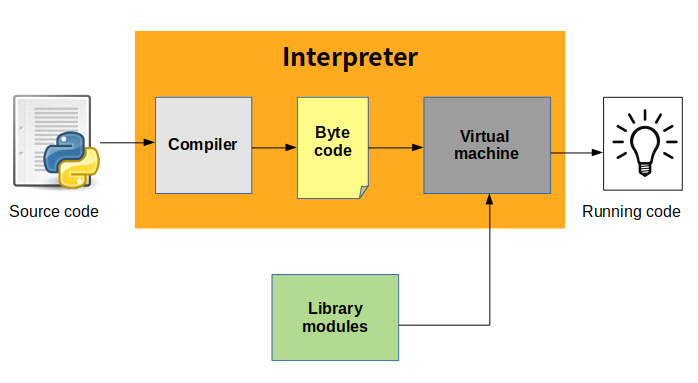
* Python can be used on a server side to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.
* The Python language itself is managed by [**Python Software Foundation**](https://www.python.org/)**,** who offer a reference implementation of python, called CPython under an open source license.

**Why Python?**

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

**Python Syntax compared to other programming languages:**

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.



**Compilation of code:**

Python first **compiles** your source code (.py file) into a format known as **byte code**. Compilation is simply a translation step, and byte code is a lower-level, and platform-independent, representation of your source code. Compiled code is usually stored in**.pyc files**, and is regenerated when the source is updated, or when otherwise necessary. In order to distribute a program to people who already have **Python** installed, you can ship either the .py files or the .pyc files.

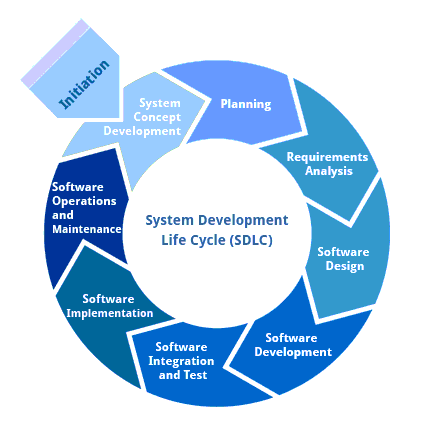
The **byte code** (.pyc file) is loaded into the Python runtime and interpreted by a **Python Virtual Machine**, which is a piece of code that reads each instruction in the **byte code** and executes whatever operation is indicated. Byte code compilation is automatic, and the PVM is just part of the Python system that you have installed on your machine. The PVM is always present as part of the **Python system**, and is the component that truly runs your scripts. Technically, it's just the last step of what is called the Python interpreter. And this is how the process is done (very general). Of course, there are **optimizations** and caches to improve the performance.

Each time an **interpreted program** is run; the interpreter must convert source code into machine code and also pull in the **runtime libraries**. This conversion process makes the program run slower than a comparable program written in a compiled language. Python do something clever to improve its **performance**. It compiles to byte code (.pyc files) the first time it executes a file. This improves substantially the **execution** of the code next time the module is imported or executed.

* **Easy-to-learn** − Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* **Easy-to-read** − Python code is more clearly defined and visible to the eyes.
* **Easy-to-maintain** − Python's source code is fairly easy-to-maintain.
* **A broad standard library** − Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* **Interactive Mode** − Python has support for an interactive mode which allows interactive testing and debugging of snippets of code.
* **Portable** − Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* **Extendable** − You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* **Databases** − Python provides interfaces to all major commercial databases.
* **GUI Programming** − Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.
* **Scalable** − Python provides a better structure and support for large programs than shell scripting.
* It supports functional and structured programming methods as well as OOP.
* It can be used as a scripting language or can be compiled to byte-code for building large applications.
* It provides very high-level dynamic data types and supports dynamic type checking.
* It supports automatic garbage collection.
* It can be easily integrated with C, C++, COM, ActiveX, CORBA, and Java.

**Software Development Life Cycle**

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in systems engineering, information systems and software engineering, is the process of creating or altering systems, and the models and methodologies use to develop these systems.



**Requirement Analysis and Design**

Analysis gathers the requirements for the system. This stage includes a detailed study of the business needs of the organization. Options for changing the business process may be considered. Design focuses on high level design like, what programs are needed and how are they going to interact, low-level design (how the individual programs are going to work), interface design (what are the interfaces going to look like) and data design (what data will be required). During these phases, the software's overall structure is defined. Analysis and Design are very crucial in the whole development cycle. Any glitch in the design phase could be very expensive to solve in the later stage of the software development. Much care is taken during this phase. The logical system of the product is developed in this phase.

**Implementation**

In this phase the designs are translated into code. Computer programs are written using a conventional programming language or an application generator. Programming tools like Compilers, Interpreters, and Debuggers are used to generate the code. Different high level programming languages like C, C++, Pascal, Java, .Net are used for coding. With respect to the type of application, the right programming language is chosen.

**Testing**

In this phase the system is tested. Normally programs are written as a series of individual modules, this subject to separate and detailed test. The system is then tested as a whole. The separate modules are brought together and tested as a complete system. The system is tested to ensure that interfaces between modules work (integration testing), the system works on the intended platform and with the expected volume of data (volume testing) and that the system does what the user requires (acceptance/beta testing).

**Maintenance**

Inevitably the system will need maintenance. Software will definitely undergo change once it is delivered to the customer. There are many reasons for the change. Change could happen because of some unexpected input values into the system. In addition, the changes in the system could directly affect the software operations. The software should be developed to accommodate changes that could happen during the post implementation period.

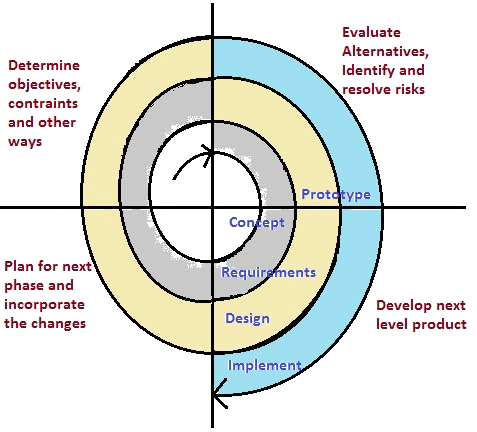
**SDLC METHDOLOGIES**

This document play a vital role in the development of life cycle (SDLC) as it describes the complete requirement of the system. It means for use by developers and will be the basic during testing phase. Any changes made to the requirements in the future will have to go through formal change approval process.

**SPIRAL MODEL** was defined by Barry Boehm in his 1988 article, “A spiral Model of Software Development and Enhancement. This model was not the first model to discuss iterative development, but it was the first model to explain why the iteration models.

As originally envisioned, the iterations were typically 6 months to 2 years long. Each phase starts with a design goal and ends with a client reviewing the progress thus far. Analysis and engineering efforts are applied at each phase of the project, with an eye toward the end goal of the project.

