**Bitcon Crypto currency Prediction**

**LIST OF CONTENTS**

1. **INTRODUCTION**
   1. INTRODUCTION TO PROJECT
   2. PURPOSE OF THE SYSTEM
   3. PROBLEMS IN EXISTING SYSTEM
   4. SOLUTION OF THESE PROBLEMS
2. **SYSTEM ANALYSIS**
   1. INTRODUCTION
   2. STUDY OF THE SYSTEM
   3. HARDWARE AND SOFTWARE REQUIRMENTS
   4. PROPOSED SYSTEM
   5. INPUT AND OUTPUT
   6. PROCESS MODULES USED WITH JUSTIFICATION
3. **FEASIBILITY REPORT**
   1. TECHNICAL FEASIBILITY
   2. OPERATIONAL FEASIBILITY
   3. ECONOMICAL FEASIBILTY
4. **SOFTWARE REQUIREMENT SPECIFICATIONS**
   1. FUNCTIONAL REQUIREMENTS
   2. PERFORMANCE REQUIREMENTS

**5. SELECTED SOFTWARE**

5.1. INTRODUCTION TO PYTHON

5.2. PYTHON

**6. SYSTEM DESIGN**

6.1. INTRODUCTION

6.2. SYSTEM WORK FLOW

6.3. UML DIAGRAMS

**7. SYSTEM TESTING AND VALIDATION**

7.1.INTRODUCTION

7.2. STRATEGIC APPROACH OF SOFTWARE TESTING

7.3. UNIT TESTING

**8. SYSTEM SECURITY**

8.1. INTRODUCTION

* 1. SECURITY IN SOFTWARE

**9. OUTPUT SCREENS**

**10. CONCLUSION**

**11. BIBLIOGRAPHY**

**Abstract**

In this project, we tried to estimate the Bit coin price precisely taking into consideration various parameters that affect the Bit coin value. In our work, we pointed to understand and identify daily changes in the Bit coin market while obtaining insight into most appropriate features surrounding Bit coin price. We will predict the daily price change with highest possible accuracy. The market capitalization of publicly traded crypto currencies is currently above $230 billion.Bitcoin, the most valuable crypto currency, serves primarily as a digital store of value, and its price predictability has been well-studied. These characteristics are outlined in the following subsection; the underlying details of Bitcoin,as they are described in depth in the cited papers.

The LSTM achieves the highest classification accuracy of 52% and a RMSE of 8%. The popular ARIMA model for time series forecasting is implemented as a comparison to the deep learning models. As expected, the non-linear deep learning methods outperform the ARIMA forecast which performs poorly.

**1. INTRODUCTION**

**1.1 INTRODUCTION:**

**2. LITERATURE SURVEY**

Bitcoin is the worlds’ most valuable cryptocurrency and is traded on over 40 exchanges worldwide accepting over 30 different currencies. It has a current market capitalization of 9 billion USD according to <https://www.blockchain.info/> and sees over 250,000 transactions taking place per day. As a currency, Bitcoin offers a novel opportunity for price prediction due its relatively young age and resulting volatility,

which is far greater than that of fiat currencies [2]. It is also unique in relation to traditional fiat currencies in terms of its open nature; no complete data exists regarding cash transactions or money in circulation for fiat currencies. Prediction of mature financial markets such as the stock market has been researched at length [3], [4]. Bitcoin presents an interesting parallel to this as it is a time series prediction problem in a market still in its transient stage. Traditional time series prediction methods such as Holt-Winters exponential smoothing models rely on linear assumptions and require data that can be broken down into trend, seasonal and noise tobe effective. This type of methodology is more suitablem for a task such as forecasting sales where seasonal effects are present. Due to the lack of seasonality in the Bitcoin market and its high volatility, these methods are not very effective for this task. Given the complexity of the task, deep learning makes for an interesting technological solution based on its performance in similar areas. The recurrent neural network (RNN) and the long short term memory (LSTM) are favoured over the traditional multilayer perceptron (MLP) due to the temporal nature of Bitcoin data.

The aim of this paper is to investigate with what accuracy the price of Bitcoin can be predicted using machine learning and compare parallelisation methods executed on multi-core and GPU environments. This paper contributes in the following manner: of approximately 653 papers published on Bitcoin [6], only 7 (at the time of writing) are related to machine learning for prediction. To facilitate a comparison to more traditional approaches in financial forecasting, an ARIMA time series model is also developed for performance comparison purposes with the neural network models.

The independent variable for this study is the closing price of Bitcoin in USD taken from the Coindesk Bitcoin Price Index. Rather than focusing on one specific exchange, we take the average price from five major Bitcoin exchanges: Bitstamp, Bitfinex, Coinbase, OkCoin and itBit. If we were to implement trades based on the signals it would be beneficial to focus on just one exchange. To assess the performance of models, we use the root mean squared error (RMSE) of the closing price and further encode the predicted price into categorical variable reflecting: price up, down or no change. This latter step allows for additional performance metrics that would be useful to a trader in the formation of a trading strategy: classification accuracy, specificity, sensitivity and precision. The dependent variables for this paper come from the Coindesk website, and Blockchain.info. In addition to the closing price, the opening price, daily high and daily low are also included as well as Blockchain data, i.e. the mining difficulty and hash rate. The features which have been engineered (considered as technical analysis indicators [7]) include two simple moving averages (SMA) and a de-noised closing price.

.

**3. SYSTEM ANALYSIS**

The Systems Development Life Cycle (SDLC), or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems. In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies).

**3.1 EXISTING SYSTEM**:

Research on predicting the price of Bitcoin using machine learning algorithms specifically is lacking. implemented a latent source model as developed by [9] to predict the price of Bitcoin noting 89% return in 50 days with a Sharpe ratio of 4.1. There has also been work using text data from social media platforms and other sources to predict Bitcoin prices. investigated sentiment analysis using support vector machines coupled with the frequency of Wikipedia views, and the network hash rate. investigated the relationship between Bitcoin price, tweets and views for Bitcoin on Google Trend

**Disadvantages:**

However, one limitation of such studies is the often small sample size, and propensity for misinformation to spread through various (social) media channels such as Twitter or on message boards such as Reddit, which artificially inflate/deflate prices.

In the Bitcoin exchanges liquidity is considerably limited. As a result, the market suffers from a greater risk of manipulation. For this reason, sentiment from social media is not considered further

**3.3 PROPOSED SYSTEM:**

Appropriate design of deep learning models in terms of network parameters is imperative to their success. The three main options available when choosing how to select parameters for deep learning models are random search, grid search and heuristic search methods such as genetic algorithms. Manual grid search and Bayesian optimisation were utilised in this study. Grid search, implemented for the Elman RNN, is the process of selecting two hyper paramaters with a minimum and maximum for each. One then searches that feature space looking for the best performing parameters.

This approach was taken for parameters which were unsuitable for Bayesian optimisation. This model was built using Keras in the Python programming language . Similar to the RNN, Bayesian optimization was chosen for selecting LTSM parameters where possible. This is a heuristic search method which works by assuming the function was sampled from a Gaussian process and maintains a posterior distribution for this function as the results of different hyper parameter selections are observed. One can then optimise the expected improvement over the best result to pick hyper parameters for the next experiment. The performance of both the RNN and LSTM network are evaluated on validation data with measures to prevent overfitting. Dropout is implemented in both layers, and we automatically stop model training if its validation loss hasn’t improved in 5 epochs.

**Advantages:**

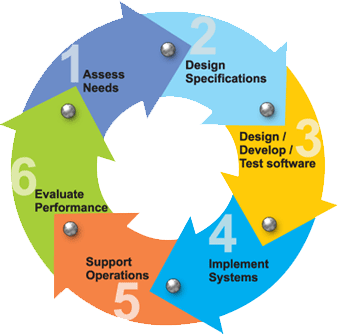
LSTM models converged between 50 and 100 epochs with early stopping. Similar to the RNN, batch size was found to have a greater effect on execution time than accuracy. This may be due to the relatively small size of the dataset.

**4. IMPLEMENTATION**

**2.1 INTRODUCTION**

**Software Development Life Cycle:**

There is various software development approaches defined and designed which are used/employed during development process of software, these approaches are also referred as "Software Development Process Models". Each process model follows a particular life cycle in order to ensure success in process of software development.



**Requirements:**

Business requirements are gathered in this phase.  This phase is the main focus of the project managers and stake holders.  Meetings with managers, stake holders and users are held in order to determine the requirements.  Who is going to use the system?  How will they use the system?  What data should be input into the system?  What data should be output by the system?  These are general questions that get answered during a requirements gathering phase.  This produces a nice big list of functionality that the system should provide, which describes functions the system should perform, business logic that processes data, what data is stored and used by the system, and how the user interface should work.  The overall result is the system as a whole and how it performs, not how it is actually going to do it.

**Design:**

The software system design is produced from the results of the requirements phase.  Architects have the ball in their court during this phase and this is the phase in which their focus lies.  This is where the details on how the system will work is produced.  Architecture, including hardware and software, communication, software design (UML is produced here) are all part of the deliverables of a design phase.

