

DEEP LEARNING MEETS WIRELESS OPTIMIZATION:IDENTIFY CRITICAL LINKS

Submitted in partial fulfillment of the requirements for
the award of
Bachelor of Engineering degree in Computer Science and Engineering

by

KAIGALA SATYA SRINIVAS (Reg. No.37110305)

KAMATAM JANAKI RAMIREDDY (Reg. No. 37110308)



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
SCHOOL OF COMPUTING**

SATHYABAMA

**INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)**

**Accredited with Grade "A" by NAAC
JEPPIAAR NAGAR, RAJIV GANDHI
SALAI, CHENNAI – 600 119**

MARCH - 2021



SATHYABAMA

INSTITUTE OF SCIENCE AND TECHNOLOGY
(DEEMED TO BE UNIVERSITY)

Accredited with "A" grade by NAAC
Jeppiaar Nagar, Rajiv Gandhi Salai, Chennai – 600 119

www.sathyabama.ac.in



DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

BONAFIDE CERTIFICATE

This is to certify that this project report is the bonafide work of **KAIGALA SATYA SRINIVAS (Reg.No.37110305)** and **KAMATAM JANAKI RAMIREDDY (Reg. No. 37110308)** who carried out the project entitled "**DEEP LEARNING MEETS WIRELESS OPTIMIZATION: IDENTIFY CRITICAL LINKS**" under my supervision from **August 2020** to **March 2021**.

Internal Guide

Mrs.S.L.Jany Shabu, M.Tech., (Ph.d).,

Head of the Department

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

I **KAIGALA SATYA SRINIVAS** hereby declare that the Project Report entitled “**DEEP LEARNING MEETS WIRELESS OPTIMIZATION:IDENTIFY CRITICAL LINKS**” is done by me under the guidance of **Mrs.S.L.Jany Shabu,M.Tech.,(Ph.d).**, Department of Computer Science and Engineering at Sathyabama Institute of Science and Technology is submitted in partial fulfillment of the requirements for the award of Bachelor of Engineering degree in Computer Science and Engineering.

DATE:

PLACE: CHENNAI

SIGNATURE OF THE CANDIDATE

ACKNOWLEDGEMENT

I am pleased to acknowledge my sincere thanks to **Board of Management** of **SATHYABAMA** for their kind encouragement in doing this project and for completing it successfully. I am grateful to them.

I convey my thanks to **Dr. T. Sasikala, M.E., Ph.D., Dean**, School of Computing, **Dr. S. Vigneswari, M.E., Ph.D.,** and **Dr. L. Lakshmanan, M.E., Ph.D., Heads of the Department** of Computer Science and Engineering for providing me necessary support and details at the right time during the progressive reviews.

I would like to express my sincere and deep sense of gratitude to my Project Guide **Mrs.S.L.Jany Shabu,M.Tech.,(Ph.d).**, for her valuable guidance, suggestions and constant encouragement paved way for the successful completion of my project work.

I wish to express my thanks to all Teaching and Non-teaching staff members of the **Department of Computer Science and Engineering** who were helpful in many ways for the completion of the project.

ABSTRACT

The superior capability of discovering intricate structure of large data sets, deep learning has been widely applied in various areas including wireless networking. While existing deep learning applications mainly focus on data analysis, the role it can play on fundamental research issues in wireless networks is yet to be explored. With the proliferation of wireless networking infrastructure and mobile applications, wireless network optimization has seen a tremendous increase in problem size and complexity, calling for a paradigm for efficient computation. This paper presents a pioneering study on how to exploit deep learning for significant performance gain in wireless network optimization. Analysis on the flow constrained optimization problems suggests the possibility that a smaller-sized problem can be solved while sharing equally optimal solutions with the original problem, by excluding the potentially unused links from the problem formulation. To this end, we design a deep learning framework to find the latent relationship between flow information and link usage by learning from past computation experience. Numerical results demonstrate that the proposed method is capable of identifying critical links and can reduce computation cost by up to 50% without affecting optimality, thus greatly improve the efficiency of solving network optimization problems.