**The following diagram shows how a spiral model acts like:**



The steps for Spiral Model can be generalized as follows:

* The new system requirements are defined in as much details as possible. This usually involves interviewing a number of users representing all the external or internal users and other aspects of the existing system.
* A preliminary design is created for the new system.
* A first prototype of the new system is constructed from the preliminary design. This is usually a scaled-down system, and represents an approximation of the characteristics of the final product.
* A second prototype is evolved by a fourfold procedure:
* Evaluating the first prototype in terms of its strengths, weakness, and risks.
* Defining the requirements of the second prototype.
* Planning an designing the second prototype.
* Constructing and testing the second prototype.
* At the customer option, the entire project can be aborted if the risk is deemed too great. Risk factors might involved development cost overruns, operating-cost miscalculation, or any other factor that could, in the customer’s judgment, result in a less-than-satisfactory final product.
* The existing prototype is evaluated in the same manner as was the previous prototype, and if necessary, another prototype is developed from it according to the fourfold procedure outlined above.
* The preceding steps are iterated until the customer is satisfied that the refined prototype represents the final product desired.
* The final system is constructed, based on the refined prototype.
* The final system is thoroughly evaluated and tested. Routine maintenance is carried on a continuing basis to prevent large scale failures and to minimize down time.

**4.1. FUNCTIONAL REQUIREMENTS**

**OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provides a permanent copy of the results for later consultation. The various types of outputs in general are:

* External Outputs, whose destination is outside the organization
* Internal Outputs whose destination is within organization and they are the
* User’s main interface with the computer.
* Operational outputs whose use is purely within the computer department.
* Interface outputs, which involve the user in communicating directly.

**OUTPUT DEFINITION**

**The outputs should be defined in terms of the following points:**

* Type of the output
* Content of the output
* Format of the output
* Location of the output
* Frequency of the output
* Volume of the output
* Sequence of the output

It is not always desirable to print or display data as it is held on a computer. It should be decided as which form of the output is the most suitable.

**INPUT DESIGN**

Input design is a part of overall system design. The main objective during the input design is as given below:

* To produce a cost-effective method of input.
* To achieve the highest possible level of accuracy.
* To ensure that the input is acceptable and understood by the user.

**INPUT STAGES:**

The main input stages can be listed as below:

* Data recording
* Data transcription
* Data conversion
* Data verification
* Data control
* Data transmission
* Data validation
* Data correction

**INPUT TYPES:**

It is necessary to determine the various types of inputs. Inputs can be categorized as follows:

* External inputs, which are prime inputs for the system.
* Internal inputs, which are user communications with the system.
* Operational, which are computer department’s communications to the system?
* Interactive, which are inputs entered during a dialogue.

**INPUT MEDIA:**

At this stage choice has to be made about the input media. To conclude about the input media consideration has to be given to;

* Type of input
* Flexibility of format
* Speed
* Accuracy
* Verification methods
* Rejection rates
* Ease of correction
* Storage and handling requirements
* Security
* Easy to use
* Portability

Keeping in view the above description of the input types and input media, it can be said that most of the inputs are of the form of internal and interactive. As

Input data is to be the directly keyed in by the user, the keyboard can be considered to be the most suitable input device.

**ERROR AVOIDANCE**

At this stage care is to be taken to ensure that input data remains accurate form the stage at which it is recorded up to the stage in which the data is accepted by the system. This can be achieved only by means of careful control each time the data is handled.

**ERROR DETECTION**

Even though every effort is make to avoid the occurrence of errors, still a small proportion of errors is always likely to occur, these types of errors can be discovered by using validations to check the input data.

**DATA VALIDATION**

Procedures are designed to detect errors in data at a lower level of detail. Data validations have been included in the system in almost every area where there is a possibility for the user to commit errors. The system will not accept invalid data. Whenever an invalid data is keyed in, the system immediately prompts the user and the user has to again key in the data and the system will accept the data only if the data is correct. Validations have been included where necessary.

The system is designed to be a user friendly one. In other words the system has been designed to communicate effectively with the user. The system has been designed with popup menus.

**USER INTERFACE DESIGN**

It is essential to consult the system users and discuss their needs while designing the user interface:

**USER INTERFACE SYSTEMS CAN BE BROADLY CLASIFIED AS:**

1. User initiated interface the user is in charge, controlling the progress of the user/computer dialogue. In the computer-initiated interface, the computer selects the next stage in the interaction.
2. Computer initiated interfaces

In the computer initiated interfaces the computer guides the progress of the user/computer dialogue. Information is displayed and the user response of the computer takes action or displays further information.

**USER\_INITIATED INTERGFACES**

User initiated interfaces fall into tow approximate classes:

1. Command driven interfaces: In this type of interface the user inputs commands or queries which are interpreted by the computer.
2. Forms oriented interface: The user calls up an image of the form to his/her screen and fills in the form. The forms oriented interface is chosen because it is the best choice.

**COMPUTER-INITIATED INTERFACES**

The following computer – initiated interfaces were used:

1. The menu system for the user is presented with a list of alternatives and the user chooses one; of alternatives.
2. Questions – answer type dialog system where the computer asks question and takes action based on the basis of the users reply.

Right from the start the system is going to be menu driven, the opening menu displays the available options. Choosing one option gives another popup menu with more options. In this way every option leads the users to data entry form where the user can key in the data.

**ERROR MESSAGE DESIGN:**

The design of error messages is an important part of the user interface design. As user is bound to commit some errors or other while designing a system the system should be designed to be helpful by providing the user with information regarding the error he/she has committed.

This application must be able to produce output at different modules for different inputs.

**Machine Learning :**

Python is a general-purpose high level programming language that is being increasingly used in data science and in designing machine learning algorithms. This tutorial provides a quick introduction to Python and its libraries like numpy, pandas, matplotlib and explains how it can be applied to develop machine learning algorithms that solve real world problems.

An introduction to machine learning and the Python language and shows you how to setup Python and its packages.

It further covers all important concepts such as exploratory data analysis, data preprocessing, feature extraction,clustering,and model performance evaluation. This also provides various projects that teaches you the techniques and functionalities such as news topic classification,prediction, stock prices forecast and other several important machine learning algorithms.