**Implementation:**

Code is produced from the deliverables of the design phase during implementation, and this is the longest phase of the software development life cycle.  For a developer, this is the main focus of the life cycle because this is where the code is produced.  Implementation my overlap with both the design and testing phases.  Many tools exists (CASE tools) to actually automate the production of code using information gathered and produced during the design phase.

**Testing:**

During testing, the implementation is tested against the requirements to make sure that the product is actually solving the needs addressed and gathered during the requirements phase.  Unit tests and system/acceptance tests are done during this phase.  Unit tests act on a specific component of the system, while system tests act on the system as a whole.

So in a nutshell, that is a very basic overview of the general software development life cycle model.  Now let’s delve into some of the traditional and widely used variations.

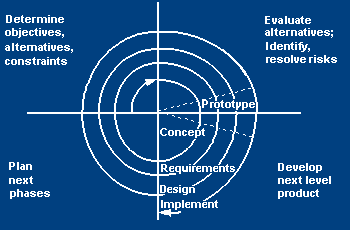
**SDLC METHDOLOGIES:**

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 usersrepresenting 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:

1. Evaluating the first prototype in terms of its strengths, weakness, and risks.
2. Defining the requirements of the second prototype.
3. Planning a designing the second prototype.
4. 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 involve 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.

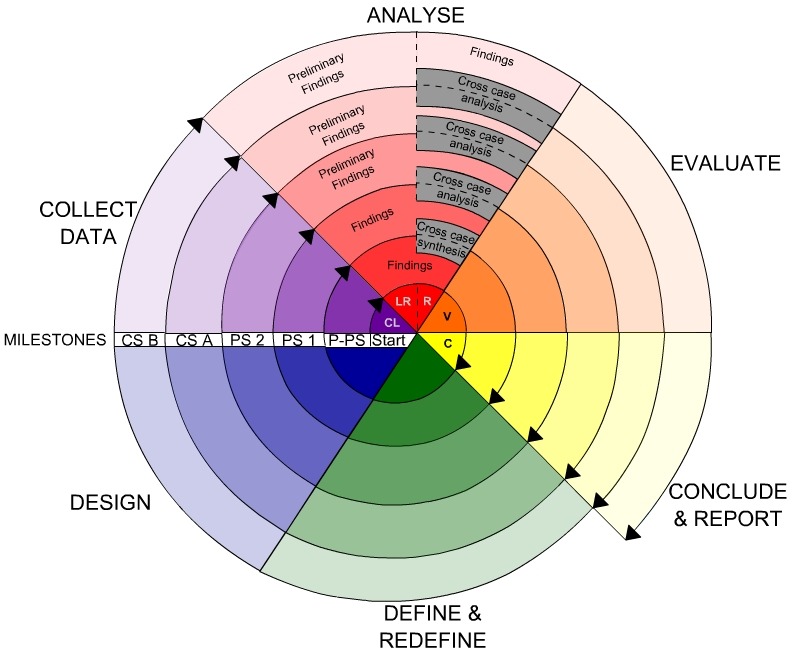
**2.2 STUDY OF THE SYSTEM**

In the flexibility of uses the interface has been developed a graphics concepts in mind, associated through a browser interface. The GUI’s at the top level has been categorized as follows

1. Administrative User Interface Design
2. The Operational and Generic User Interface Design

The administrative user interface concentrates on the consistent information that is practically, part of the organizational activities and which needs proper authentication for the data collection. The Interface helps the administration with all the transactional states like data insertion, data deletion, and data updating along with executive data search capabilities.

The operational and generic user interface helps the users upon the system in transactions through the existing data and required services. The operational user interface also helps the ordinary users in managing their own information helps the ordinary users in managing their own information in a customized manner as per the assisted flexibilities.



**2.3. INPUT AND OUTPUT**

**2.3.1.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.

**INPUTMEDIA:**

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.

**2.3.2.OUTPUT DESIGN**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide 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.

**3. Fundamental Concepts on (Domain)**

The data was also standardised to give it a mean of 0 and standard deviation of 1. Standardisation was chosen over normalisation as it better suits the activation functions used by the deep learning models.

***Feature Engineering and Feature Evaluation***

Feature engineering is the art of extracting useful patterns from data to make it easier for machine learning models to perform their predictions. It can be considered one of the most important parts of the data mining process in order to achieve good results in prediction tasks [27], [28]. Several papers in recent years have included indicators includingthe Simple Moving Average (SMA) for machine learning classification tasks [29], [30]. An example of an appropriate technical indicator is a SMA recording the average price over the previous x days, and is correspondingly included.

To evaluate which features to include, Boruta (a wrapper built around the random forest classification algorithm) was used. This is an ensemble method in which classification is performed by voting of multiple classifiers. The algorithm works on a similar principle as the random forest classifier. It adds randomness to the model and collects results from the ensemble of randomised samples to evaluate attributes and provides a clear view on which attributes are important [31]. All features were deemed important to the model based on the random forest, with 5 day and 10 days (via SMA) the highest

importance among the tested averages. The de-noised closing price was one of the most important variables also.

***Deep Learning Models***

Appropriate design of deep learning models in terms of network parameters is imperative to their success. The three main options available when choosing how to select parameters for deep learning models are random search, grid search and heuristic search methods such as genetic algorithms. Manual grid search and Bayesian optimization were utilized in this study. Grid search, implemented for the Elman RNN, is the process of selecting two hyperparamaters with a minimum and maximum for each. One then searches that feature space looking for the best performing parameters. This approach

was taken for parameters which were unsuitable for Bayesian optimization. This model was built using Keras in the Python

Programming language. Similar to the RNN, Bayesian optimization was chosen for selecting LTSM parameters where possible. This is a heuristic search method which works by assuming the function was sampled from a Gaussian process and maintains a posterior distribution for this function as the

results of different hyper parameter selections are observed. One can then optimize the expected improvement over the best result to pick hyper parameters for the next experiment. The performance of both the RNN and LSTM network are evaluated on validation data with measures to prevent

over fitting. Dropout is implemented in both layers, and we automatically stop model training if its validation loss hasn’t improved in 5 epochs.

**Implementation:**

*RNN* The first parameter to consider was the temporal length window. As suggested by supporting literature these type of networks may struggle to learn long term dependencies using gradient based optimisation. An autocorrelation function (ACF) was run for the closing price time series to assess the relationship between the current closing price and previous or future closing prices. While this is not a guarantee of predictive power for this length, it was a better choice than random choice. Closing price is correlated with a lag of up to 20 days in many cases, with isolated cases at 34, 45 and 47 days. This led the grid search for the temporal window to test from 2 to 20, 34, 45 and 47 days. To ensure a robust search, larger time periods of up to 100 days were also tested in increments of five. The most effective window temporal length was 24.

In addition to the temporal window, some hyperparameters also need tuning: Learning rate is the parameter that guides stochastic gradient descent (SGD), i.e. how the network learns. Similarly, momentum updates the learning rate to avoid the model falling into local minima (in terms of error) and attempts to move towards the global minimum of the error function [35]. We used the RMSprop optimiser to improve on SGD, as it keeps a running average of recent gradients and as a result is more robust against information loss . According to Heaton [37], one hidden layer is enough to approximatethe vast majority of non-linear functions. Two hidden layers were also explored and were chosen as they achieved lower validation error. Heaton also recommends for the number of hidden nodes to select between the number of input and output nodes. In this case, less than 20 nodes per layer resulted in poor performance. 50 and 100 nodes were tested with good performance.

However, too many nodes can increase the chancesof overfitting, and significantly increase the time needed to train the network. As 20 nodes performed sufficiently well this was chosen for the final model. An activation function, a nonlinear stepwise equation that passes signals between layers,is also needed. The options explored were Tanh, ReLu, and

Sigmoid. Tanh performed the best but the differences were not significant. The final parameters for selection are batch size and number of training epochs. Batch size was found to have little effect on accuracy but considerable effect on training time when using smaller batches in this case. The number of epochs tested ranged from 10 to 10000, however,too many training epochs can result in overfitting. To reduce the risk of overfitting, dropout was implemented as discussedabove. Optimal dropout between 0.1 and 1 was searched for both layers with .5 dropout the optimal solution for both layers. A Keras callback method was also used to stop the training of the model if its performance on validation data did not improve after 5 epochs to prevent overfitting. Generally, the RNN converged between 20 and 40 epochs with early stopping.

*LSTM*

In terms of temporal length, the LSTM is considerably better at learning long term dependencies. As a result, picking a long window was less detrimental for the LSTM. This process followed a similar process to the RNN in which autocorrelation lag was used as a guideline. The LSTM performed poorly on smaller window sizes. Its most effective length found was 100 days, and two hidden LSTM layers were chosen. For a time series task two layers is enough to find nonlinear relationships among the data. 20 hidden nodes were also chosen for both layers as per the RNN model. The Hyperaslibrary2 was used to implement the Bayesian optimisation of the network parameters. The optimiser searched for the optimal model in terms of how much dropout per layer and which optimizer to use.