**Monte Carlo method:**

Monte-Carlo methods: I have been invented in the context of the development of the atomic bomb in the 1940’s I are a class of computational algorithms I can be applied to vast ranges of problems I are not a statistical tool I rely on repeated random sampling I provide generally approximate solutions I are used in cases where analytical or numerical solutions don’t exist or are too difficult to implement I can be used by the Lazy ScientistTM even when an analytical or numerical solution can be implemented Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Overview of the method Monte-Carlo methods generally follow the following steps: 1. Determine the statistical properties of possible inputs 2. Generate many sets of possible inputs which follows the above properties 3. Perform a deterministic calculation with these sets 4. Analyze statistically the results The error on the results typically decreases as 1/ √ N Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Outline What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Numerical integration Most problems can be solved by integration Monte-Carlo integration is the most common application of Monte-Carlo methods Basic idea: Do not use a fixed grid, but random points, because: 1. Curse of dimensionality: a fixed grid in D dimensions requires N D points 2. The step size must be chosen first Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Error estimation Given any arbitrary probability distribution and provided one is able to sample properly the distribution with a random variable (i.e., x ∼ f(x)), Monte-Carlo simulations can be used to: I determine the distribution properties (mean, variance,. . . ) I determine confidence intervals, i.e. P(x > α) = R ∞ α f(x)dx I determine composition of distributions, i.e. given P(x), find P(h(x)), h(x) = x 2 ; cos(x) − sin(x); . . . Note that these are all integrals! Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Optimization problems Numerical solutions to optimization problems incur the risk of getting stuck in local minima. Monte-Carlo approach can alleviate the problem by permitting random exit from the local minimum and find another, hopefully better minimum Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Numerical simulations I Radiation transfer is Google-wise the main astrophysical application of Monte-Carlo simulations in astrophysics I In particle physics and high-energy astrophysics, many more physical processes can be simulated Some physical processes are discretized and random by nature, so Monte-Carlo is particularly adapted Monte Carlo Methods Stéphane Paltani What are Monte-Carlo methods? General concepts Applications Simple examples Generation of random variables Markov chains Monte-Carlo Error estimation Numerical integration Optimization Numerical simulations GEANT4 GEANT4 is also used to determine the performance of X-ray and gamma-ray detectors for astrophysics

**3.TOOLS**

**3.1. NumPy**

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

a powerful N-dimensional array object

sophisticated (broadcasting) functions

tools for integrating C/C++ and Fortran code

useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.

**3.2. Pandas**

Pandas is an open source, BSD-licensed library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language.

pandas is a NumFOCUS sponsored project. This will help ensure the success of development of pandas as a world-class open-source project, and makes it possible to donate to the project.

**3.3. MATPLOTLIB**

Matplotlib is a plotting library for the Python programming language and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt or GTK+. There is also a procedural "pylab" interface based on a state machine (like OpenGL),designed to closely resemble that of MATLAB, though its use is discouraged. SciPy makes use of matplotlib.

**4.SOFTWARE REQUIREMENT**

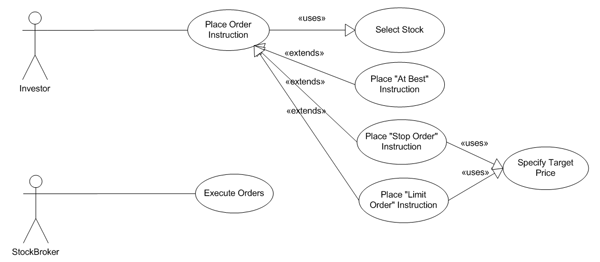
**4.1. Jupyter Notebook**

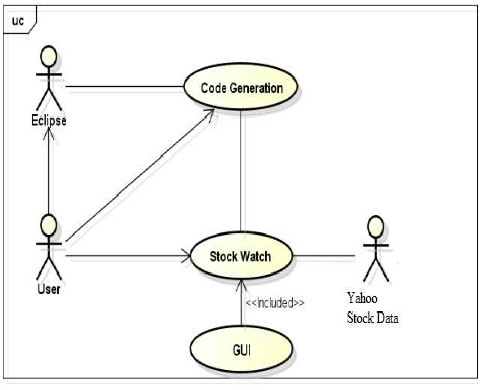
The Jupyter Notebook is an open-source web application that allows you to create and share documents that contain live code, equations, visualizations and narrative text. Uses include: data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.The Jupyter Notebook App is a server-client application that allows editing and running notebook documents via a web browser. The Jupyter Notebook App can be executed on a local desktop requiring no internet access (as described in this document) or can be installed on a remote server and accessed through the internet. In addition to displaying/editing/running notebook documents, the Jupyter Notebook App has a “Dashboard” (Notebook Dashboard), a “control panel” showing local files and allowing to open notebook documents or shutting down their kernels.

**5.PROBLEMS FOR THE ANALYSIS**

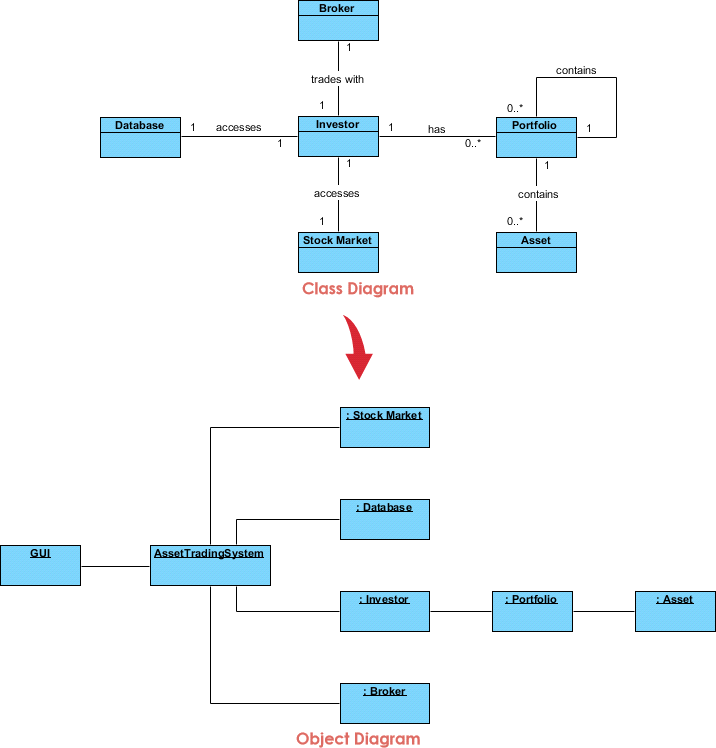
* Import the financial data using quandl for the following companies:
* Apple
* Amazon
* Microsoft
* Google
* Perform fundamental data analysis
* Fetch the last one year’s data
* View the values of Apple’s stock
* Display the plot of closing price
* Display the stock trade by volume
* Plot all companies’ data together for closing price
* Perform Daily Return Analysis and show the relationship between different stocks
* Plot the percentage change plot for Apple’s stock
* Show a joint plot for Apple and Google
* Use PairPlot to show the correlation between all the stocks
* Perform risk analysis