RMSprop again performed the best for this task. In the LSTM model, activation functions weren’t changed as the LSTM has a particular sequence of tanh and sigmoid activation functions for the different gates within the cell. LSTM models converged between 50 and 100 epochs with early stopping. Similar to the RNN, batch size was found to have a greater effect on execution time than accuracy. This may be due to the relatively small size of the dataset.

*Model Comparison*

A confusion matrix representing the ratio of true/false and positive/negative classifications is used to derive the ratings metrics. Accuracy can be defined as the total number of correctly classified predictions (price up, down, and no change). To combat inherent class imbalance (bitcoin price predominately increases) the metrics sensitivity, specificityand precision are also analysed. Sensitivity represents how good a model is at detecting positives. Specificity represent how good the model is at avoiding false alarms. Finally, precision represents how many positively classified predictions were relevant. Root Mean Square Error (RMSE) is used to evaluate and compare the regression accuracy. To instrument the evaluation of models, a 80/20 holdout validation strategy is used.

In order to facilitate a comparison of the deep learning methods to more traditional methods we built (and optimised) an ARIMA model, as they have been extensively used in price prediction problems . The ARIMA forecast wascreated by splitting the data into 5 periods and then predicting 30 days into the future. The data was differenced before being fit with several ARIMA models. The best fit was found byauto.arima from the R forecast package.

**4. System Analysis**

The **Systems Development Life Cycle (SDLC)**, or Software Development Life Cycle in [systems engineering](http://en.wikipedia.org/wiki/Systems_engineering), [information systems](http://en.wikipedia.org/wiki/Information_systems) and [software engineering](http://en.wikipedia.org/wiki/Software_engineering), is the process of creating or altering systems, and the models and [methodologies](http://en.wikipedia.org/wiki/Methodologies) that people use to develop these systems.

In software engineering the SDLC concept underpins many kinds of [software development methodologies](http://en.wikipedia.org/wiki/Software_development_methodologies). These methodologies form the framework for planning and controlling the creation of an information system the [software development process](http://en.wikipedia.org/wiki/Software_development_process).

**SOFTWARE MODEL OR ARCHITECTURE ANALYSIS:**

Structured project management techniques (such as an SDLC) enhance management’s control over projects by dividing complex tasks into manageable sections. A software life cycle model is either a descriptive or prescriptive characterization of how software is or should be developed. But none of the SDLC models discuss the key issues like Change management, Incident management and Release management processes within the SDLC process, but, it is addressed in the overall project management. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. In the proposed hypothetical model, the concept of user-developer interaction in the conventional SDLC model has been converted into a three dimensional model which comprises of the user, owner and the developer. The ―one size fits all‖ approach to applying SDLC methodologies is no longer appropriate. We have made an attempt to address the above mentioned defects by using a new hypothetical model for SDLC described elsewhere. The drawback of addressing these management processes under the overall project management is missing of key technical issues pertaining to software development process that is, these issues are talked in the project management at the surface level but not at the ground level.

**2.5Functional requirements**

Outputs from computer systems are required primarily to communicate the results of processing to users. They are also used to provide 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.
* Understanding user’s preferences, expertise level and his business requirements through a friendly questionnaire.
* Input data can be in four different forms - Relational DB, text files, .xls and xml files. For testing and demo you can choose data from any domain. User-B can provide business data as input.

**Non-Functional Requirements:**

1. Secure access of confidential data (user’s details). SSL can be used.
2. 24 X 7 availability.
3. Better component design to get better performance at peak time
4. Flexible service based architecture will be highly desirable for future extension

**FEASIBILITY STUDY**

**FEASIBILITY STUDY:**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Economical Feasibility
* Operation Feasibility

**2.1. Technical Feasibility:**

In the feasibility study first step is that the organization or company has to decide that what technologies are suitable to develop by considering existing system.

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipment have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

Here in this application used the technologies like Visual Studio 2012 and SqlServer 2014. These are free software that would be downloaded from web.

Visual Studio 2013 –it is tool or technology.

**2.2. ECONOMICAL FEASIBILITY**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

### Determining Economic Feasibility:

Assessing the economic feasibility of an implementation by performing a cost/benefit analysis, which as its name suggests compares the full/real costs of the application to its full/real financial benefits.  The alternatives should be evaluated on the basis of their contribution to net cash flow, the amount by which the benefits exceed the costs, because the primary objective of all investments is to improve overall organizational performance.

|  |  |  |
| --- | --- | --- |
| **Type** | **Potential Costs** | **Potential Benefits** |
| Quantitative | * Hardware/software upgrades * Fully-burdened cost of labor (salary + benefits) * Support costs for the application * Expected operational costs * Training costs for users to learn the application * Training costs to train developers in new/updated technologies | * Reduced operating costs * Reduced personnel costs from a reduction in staff * Increased revenue from additional sales of your organizations products/services |
| Qualitative | * Increased employee dissatisfaction from fear of change | * Improved decisions as the result of access to accurate and timely information * Raising of existing, or introduction of a new, barrier to entry within your industry to keep competition out of your market * Positive public perception that your organization is an innovator |

 The table includes both qualitative factors, costs or benefits that are subjective in nature, and quantitative factors, costs or benefits for which monetary values can easily be identified.  I will discuss the need to take both kinds of factors into account when performing a cost/benefit analysis.

**2.3. OPERATIONAL FEASIBILITY**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

Not only must an application make economic and technical sense, it must also make operational sense.

|  |  |
| --- | --- |
| **Operations Issues** | **Support Issues** |
| * What tools are needed to support operations? * What skills will operators need to be trained in? * What processes need to be created and/or updated? * What documentation does operations need? | * What documentation will users be given? * What training will users be given? * How will change requests be managed? |

Very often you will need to improve the existing operations, maintenance, and support infrastructure to support the operation of the new application that you intend to develop.  To determine what the impact will be you will need to understand both the current operations and support infrastructure of your organization and the operations and support characteristics of your new application. To operate this application END-TO-END VMS. The user no need to require any technical knowledge that we are used to develop this project is Asp.net C#.net. That the application providing rich user interface by user can do the operation in flexible manner.

**SELECTED SOFTWARE**

**IMPLEMENTATION ON (PYTHON):**

**What Is A Script?**

Up to this point, I have concentrated on the interactive programming capability of Python.  This is a very useful capability that allows you to type in a program and to have it executed immediately in an interactive mode

**Scripts are reusable:**

Basically, a script is a text file containing the statements that comprise a Python program.  Once you have created the script, you can execute it over and over without having to retype it each time.

**Scripts are editable:**

Perhaps, more importantly, you can make  different versions of the script by modifying the statements from one file to the next using a text editor.  Then you can execute each of the individual versions.  In this way, it is easy to create different programs with a minimum amount of typing.

**You will need a text editor:**

Just about any text editor will suffice for creating Python script files.

You can use Microsoft Notepad, Microsoft WordPad, Microsoft Word, or just about any word processor if you want to.

**Difference between a script and a program**

**Script:** Scripts are distinct from the core code of the application, which is usually written in a different language, and are often created or at least modified by the end-user. Scripts are often interpreted from source code or byte code, where as the applications they control are traditionally compiled to native machine code.

**Program:** The program has an executable form that the computer can use directly to execute the instructions.

The same program in its human-readable source code form, from which executable programs are derived(e.g., compiled)

**Python**

what is Python? Chances you are asking yourself this. You may have found this book because you want to learn to program but don’t know anything about programming languages. Or you may have heard of programming languages like C, C++, C#, or Java and want to know what Python is and how it compares to “big name” languages. Hopefully I can explain it for you.

**Python concepts**

If you not interested in the how and whys of Python, feel free to skip to the next chapter. In this chapter I will try to explain to the reader why I think Python is one of the best languages available and why it’s a great one to start programming with.

• Open source general-purpose language.

• Object Oriented, Procedural, Functional

• Easy to interface with C/ObjC/Java/Fortran

• Easy-ish to interface with C++ (via SWIG)

• Great interactive environment

Python is a high-level, interpreted, interactive and object-oriented scripting language. Python is designed to be highly readable. It uses English keywords frequently where as other languages use punctuation, and it has fewer syntactical constructions than other languages.

* **Python is Interpreted** − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* **Python is Interactive** − You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* **Python is Object-Oriented** − Python supports Object-Oriented style or technique of programming that encapsulates code within objects.
* **Python is a Beginner's Language** − Python is a great language for the beginner-level programmers and supports the development of a wide range of applications from simple text processing to WWW browsers to games.

**History of Python**

Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.

Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, SmallTalk, and Unix shell and other scripting languages.

Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).

Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**Python Features**

Python's features include −

* **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.

Apart from the above-mentioned features, Python has a big list of good features, few are listed below −

* 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.

**Dynamic vs Static**

Types Python is a dynamic-typed language. Many other languages are static typed, such as C/C++ and Java. A static typed language requires the programmer to explicitly tell the computer what type of “thing” each data value is.

For example, in C if you had a variable that was to contain the price of something, you would have to declare the variable as a “float” type.

This tells the compiler that the only data that can be used for that variable must be a floating point number, i.e. a number with a decimal point.