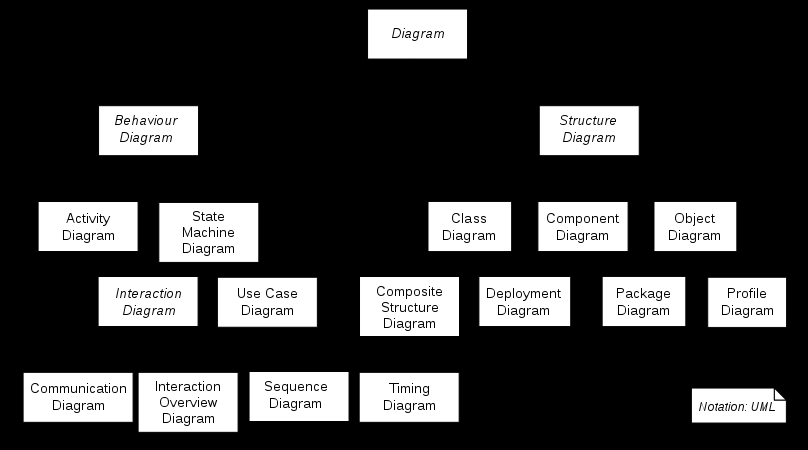
**UML DIAGRAMS**:





**Class Diagrams**:

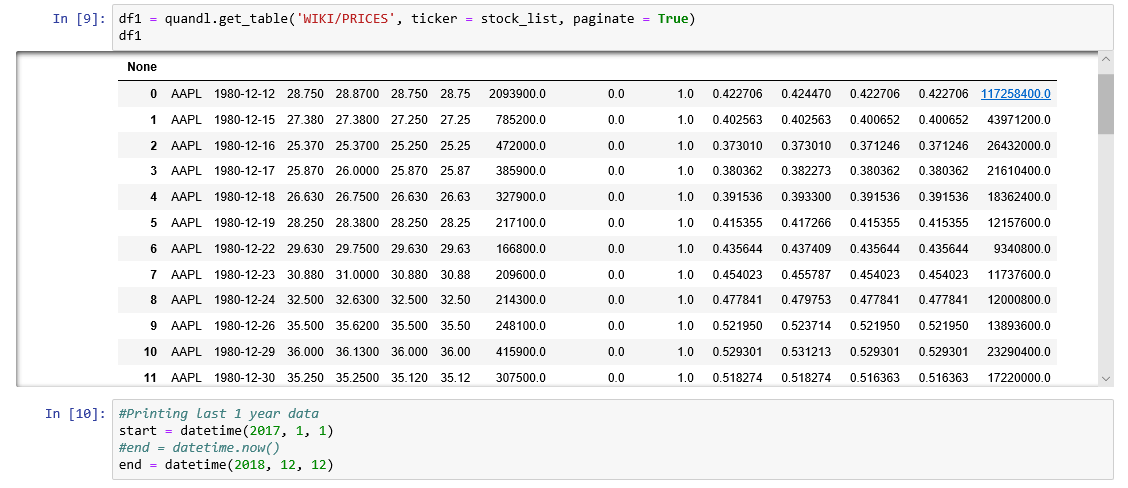




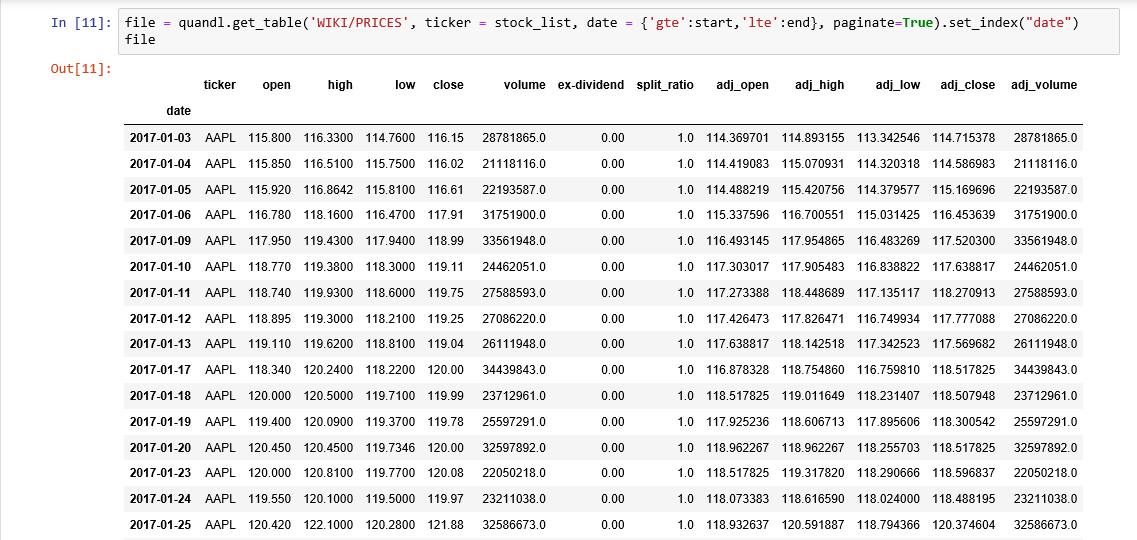
**6.CODE WITH OUTPUT**



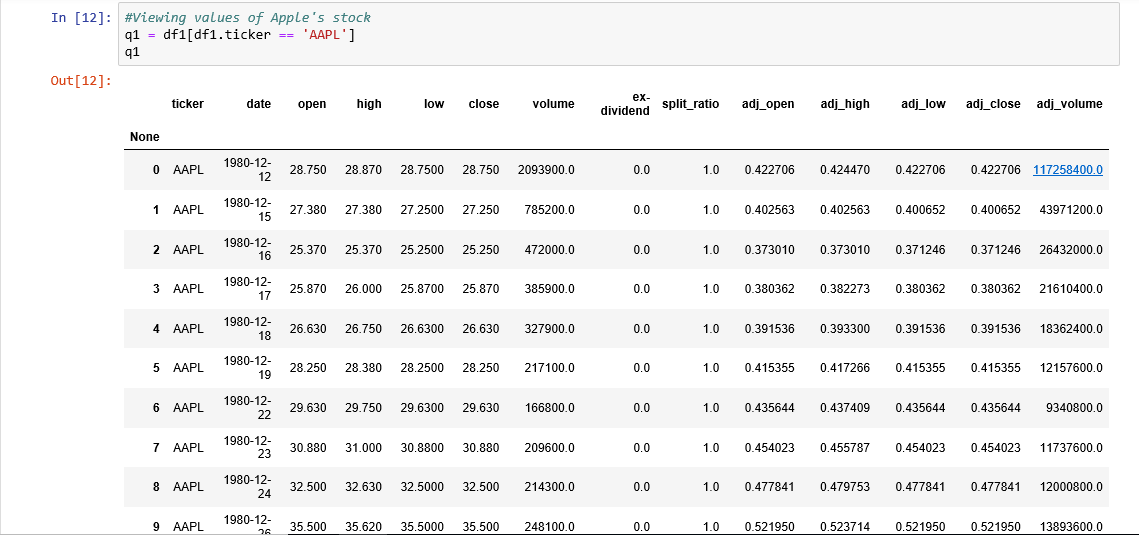
* **Importing the financial data using quandl:**