If any other data value was assigned to that variable, the compiler would give an error when trying to compile the program.

Python, however, doesn’t require this. You simply give your variables names and assign values to them. The interpreter takes care of keeping track of what kinds of objects your program is using. This also means that you can change the size of the values as you develop the program. Say you have another decimal number (a.k.a. a floating point number) you need in your program.

With a static typed language, you have to decide the memory size the variable can take when you first initialize that variable. A double is a floating point value that can handle a much larger number than a normal float (the actual memory sizes depend on the operating environment).

If you declare a variable to be a float but later on assign a value that is too big to it, your program will fail; you will have to go back and change that variable to be a double.

With Python, it doesn’t matter. You simply give it whatever number you want and Python will take care of manipulating it as needed. It even works for derived values.

For example, say you are dividing two numbers. One is a floating point number and one is an integer. Python realizes that it’s more accurate to keep track of decimals so it automatically calculates the result as a floating point number

**Variables**

Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.

Based on the data type of a variable, the interpreter allocates memory and decides what can be stored in the reserved memory. Therefore, by assigning different data types to variables, you can store integers, decimals or characters in these variables.

**Standard Data Types**

The data stored in memory can be of many types. For example, a person's age is stored as a numeric value and his or her address is stored as alphanumeric characters. Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.

Python has five standard data types −

* Numbers
* String
* List
* Tuple
* Dictionary

## Python Numbers

Number data types store numeric values. Number objects are created when you assign a value to them

## Python Strings

Strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.

## Python Lists

Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.

The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

**Python Tuples**

A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.

The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as **read-only** lists.

## Python Dictionary

Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.

Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).

**Different modes in python**

Python has two basic modes: normal and interactive.

The normal mode is the mode where the scripted and finished .py files are run in the Python interpreter.

Interactive mode is a command line shell which gives immediate feedback for each statement, while running previously fed statements in active memory. As new lines are fed into the interpreter, the fed program is evaluated both in part and in whole

# 20 Python libraries

**1.** Requests. The most famous http library written by kenneth reitz. It’s a must have for every python developer.

**2.** Scrapy. If you are involved in webscraping then this is a must have library for you. After using this library you won’t use any other.

**3.** wxPython. A gui toolkit for python. I have primarily used it in place of tkinter. You will really love it.

**4.** Pillow. A friendly fork of PIL (Python Imaging Library). It is more user friendly than PIL and is a must have for anyone who works with images.

**5.** SQLAlchemy. A database library. Many love it and many hate it. The choice is yours.

**6.** BeautifulSoup. I know it’s slow but this xml and html parsing library is very useful for beginners.

**7.** Twisted. The most important tool for any network application developer. It has a very beautiful api and is used by a lot of famous python developers.

**8.** NumPy. How can we leave this very important library ? It provides some advance math functionalities to python.

**9.** SciPy. When we talk about NumPy then we have to talk about scipy. It is a library of algorithms and mathematical tools for python and has caused many scientists to switch from ruby to python.

**10.** matplotlib. A numerical plotting library. It is very useful for any data scientist or any data analyzer.

**11.** Pygame. Which developer does not like to play games and develop them ? This library will help you achieve your goal of 2d game development.

**12.** Pyglet. A 3d animation and game creation engine. This is the engine in which the famous [python port](https://github.com/fogleman/Minecraft) of minecraft was made

**13.** pyQT. A GUI toolkit for python. It is my second choice after wxpython for developing GUI’s for my python scripts.

**14.** pyGtk. Another python GUI library. It is the same library in which the famous Bittorrent client is created.

**15.** Scapy. A packet sniffer and analyzer for python made in python.

**16.** pywin32. A python library which provides some useful methods and classes for interacting with windows.

**17.** nltk. Natural Language Toolkit – I realize most people won’t be using this one, but it’s generic enough. It is a very useful library if you want to manipulate strings. But it’s capacity is beyond that. Do check it out.

**18.** nose. A testing framework for python. It is used by millions of python developers. It is a must have if you do test driven development.

**19.** SymPy. SymPy can do algebraic evaluation, differentiation, expansion, complex numbers, etc. It is contained in a pure Python distribution.

**20.** IPython. I just can’t stress enough how useful this tool is. It is a python prompt on steroids. It has completion, history, shell capabilities, and a lot more. Make sure that you take a look at it.

**Numpy**

NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers. In NumPy dimensions are called axes. The number of axes is rank.

• Offers Matlab-ish capabilities within Python

• Fast array operations

• 2D arrays, multi-D arrays, linear algebra etc.

**Matplotlib**

• High quality plotting library.

**Python class and objects**

These are the building blocks of OOP class creates a new object. This object can be anything, whether an abstract data concept or a model of a physical object, e.g. a chair. Each class has individual characteristics unique to that class, including variables and methods. Classes are very powerful and currently “the big thing” in most programming languages. Hence, there are several chapters dedicated to OOP later in the book.

The class is the most basic component of object-oriented programming. Previously, you learned how to use functions to make your program do something.

Now will move into the big, scary world of Object-Oriented Programming (OOP). To be honest, it took me several months to get a handle on objects.

When I first learned C and C++, I did great; functions just made sense for me.

Having messed around with BASIC in the early ’90s, I realized functions were just like subroutines so there wasn’t much new to learn.

However, when my C++ course started talking about objects, classes, and all the new features of OOP, my grades definitely suffered.

Once you learn OOP, you’ll realize that it’s actually a pretty powerful tool. Plus many Python libraries and APIs use classes, so you should at least be able to understand what the code is doing.

One thing to note about Python and OOP: it’s not mandatory to use objects in your code in a way that works best; maybe you don’t need to have a full-blown class with initialization code and methods to just return a calculation. With Python, you can get as technical as you want.

As you’ve already seen, Python can do just fine with functions. Unlike languages such as Java, you aren’t tied down to a single way of doing things; you can mix functions and classes as necessary in the same program. This lets you build the code

Objects are an encapsulation of variables and functions into a single entity. Objects get their variables and functions from classes. Classes are essentially a template to create your objects.

Here’s a brief list of Python OOP ideas:

• The class statement creates a class object and gives it a name. This creates a new namespace.

• Assignments within the class create class attributes. These attributes are accessed by qualifying the name using dot syntax: ClassName.Attribute.

• Class attributes export the state of an object and its associated behavior. These attributes are shared by all instances of a class.

• Calling a class (just like a function) creates a new instance of the class.

This is where the multiple copies part comes in.

• Each instance gets ("inherits") the default class attributes and gets its own namespace. This prevents instance objects from overlapping and confusing the program.

• Using the term self identifies a particular instance, allowing for per-instance attributes. This allows items such as variables to be associated with a particular instance.

**Inheritance**

First off, classes allow you to modify a program without really making changes to it.

To elaborate, by subclassing a class, you can change the behavior of the program by simply adding new components to it rather than rewriting the existing components.

As we’ve seen, an instance of a class inherits the attributes of that class.

However, classes can also inherit attributes from other classes. Hence, a subclass inherits from a superclass allowing you to make a generic superclass that is specialized via subclasses.

The subclasses can override the logic in a superclass, allowing you to change the behavior of your classes without changing the superclass at all.

Operator Overloads

Operator overloading simply means that objects that you create from classes can respond to actions (operations) that are already defined within Python, such as addition, slicing, printing, etc.

Even though these actions can be implemented via class methods, using overloading ties the behavior closer to Python’s object model and the object interfaces are more consistent to Python’s built-in objects, hence overloading is easier to learn and use.

User-made classes can override nearly all of Python’s built-in operation methods.

**Exceptions**

I’ve talked about exceptions before but now I will talk about them in depth. Essentially, exceptions are events that modify program’s flow, either intentionally or due to errors.

They are special events that can occur due to an error, e.g. trying to open a file that doesn’t exist, or when the program reaches a marker, such as the completion of a loop.

Exceptions, by definition, don’t occur very often; hence, they are the "exception to the rule" and a special class has been created for them. Exceptions are everywhere in Python.

Virtually every module in the standard Python library uses them, and Python itself will raise them in a lot of different circumstances.

Here are just a few examples:

• Accessing a non−existent dictionary key will raise a KeyError exception.

• Searching a list for a non−existent value will raise a ValueError exception

. • Calling a non−existent method will raise an AttributeError exception.

• Referencing a non−existent variable will raise a NameError exception.

• Mixing datatypes without coercion will raise a TypeError exception.

One use of exceptions is to catch a fault and allow the program to continue working; we have seen this before when we talked about files.

This is the most common way to use exceptions. When programming with the Python command line interpreter, you don’t need to worry about catching exceptions.

Your program is usually short enough to not be hurt too much if an exception occurs.

Plus, having the exception occur at the command line is a quick and easy way to tell if your code logic has a problem.

However, if the same error occurred in your real program, it will fail and stop working. Exceptions can be created manually in the code by raising an exception.

It operates exactly as a system-caused exceptions, except that the programmer is doing it on purpose. This can be for a number of reasons. One of the benefits of using exceptions is that, by their nature, they don’t put any overhead on the code processing.

Because exceptions aren’t supposed to happen very often, they aren’t processed until they occur.

Exceptions can be thought of as a special form of the if/elif statements. You can realistically do the same thing with if blocks as you can with exceptions.

However, as already mentioned, exceptions aren’t processed until they occur; if blocks are processed all the time.

Proper use of exceptions can help the performance of your program.

The more infrequent the error might occur, the better off you are to use exceptions; using if blocks requires Python to always test extra conditions before continuing.

Exceptions also make code management easier: if your programming logic is mixed in with error-handling if statements, it can be difficult to read, modify, and debug your program.

User-Defined Exceptions

I won’t spend too much time talking about this, but Python does allow for a programmer to create his own exceptions.

You probably won’t have to do this very often but it’s nice to have the option when necessary.

However, before making your own exceptions, make sure there isn’t one of the built-in exceptions that will work for you.

They have been "tested by fire" over the years and not only work effectively, they have been optimized for performance and are bug-free.

Making your own exceptions involves object-oriented programming, which will be covered in the next chapter

. To make a custom exception, the programmer determines which base exception to use as the class to inherit from, e.g. making an exception for negative numbers or one for imaginary numbers would probably fall under the Arithmetic Error exception class.

To make a custom exception, simply inherit the base exception and define what it will do.

**Python modules**

Python allows us to store our code in files (also called modules). This is very useful for more serious programming, where we do not want to retype a long function definition from the very beginning just to change one mistake. In doing this, we are essentially defining our own modules, just like the modules defined already in the Python library.

To support this, Python has a way to put definitions in a file and use them in a script or in an interactive instance of the interpreter. Such a file is called a module; definitions from a module can be imported into other modules or into the main module.

**Testing code**

As indicated above, code is usually developed in a file using an editor.

To test the code, import it into a Python session and try to run it.

Usually there is an error, so you go back to the file, make a correction, and test again.

This process is repeated until you are satisfied that the code works. T

he entire process is known as the development cycle.

There are two types of errors that you will encounter. Syntax errors occur when the form of some command is invalid.

This happens when you make typing errors such as misspellings, or call something by the wrong name, and for many other reasons. Python will always give an error message for a syntax error.

## Functions in Python

## It is possible, and very useful, to define our own functions in Python. Generally speaking, if you need to do a calculation only once, then use the interpreter. But when you or others have need to perform a certain type of calculation many times, then define a function.

## You use functions in programming to bundle a set of instructions that you want to use repeatedly or that, because of their complexity, are better self-contained in a sub-program and called when needed. That means that a function is a piece of code written to carry out a specified task.

## To carry out that specific task, the function might or might not need multiple inputs. When the task is carred out, the function can or can not return one or more values.There are three types of functions in python:

## help(),min(),print().

## Python Namespace

Generally speaking, a **namespace** (sometimes also called a context) is a naming system for making names unique to avoid ambiguity. Everybody knows a namespacing system from daily life, i.e. the naming of people in firstname and familiy name (surname).

An example is a network: each network device (workstation, server, printer, ...) needs a unique name and address. Yet another example is the directory structure of file systems.

The same file name can be used in different directories, the files can be uniquely accessed via the pathnames.   
Many programming languages use namespaces or contexts for identifiers. An identifier defined in a namespace is associated with that namespace.

This way, the same identifier can be independently defined in multiple namespaces. (Like the same file names in different directories) Programming languages, which support namespaces, may have different rules that determine to which namespace an identifier belongs.

Namespaces in Python are implemented as Python dictionaries, this means it is a mapping from names (keys) to objects (values). The user doesn't have to know this to write a Python program and when using namespaces.

Some namespaces in Python:

* **global names** of a module
* **local names** in a function or method invocation
* **built-in names**: this namespace contains built-in functions (e.g. abs(), cmp(), ...) and built-in exception names

**Garbage Collection**

Garbage Collector exposes the underlying memory management mechanism of Python, the automatic garbage collector. The module includes functions for controlling how the collector operates and to examine the objects known to the system, either pending collection or stuck in reference cycles and unable to be freed.

**Python XML Parser**

XML is a portable, open source language that allows programmers to develop applications that can be read by other applications, regardless of operating system and/or developmental language.

What is XML? The Extensible Markup Language XML is a markup language much like HTML or SGML.

This is recommended by the World Wide Web Consortium and available as an open standard.

XML is extremely useful for keeping track of small to medium amounts of data without requiring a SQL-based backbone.

XML Parser Architectures and APIs the Python standard library provides a minimal but useful set of interfaces to work with XML.

The two most basic and broadly used APIs to XML data are the SAX and DOM interfaces.

Simple API for XML SAX : Here, you register callbacks for events of interest and then let the parser proceed through the document.

This is useful when your documents are large or you have memory limitations, it parses the file as it reads it from disk and the entire file is never stored in memory.

Document Object Model DOM API : This is a World Wide Web Consortium recommendation wherein the entire file is read into memory and stored in a hierarchical tree − based form to represent all the features of an XML document.

SAX obviously cannot process information as fast as DOM can when working with large files. On the other hand, using DOM exclusively can really kill your resources, especially if used on a lot of small files.

SAX is read-only, while DOM allows changes to the XML file. Since these two different APIs literally complement each other, there is no reason why you cannot use them both for large projects.

**Python Web Frameworks**

A web framework is a code library that makes a developer's life easier when building reliable, scalable and maintainable web applications.

**Why are web frameworks useful?**

Web frameworks encapsulate what developers have learned over the past twenty years while programming sites and applications for the web. Frameworks make it easier to reuse code for common HTTP operations and to structure projects so other developers with knowledge of the framework can quickly build and maintain the application.

Common web framework functionality

Frameworks provide functionality in their code or through extensions to perform common operations required to run web applications. These common operations include:

1. URL routing
2. HTML, XML, JSON, and other output format templating
3. Database manipulation
4. Security against Cross-site request forgery (CSRF) and other attacks
5. Session storage and retrieval

Not all web frameworks include code for all of the above functionality. Frameworks fall on the spectrum from executing a single use case to providing every known web framework feature to every developer. Some frameworks take the "batteries-included" approach where everything possible comes bundled with the framework while others have a minimal core package that is amenable to extensions provided by other packages.

**Comparing web frameworks**

There is also a repository called [compare-python-web-frameworks](https://github.com/mattmakai/compare-python-web-frameworks) where the same web application is being coded with varying Python web frameworks, templating engines and object.

**Web framework resources**

* When you are learning how to use one or more web frameworks it's helpful to have an idea of what the code under the covers is doing.
* Frameworks is a really well done short video that explains how to choose between web frameworks. The author has some particular opinions about what should be in a framework. For the most part I agree although I've found sessions and database ORMs to be a helpful part of a framework when done well.
* what is a web framework? is an in-depth explanation of what web frameworks are and their relation to web servers.
* Django vs Flash vs Pyramid: Choosing a Python web framework contains background information and code comparisons for similar web applications built in these three big Python frameworks.
* This fascinating blog post takes a look at the  code complexity of several Python web frameworks by providing visualizations based on their code bases.
* Python’s web frameworks benchmarks  is a test of the responsiveness of a framework with encoding an object to JSON and returning it as a response as well as retrieving data from the database and rendering it in a template. There were no conclusive results but the output is fun to read about nonetheless.
* What web frameworks do you use and why are they awesome? is a language agnostic Reddit discussion on web frameworks. It's interesting to see what programmers in other languages like and dislike about their suite of web frameworks compared to the main Python frameworks.
* This user-voted question & answer site asked "What are the best general purpose Python web frameworks usable in production?". The votes aren't as important as the list of the many frameworks that are available to Python developers.

## Web frameworks learning checklist

1. Choose a major Python web framework (Django or Flask are recommended) and stick with it. When you're just starting it's best to learn one framework first instead of bouncing around trying to understand every framework.
2. Work through a detailed tutorial found within the resources links on the framework's page.
3. Study open source examples built with your framework of choice so you can take parts of those projects and reuse the code in your application.
4. Build the first simple iteration of your web application then go to the [deployment](https://www.fullstackpython.com/deployment.html)section to make it accessible on the web.

**Python-Data Base Communication**

Connector/Python provides a connect() call used to establish connections to the MySQL server. The following sections describe the permitted arguments for connect() and describe how to use option files that supply additional arguments.

A database is an organized collection of data. The data are typically organized to model aspects of reality in a way that supports processes requiring this information.

The term "database" can both refer to the data themselves or to the database management system. The Database management system is a software application for the interaction between users database itself.

Databases are popular for many applications, especially for use with web applications or customer-oriented programs

Users don't have to be human users. They can be other programs and applications as well. We will learn how Python or better a Python program can interact as a user of an SQLdatabase.   
  
This is an introduction into using SQLite and MySQL from Python.

The Python standard for database interfaces is the Python DB-API, which is used by Python's database interfaces.

The DB-API has been defined as a common interface, which can be used to access relational databases.

In other words, the code in Python for communicating with a database should be the same, regardless of the database and the database module used. Even though we use lots of SQL examples, this is not an introduction into SQL but a tutorial on the Python interface.