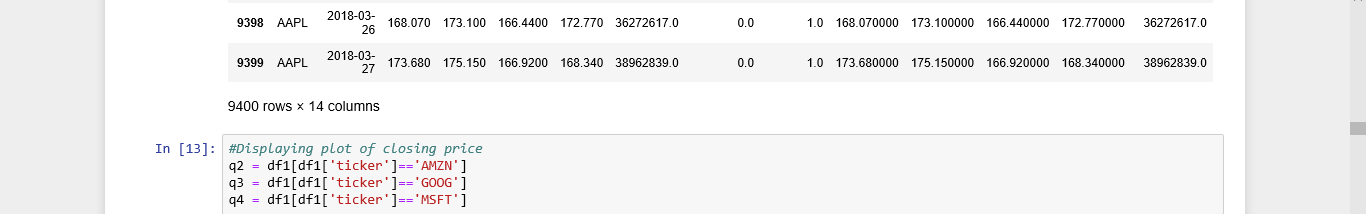
* **Fetching last one year’s data:**

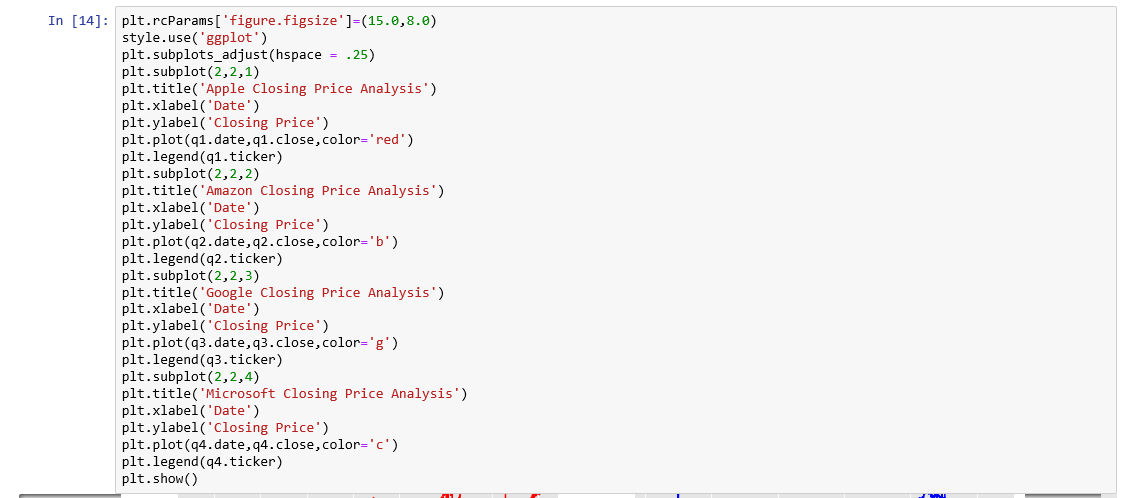


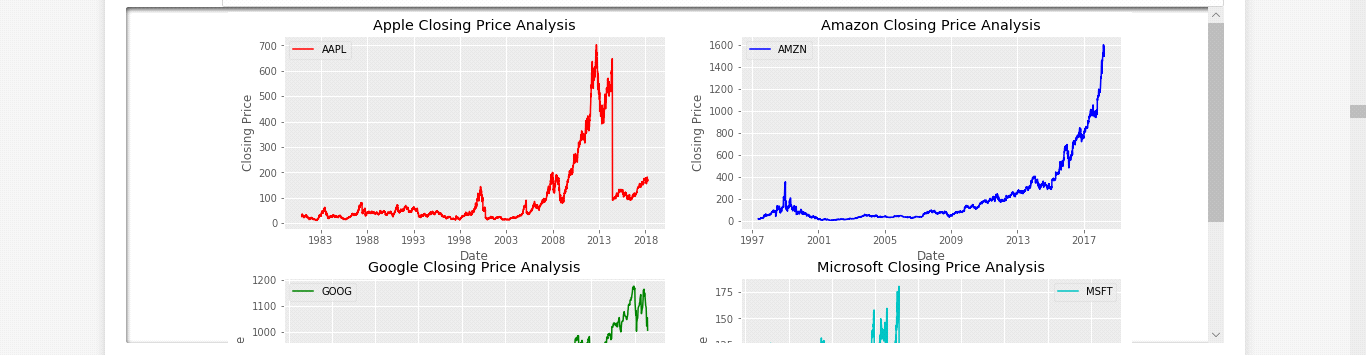
* **Viewing the values of Apple’s Stock:**



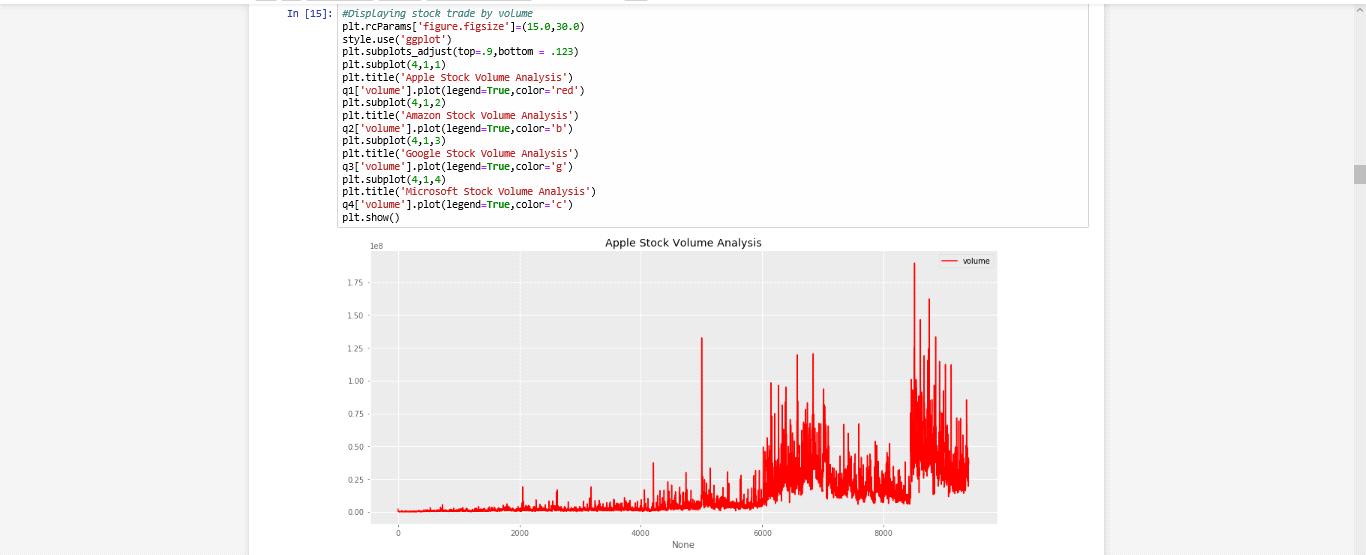
* **Displaying the plot of closing price:**