SQLite is a simple relational database system, which saves its data in regular data files or even in the internal memory of the computer, i.e. the RAM.

It was developped for embedded applications, like Mozilla-Firefox (Bookmarks), Symbian OS or Android.

SQLITE is "quite" fast, even though it uses a simple file. It can be used for large databases as well.

If you want to use SQLite, you have to import the module sqlite3. To use a database, you have to create first a Connection object.

The connection object will represent the database. The argument of connection - in the following example "companys.db" - functions both as the name of the file, where the data will be stored, and as the name of the database. If a file with this name exists, it will be opened.

It has to be a SQLite database file of course! In the following example, we will open a database called company.

MySQL Connector/Python enables Python programs to access MySQL databases, using an API that is compliant with the Python Database API Specification v2.0 (PEP 249). It is written in pure Python and does not have any dependencies except for the Python Standard Library.

For notes detailing the changes in each release of Connector/Python, see MySQL Connector/Python Release Notes.

MySQL Connector/Python includes support for:

* Almost all features provided by MySQL Server up to and including MySQL Server version 5.7.
* Converting parameter values back and forth between Python and MySQL data types, for example Python datetimeand MySQL DATETIME. You can turn automatic conversion on for convenience, or off for optimal performance
* All MySQL extensions to standard SQL syntax.
* Protocol compression, which enables compressing the data stream between the client and server.
* Connections using TCP/IP sockets and on Unix using Unix sockets.
* Secure TCP/IP connections using SSL.
* Self-contained driver. Connector/Python does not require the MySQL client library or any Python modules outside the standard library

#### 5. SOFTWARE REQUIREMENT SPECIFICATION

#### 5.1 Requirements Specification:

#### Requirement Specification provides a high secure storage to the web server efficiently. Software requirements deal with software and hardware resources that need to be installed on a serve which provides optimal functioning for the application. These software and hardware requirements need to be installed before the packages are installed. These are the most common set of requirements defined by any operation system. These software and hardware requirements provide a compatible support to the operation system in developing an application.

#### 5.1.1 HARDWARE REQUIREMENTS:

#### The hardware requirement specifies each interface of the software elements and the hardware elements of the system. These hardware requirements include configuration characteristics.

#### System : Pentium IV 2.4 GHz.

#### Hard Disk : 100 GB.

#### Monitor : 15 VGA Color.

#### Mouse : Logitech.

#### RAM : 1 GB.

#### 5.1.2 SOFTWARE REQUIREMENTS:

#### The software requirements specify the use of all required software products like data management system. The required software product specifies the numbers and version. Each interface specifies the purpose of the interfacing software as related to this software product.

#### Operating system : Windows XP/7/10

* Coding Language : Html, JavaScript,
* IDE : Anaconda prompt
* Libraries : pandas, Tensorflow,Numpy,time

#### 5.2 FUNCTIONAL REQUIREMENTS:

The functional requirement refers to the system needs in an exceedingly computer code engineering method.

The key goal of determinant “functional requirements” in an exceedingly product style and implementation is to capture the desired behavior of a software package in terms of practicality and also the technology implementation of the business processes.

#### 5.3 NON FUNCTIONAL REQUIREMENTS

All the other requirements which do not form a part of the above specification are categorized as Non-Functional needs. A system perhaps needed to gift the user with a show of the quantity of records during info. If the quantity must be updated in real time, the system architects should make sure that the system is capable of change the displayed record count at intervals associate tolerably short interval of the quantity of records dynamic. Comfortable network information measure may additionally be a non-functional requirement of a system.

The following are the features:

* Accessibility
* Availability
* Backup
* Certification
* Compliance
* Configuration Management
* Documentation
* Disaster Recovery
* Efficiency(resource consumption for given load)
* Interoperability

**5.4 PERFORMANCE** **REQUIREMENTS**

Performance is measured in terms of the output provided by the application. Requirement specification plays an important part in the analysis of a system. Only when the requirement specifications are properly given, it is possible to design a system, which will fit into required environment. It rests largely with the users of the existing system to give the requirement specifications because they are the people who finally use the system. This is because the requirements have to be known during the initial stages so that the system can be designed according to those requirements. It is very difficult to change the system once it has been designed and on the other hand designing a system, which does not cater to the requirements of the user, is of no use.

The requirement specification for any system can be broadly stated as given below:

* The system should be able to interface with the existing system
* The system should be accurate
* The system should be better than the existing system

The existing system is completely dependent on the user to perform all the duties.

**5.5 Feasibility Study:**

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All systems are feasible if they are given unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Technical Feasibility
* Operation Feasibility

Economical Feasibility

**5.5.1 Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?

Are there technical guarantees of accuracy, reliability, ease of access and data security?

**5.5.2 Operational Feasibility**

**User-friendly**

Customer will use the forms for their various transactions i.e. for adding new routes, viewing the routes details. Also the Customer wants the reports to view the various transactions based on the constraints. These forms and reports are generated as user-friendly to the Client.

**Reliability**

The package wills pick-up current transactions on line. Regarding the old transactions, User will enter them in to the system.

**Security**

The web server and database server should be protected from hacking, virus etc

**Portability**

The application will be developed using standard open source software (Except Oracle) like Java, tomcat web server, Internet Explorer Browser etc these software will work both on Windows and Linux o/s. Hence portability problems will not arise.

**Availability**

This software will be available always.

**Maintainability**

The system uses the 2-tier architecture. The 1st tier is the GUI, which is said to be front-end and the 2nd tier is the database, which uses sqllite, which is the back-end.

The front-end can be run on different systems (clients). The database will be running at the server. Users access these forms by using the user-ids and the passwords.

**5.5.3 Economic Feasibility**

The computerized system takes care of the present existing system’s data flow and procedures completely and should generate all the reports of the manual system besides a host of other management reports.

It should be built as a web based application with separate web server and database server. This is required as the activities are spread throughout the organization customer wants a centralized database. Further some of the linked transactions take place in different

## 7. System Design

## 7.1 SYSTEM ARCHITECTURE

## 

## The purpose of the design phase is to arrange an answer of the matter such as by the necessity document. This part is that the opening moves in moving the matter domain to the answer domain. The design phase satisfies the requirements of the system. The design of a system is probably the foremost crucial issue warm heartedness the standard of the software package. It’s a serious impact on the later part, notably testing and maintenance.

## The output of this part is that the style of the document. This document is analogous to a blueprint of answer and is employed later throughout implementation, testing and maintenance. The design activity is commonly divided into 2 separate phases System Design and Detailed Design.

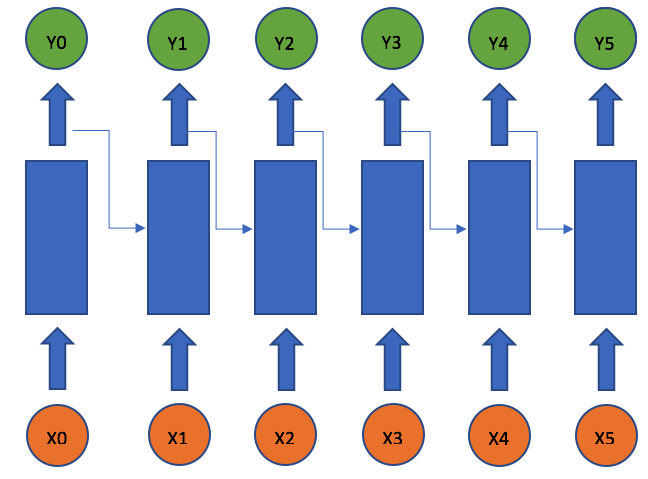
## System Design conjointly referred to as top-ranking style aims to spot the modules that ought to be within the system, the specifications of those modules, and the way them move with one another to supply the specified results.

## At the top of the system style all the main knowledge structures, file formats, output formats, and also the major modules within the system and their specifications square measure set. System design is that the method or art of process the design, components, modules, interfaces, and knowledge for a system to satisfy such as needs. Users will read it because the application of systems theory to development.

## Detailed Design, the inner logic of every of the modules laid out in system design is determined. Throughout this part, the small print of the info of a module square measure sometimes laid out in a high-level style description language that is freelance of the target language within which the software package can eventually be enforced.

## In system design the main target is on distinguishing the modules, whereas throughout careful style the main target is on planning the logic for every of the modules.

## 



## Figure 7.1: Architecture diagram

## 7.2 DATA FLOW DIAGRAMS

## Data Flow Diagram can also be termed as bubble chart. It is a pictorial or graphical form, which can be applied to represent the input data to a system and multiple functions carried out on the data and the generated output by the system.

## A graphical tool accustomed describe and analyze the instant of knowledge through a system manual or automatic together with the method, stores of knowledge, and delays within the system. The transformation of knowledge from input to output, through processes, is also delineate logically and severally of the physical elements related to the system. The DFD is also known as a data flow graph or a bubble chart.The BasicNotation used to create a DFD’s are as follows:

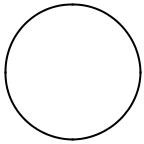
* **Dataflow:**





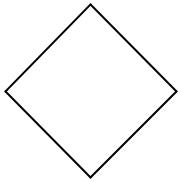
* **Process:**

.