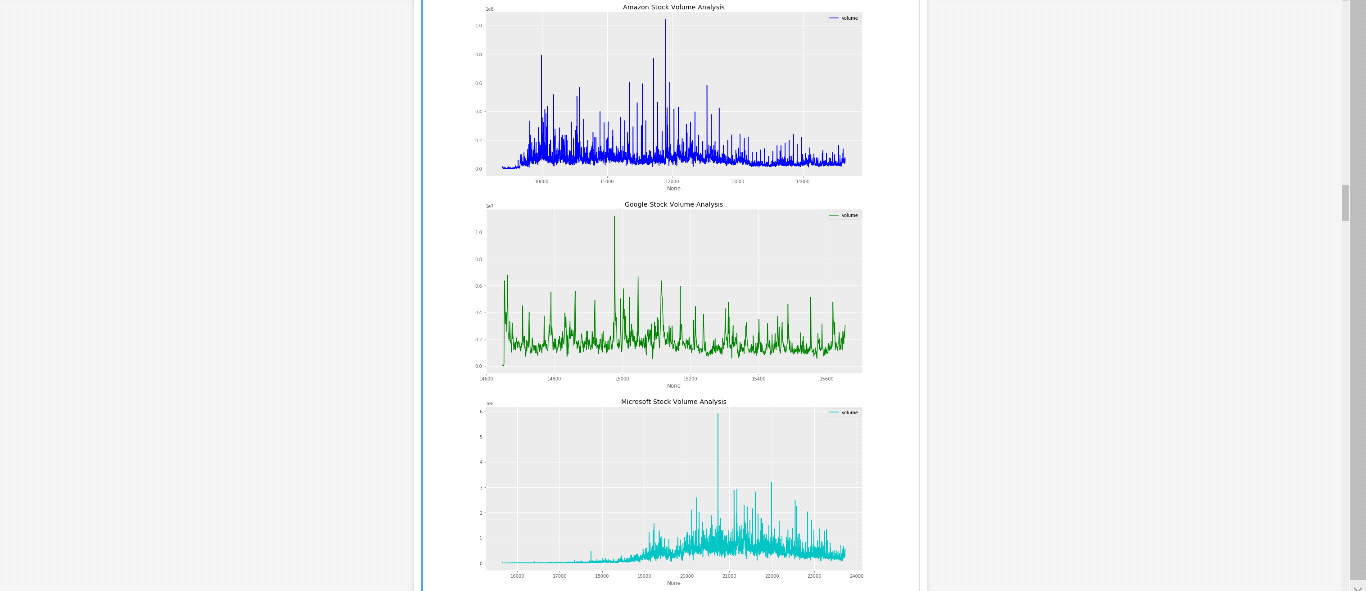




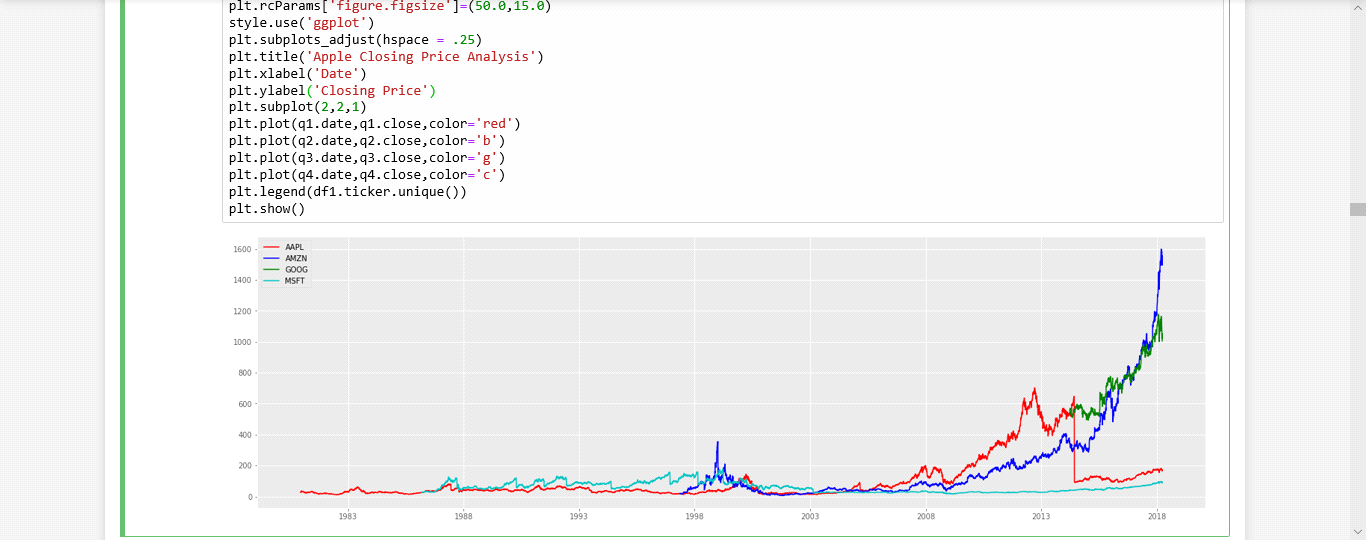


* **Displaying the Stock trade by volume:**

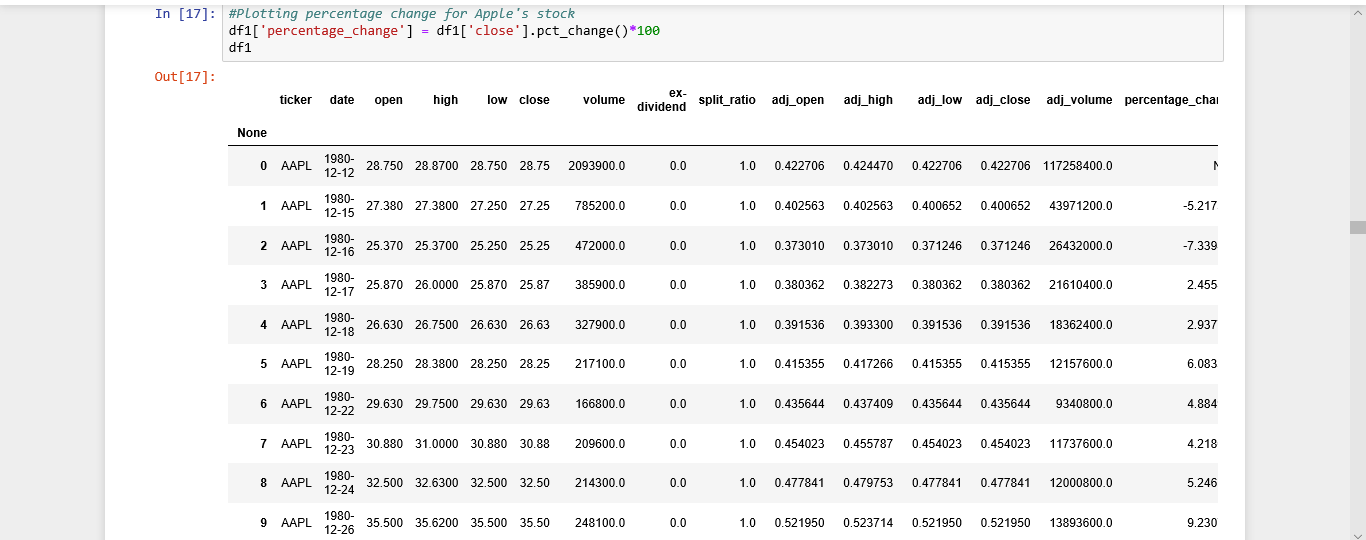


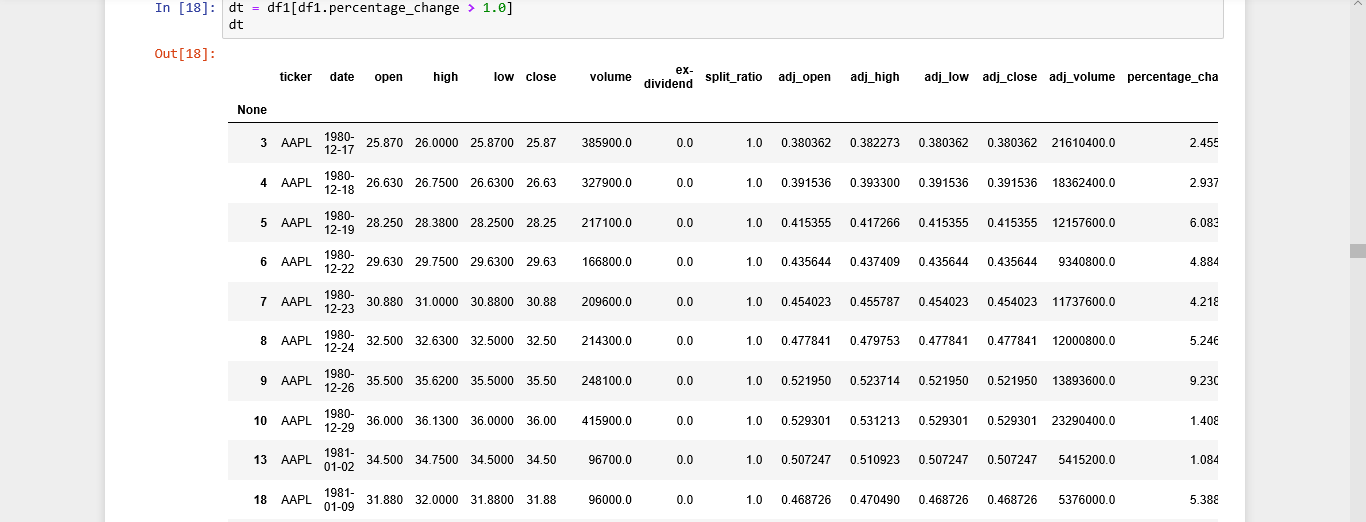