* **Source:**
* **Data Store:**



* **Rhombus**: decision

**7.3 UML DIAGRAMS**

The Unified Modeling Language allows the software engineer to express an analysis model using the modeling notation that is governed by a set of syntactic semantic and pragmatic rules.

A UML system is represented using five different views that describe the system from distinctly different perspective. Each view is defined by a set of diagram, which is as follows.

**User Model View**

This view represents the system from the users perspective. The analysis representation describes a usage scenario from the end-users perspective.

**Structural Model view**

In this model the data and functionality are arrived from inside the system. This model view models the static structures.

**Behavioral Model View**

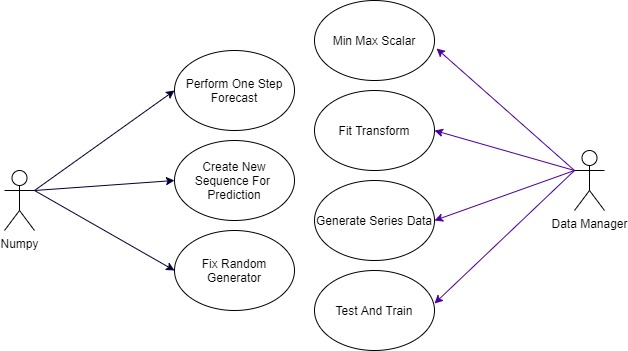
It represents the dynamic of behavioral as parts of the system, depicting the interactions of collection between various structural elements described in the user model and structural model view.

**Implementation Model View**

In this the structural and behavioral as parts of the system are represented as they are to be built.

**5.3.1 USE CASE DIAGRAM**

A use case diagram at its simplest is a representation of a user's interaction with the system and depicting the specifications of a use case. A use case diagram can portray the different types of users of a system and the various ways that they interact with the system. This type of diagram is typically used in conjunction with the textual use case and will often be accompanied by other types of diagrams as well.



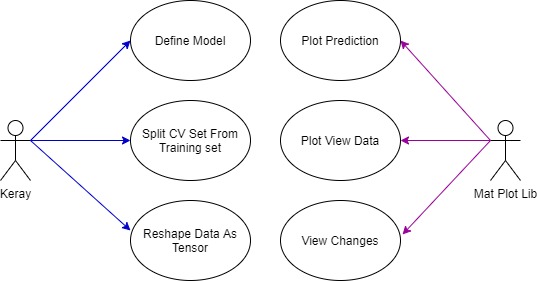


Figure 7.3.1 Use Case Diagram

* + 1. **CLASS DIAGRAM**

The class diagram is the main building block of object oriented modeling. It is used both for general conceptual modeling of the systematic of the application, and for detailed modeling translating the models into programming code. Class diagrams can also be used for data modeling. The classes in a class diagram represent both the main objects, interactions in the application and the classes to be programmed. A class with three sections, in the diagram, classes is represented with boxes which contain three parts:

The upper part holds the name of the class

The middle part contains the attributes of the class

The bottom part gives the methods or operations the class can take or undertake.

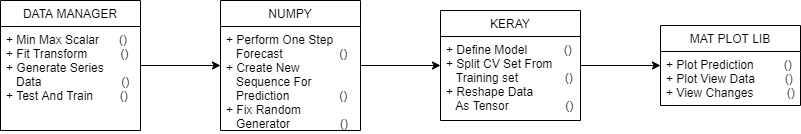
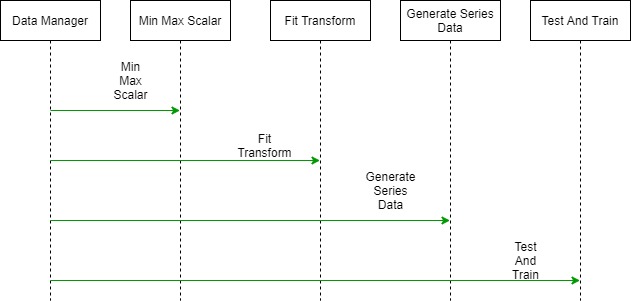
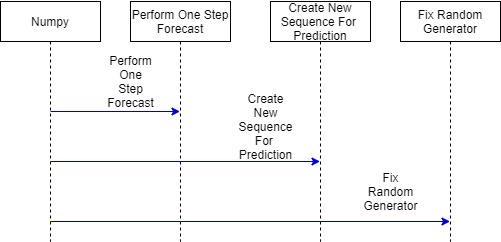


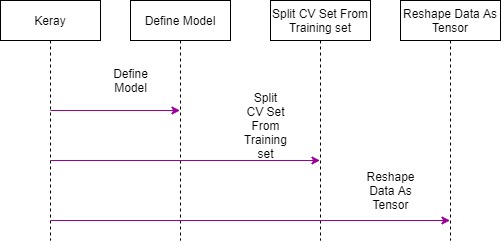
Figure 7.3.2: Class Diagram.

## 5.3.3 SEQUENCEDIAGRAM

A sequence diagram is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. Sequence diagrams are typically associated with use case realizations in the Logical View of the system under development. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.







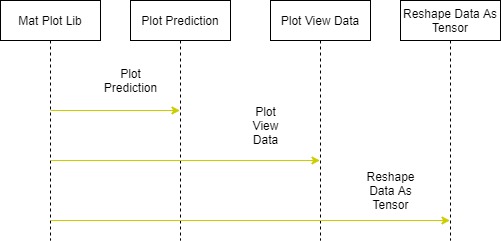


Figure 7.3.3: Sequence diagram

## 5.3.4 ACTIVITY DIAGRAM

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

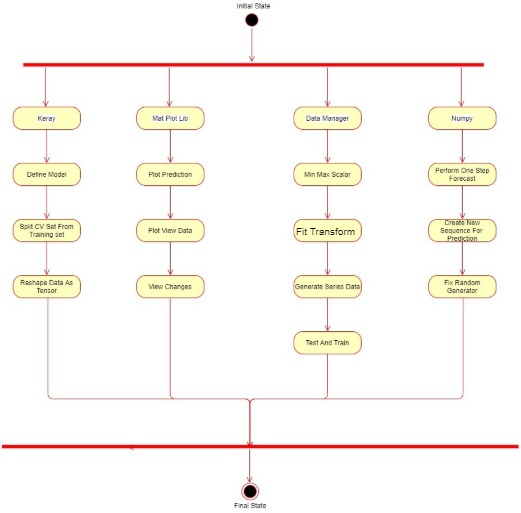
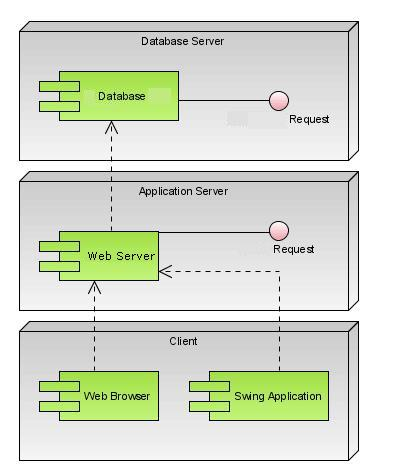


Figure 7.3.4: Activity Diagram

**Component Diagram**

**Deployment Diagram**

****

****

**8. TESTING**

## Testing is the process where the test data is prepared and is used for testing the modules individually and later the validation given for the fields. Then the system testing takes place which makes sure that all components of the system property functions as a unit. The test data should be chosen such that it passed through all possible condition. The following is the description of the testing strategies, which were carried out during the testing period.

## 8.1 SYSTEM TESTING

## Testing has become an integral part of any system or project especially in the field of information technology. The importance of testing is a method of justifying, if one is ready to move further, be it to be check if one is capable to with stand the rigors of a particular situation cannot be underplayed and that is why testing before development is so critical. When the software is developed before it is given to user to user the software must be tested whether it is solving the purpose for which it is developed. This testing involves various types through which one can ensure the software is reliable. The program was tested logically and pattern of execution of the program for a set of data are repeated. Thus the code was exhaustively checked for all possible correct data and the outcomes were also checked.

## 8.2 MODULE TESTING

## To locate errors, each module is tested individually. This enables us to detect error and correct it without affecting any other modules. Whenever the program is not satisfying the required function, it must be corrected to get the required result. Thus all the modules are individually tested from bottom up starting with the smallest and lowest modules and proceeding to the next level. Each module in the system is tested separately. For example the job classification module is tested separately. This module is tested with different job and its approximate execution time and the result of the test is compared with the results that are prepared manually. Each module in the system is tested separately. In this system the resource classification and job scheduling modules are tested separately and their corresponding results are obtained which reduces the process waiting time.

## 8.3 INTEGRATION TESTING

## After the module testing, the integration testing is applied. When linking the modules there may be chance for errors to occur, these errors are corrected by using this testing. In this system all modules are connected and tested. The testing results are very correct. Thus the mapping of jobs with resources is done correctly by the system

## 8.4 ACCEPTANCE TESTING

## When that user fined no major problems with its accuracy, the system passers through a final acceptance test. This test confirms that the system needs the original goals, objectives and requirements established during analysis without actual execution which elimination wastage of time and money acceptance tests on the shoulders of users and management, it is finally acceptable and ready for the operation.