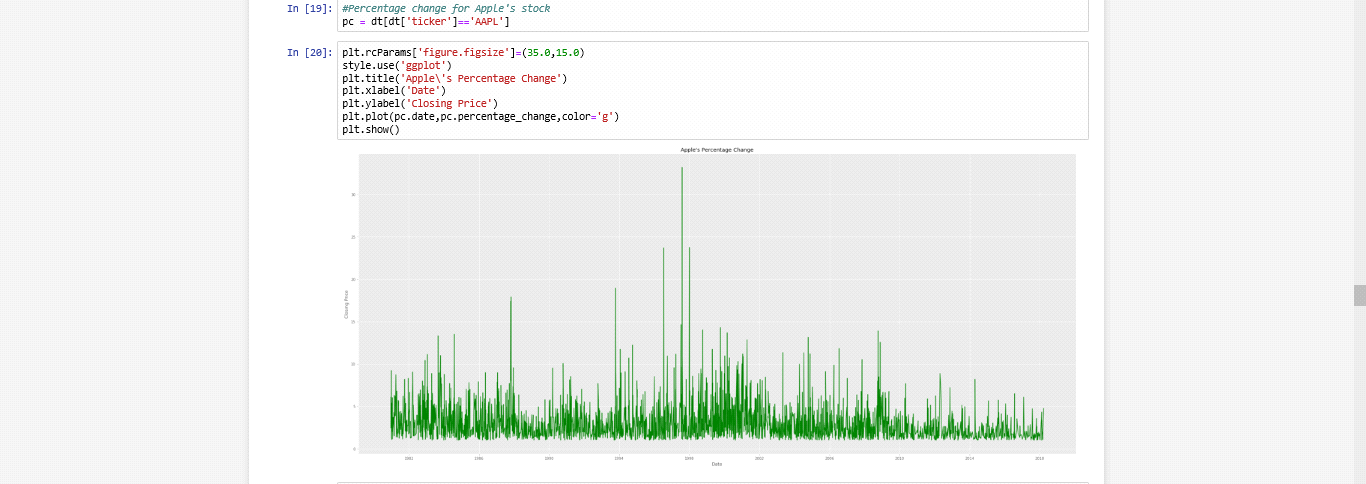
* **Plotting all companies’ data together for closing prices:**



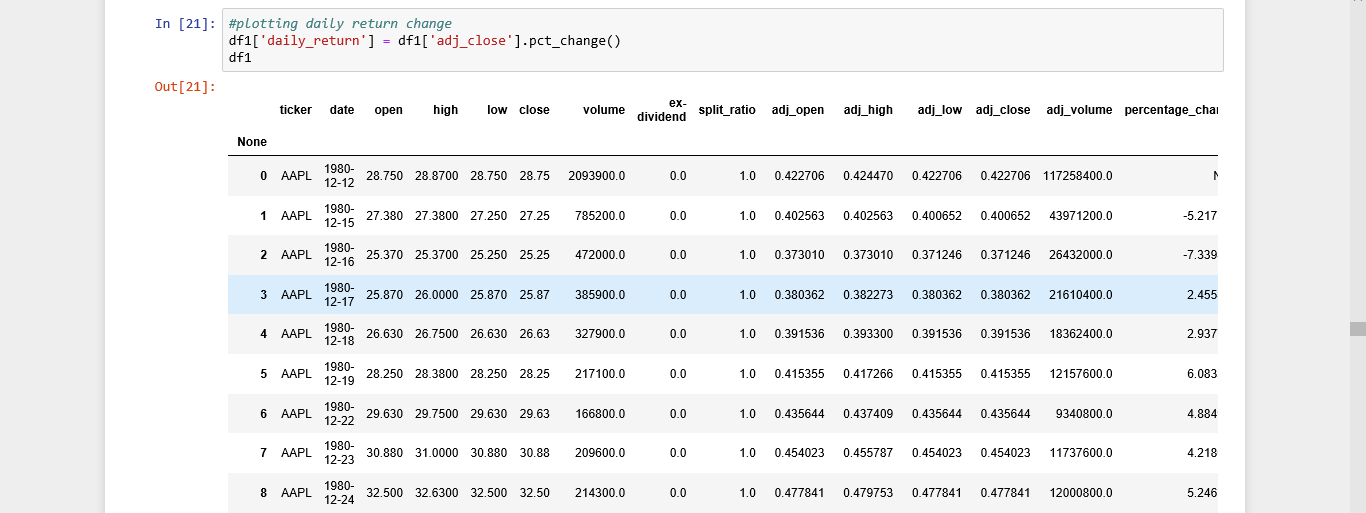
* **Plotting the percentage change plot for Apple’s Stock:**