**8.5 TEST CASES:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Test Case Id** | **Test Case Name** | **Test Case Desc.** | **Test Steps** | | | **Test Case Status** | **Test Priority** |
| **Step** | **Expected** | **Actual** |
| 01 | Upload the tasks dataset | Verify either file is loaded or not | If dataset is not uploaded | It cannot display the file loaded message | File is loaded which displays task waiting time | High | High |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |

TABLE 8.5.1 TESTCASES

**9. SCREEN SHOTS**

**10. CONCLUSION**

Deep learning models such as the RNN and LSTM are evidently effective for Bitcoin prediction with the LSTM more capable for recognising longer-term dependencies. However, a high variance task of this nature makes it difficult to transpire this into impressive validation results. As a result it remains a difficult task. There is a fine line between overfitting a model and preventing it from learning sufficiently. Dropout is a valuable feature to assist in improving this. However, despite using Bayesian optimisation to optimize the selection of dropout it still couldn’t guarantee good validation results. Despite the metrics of sensitivity, specificity and precision indicating good performance, the actual performance of the ARIMA forecast based on error was significantly worse than the neural network models. The LSTM outperformed the RNN marginally, but not significantly. However, the LSTM takes considerably longer to train.

**Future Enhancements:**

It is not possible to develop a system that makes all the requirements of the user. User requirements keep changing as the system is being used. Some of the future enhancements that can be done to this system are:

* As the technology emerges, it is possible to upgrade the system and can be adaptable to desired environment.
* Based on the future security issues, security can be improved using emerging technologies like single sign-on.

**11. REFERENCES**

[1] Peter Mel and Tim Grace, "The NIST Definition of Cloud Computing", NIST, 2010.

[2]Achill Buhl, "Rising Security Challenges in Cloud Computing", in Proc. of World Congress on Information and correspondence Technologies ,pp. 217-222, Dec. 2011.

[3]Srinivasarao D et al., "Breaking down the Superlative symmetric Cryptosystem Encryption Algorithm", Journal of Global Research in Computer Science, vol. 7, Jul. 2011

[4] Tingyuan Nye and Tang Zhang "An investigation of DES and Blowfish encryption algorithm‟ , in Proc. IEEE Region 10 Conference, pp. 1-4 ,Jan. 2009.

[5] Jitendra Singh Adam et al.," Modified RSA Public Key Cryptosystem Using Short Range Natural Number Algorithm" , International Journal of Advanced Research in Computer Science and Software Engineering ,vol. 2,Aug. 2012.

[6] Manikandan.G et al., "A changed cryptographic plan improving information", Journal of Theoretical and Applied Information Technology, vol. 35, no.2, Jan. 2012.

[7] Niles Maintain and Subhead Bhingarkar, " The examination and Judgment of Nimbus, Open Nebula and Eucalyptus", International Journal of Computational Biology , vol. 3, issue 1, pp 44-47, 2012.

**BIBLIOGRAPHY**

python Technologies

python Complete Reference

*python* Script Programming by Yehuda Shiran

Mastering *python* Security

*python* Professional by Shadab siddiqui

*python* server pages by Larne Pekowsley

*python* Server pages by Nick Todd

HTML

HTML Black Book by Holzner

sqllite

## 

## SAMPLE CODE

## App.py:

import os

import poloniex

import time

import pandas as pd

import matplotlib.pyplot as plt

import numpy as np

from sklearn import preprocessing

import constants

class DataManager:

def \_\_init\_\_(self):

"""

constructor

Args:

"""

self.dataframe = None

self.time\_train = None

self.time\_test = None

self.train\_original\_df = None

self.test\_original\_df = None

self.train\_diffrenced = None

self.test\_diffrenced = None

self.train\_scaled = None

self.test\_scaled = None

self.scaler = None

self.train\_percent = 0.8

self.diff\_interval = 1

self.column = "close"

def return\_chart\_data(self, pair, period, start\_time, end\_time):

"""

retrieve data from poloniex.

"""

polo = poloniex.Poloniex()

chart\_data = polo.returnChartData(pair,

period=period,

start=time.mktime(time.strptime(start\_time, "%Y-%m-%d %H:%M")),

end=time.mktime(time.strptime(end\_time, "%Y-%m-%d %H:%M")))

df = pd.DataFrame(chart\_data)

df["datetime"] = pd.to\_datetime(df["date"].astype(int) , unit="s")

df = df.set\_index("datetime")

fig, ax1 = plt.subplots(figsize=(20, 10))

ax1.plot(df.index, df[self.column].astype(np.float32), label = "Coin Price", color="deeppink")

ax2 = ax1.twinx()

ax2.plot(df.index, df["quoteVolume"].astype(np.float32), label = "Trading Volume of Coin", color="dodgerblue")

ax1.legend(loc=2, fontsize=14)

ax2.legend(loc=1, fontsize=14)

ax1.tick\_params(labelsize=14)

ax2.tick\_params(labelsize=14)

plt.show()

self.dataframe = df.loc[:, ["close"]].astype(np.float32)

def range\_scale\_data(self, matrix):

"""

scale data to a specified range

Args:

dataframe: input data

Returns:

range scaled dataframe

"""

min\_max\_scaler = preprocessing.MinMaxScaler(feature\_range=(-1, 1))

return min\_max\_scaler.fit\_transform(matrix), min\_max\_scaler

def inverse\_scaled\_data(self, data):

"""

inverse scaled data

"""

return self.scaler.inverse\_transform(data)

def \_\_generate\_series\_data\_for\_supervised\_learning(self, data\_array, datetime\_array):

"""

split "close" column as explanatory\_and\_target\_variables.

"""

t = np.empty((0,1), int)

x = np.empty((0,constants.SEQ\_LENGTH), np.float32)

y = np.empty((0,1), np.float32)

m = len(data\_array)

for n in range(constants.SEQ\_LENGTH, m):

new\_t = np.array([[datetime\_array[n]]])

new\_x = np.array([data\_array[n-constants.SEQ\_LENGTH: n]])

new\_y = np.array([[data\_array[n]]])

t = np.append(t, new\_t, axis=0) # append row

x = np.append(x, new\_x, axis=0)

y = np.append(y, new\_y, axis=0)

self.dataframe = self.dataframe.iloc[constants.SEQ\_LENGTH:, :]

return t, np.concatenate([x, y], axis=1)

def \_\_difference\_data(self, data\_array):

"""

difference the dataframe.

this step trys to remove the trend.

"""

diff = np.empty((0,1), np.float32)

for i in range(self.diff\_interval, len(data\_array)):

value = data\_array[i] - data\_array[i - self.diff\_interval]

diff = np.append(diff, value)

print(self.dataframe)

self.dataframe = self.dataframe.iloc[self.diff\_interval:, :]

print(self.dataframe)

fig, ax1 = plt.subplots(figsize=(20, 10))

ax1.plot(self.dataframe.index, diff, label = "differenced", color="deeppink")

ax1.legend(loc=2, fontsize=14)

ax1.tick\_params(labelsize=14)

plt.show()

return diff

def inverse\_differenced\_data(self, y, previous\_y):

"""

this method only revert data for one step.

"""

return y + previous\_y

def \_\_split\_data(self, data\_array, datetime\_array):

"""

split dataset as training and test data.

"""

m = len(data\_array)

train\_batches = int(m \* self.train\_percent / constants.BATCH\_SIZE)

# 80% training data(cv included), 20% test data

m\_train = train\_batches \* constants.BATCH\_SIZE

time\_train, time\_test = datetime\_array[:m\_train], datetime\_array[m\_train:]

data\_train, data\_test = data\_array[:m\_train], data\_array[m\_train:]

self.train\_original\_df = self.dataframe.iloc[:m\_train, :]

self.test\_original\_df = self.dataframe.iloc[m\_train:, :]

self.time\_train = time\_train

self.time\_test = time\_test

self.train\_diffrenced = data\_train

self.test\_diffrenced = data\_test

def prepare\_data(self):

"""

prepare training, cross validation and test data.

"""

# difference data

data\_array = self.\_\_difference\_data(self.dataframe[self.column].values)

time\_array, data\_array = self.\_\_generate\_series\_data\_for\_supervised\_learning(data\_array, self.dataframe.index)

self.\_\_split\_data(data\_array, time\_array)

data\_train\_scaled, scaler = self.range\_scale\_data(self.train\_diffrenced)

self.scaler = scaler

self.train\_scaled = data\_train\_scaled

self.test\_scaled = scaler.transform(self.test\_diffrenced)

def inverse\_data(self, x, y\_pred, previos\_y):

# inverse range

y\_pred\_unscaled = self.inverse\_scaled\_data(np.concatenate([x, y\_pred], axis=1))[:, -1]

# inverse difference

y\_pred\_indiffereced = self.inverse\_differenced\_data(y\_pred\_unscaled, previos\_y)

return y\_pred\_indiffereced