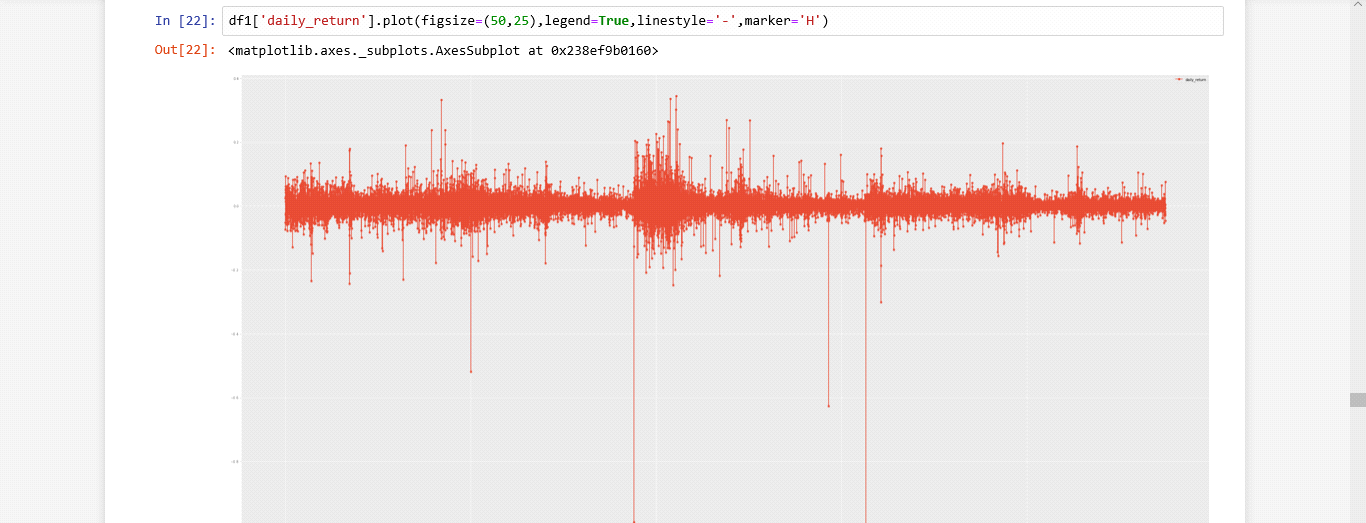




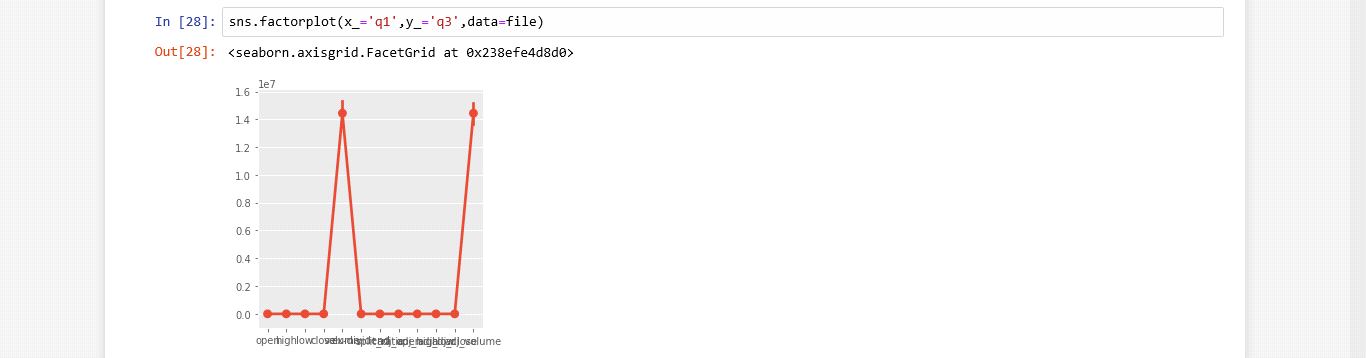


* **Plotting daily return change:**





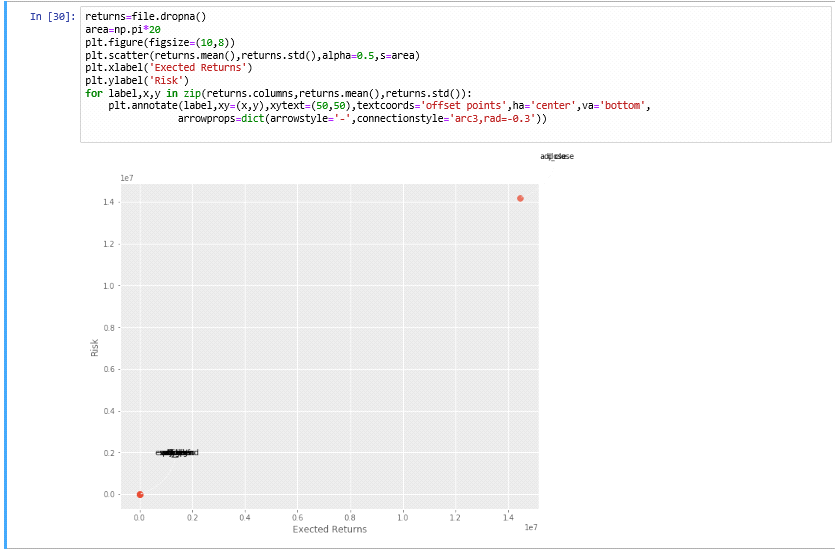
* **Showing a joint lot for Apple and Google:**



* **Showing PairPlot correlation between all the stocks:**



* **Performing risk analysis:**



**7. CONCLUSION AND FUTURE SCOPE**

We evaluated two Fundamental Analysis metrics and found no conclusive proof of their predictive value.

These predictions are also very long term, looking one year into the future. Predictions on this time scale are not the focus of the project. Instead we will focus on predicting daily trends in the market. Due to these issues that we moved away from Fundamental Analysis.