ABSTRACT

v

CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	1.1 INTROCUCTION	1
	1.2 SCOPE OF THE PROJECT	1
	1.3 DOMAIN OVERVIEW	1
	1.3.1 DEEP LEARNING	2
2.	LITERATURE SURVEY	12
3.	SYSTEM ANALYSIS	14
	3.1 EXISTING SYSTEM	19
	3.2 DRAWBACKS FOR EXISTING SYSTEM	19
	3.3 PROPOSED SYSTEM	20
	3.4 ADVANTAGES OF PROPOSED SYSTEM	20
4.	REQUIRES SPECIFICATION	21
	4.1 INTRODUCTION	21
	4.2 HARDWARE & SOFTWARE SPECIFICATION	21
	4.3 SOFTWARE DESCRIPTION	22
	4.3.1 ABOUT PYTHON	22
	4.3.2 ABOUT IDE	32
5.	SYSTEM DESIGN	37
	5.1 INTRODUCTION	37
	5.2 ARCHITECTURE DIAGRAM	37

6.	SYSTEM IMPLEMENTATION	39
	6.1 LIST OF MODULES	39
	6.1.1 MODULE 1: EXPLORATORY DATA EVALUATION	39
	6.1.2 MODULE 2: PRE- PROCESSING	39
	6.1.3 MODULE 3: FEATURE ENGINEERING	40
	6.1.4 MODULE 4: PREDICTION	41
7.	SOFTWARE TESTING	42
	7.1 GENERAL	47
	7.2 TEST DRIVEN DEVELOPMENT	48
	7.3 UNIT TESTING	49
	7.4 BLACKBOX TESTING	50
	7.5 INTEGRATION TESTING	50
	7.6 SYSTEM TESTING	52
	7.7 SANITY TESTING	53
	7.8 REGRESSION TESTING	55
8.	CONCLUSION AND FUTURE ENHANCEMENT	59
	REFERENCES	60
	APPENDIX	
	A. PLAGIARISM REPORT	62
	B. JOURNAL PAPER	74
	C. SOURCE CODE	79

FIGURE No.	FIGURE NAME	PAGE No.
1.3.1.1	CLASSIFICATION	8
1.3.1.2	DEEP	9
	LEARNING	
3.1	WATERFALL MODEL	15
3.2	RAD MODEL	16
4.3.1	PYTHON	22
4.3.1-2	PYTHON	22
	IMPLEMENTATING	
	MACHINE LEARNING	
5.1.1	ARCHITECTURE	38
	DIAGRAM	
7.1	IMPLEMENTATION	45
7.2	BENEFITS OF TESTING	47
7.3	INTEGRATION	51
	TESTING	
7.4	SYSTEM TESTING	52
7.5	PERFORMANCE	57
	TESTING	

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

Aiming at these two problems, this paper first models a nonlinear high-speed link based on Simulink, which can flexibly adjust the nonlinear behaviours and analyse the impact of nonlinear behaviours on the performance of the high-speed links. Then a metric is presented based on the principal component analysis (PCA) to quantify the level of the nonlinearity of the high-speed links. Simulation results show that the metric can effectively capture the nonlinear severity of the links.

1.2 SCOPE OF OBJECT

The main contributions of this project therefore are:

- Data Analysis
- Dataset Preprocessing
- Training the Model
- Testing of Dataset

1.3 DOMAIN OVERVIEW

Deep learning has evolved hand-in-hand with the digital era, which has brought about an explosion of data in all forms and from every region of the world. This data, known simply as big data, is drawn from sources like social media, internet search engines, e-commerce platforms, and online cinemas, among others. This enormous amount of data is readily accessible and can be shared through fintech applications like cloud computing.

Deep learning is a machine learning technique that teaches computers to do what comes naturally to humans: learn by example. Deep learning is a key technology behind driverless cars, enabling them to recognize a stop sign, or to distinguish a pedestrian from a lamppost. It is the key to voice control in consumer devices like phones, tablets, TVs, and hands-free speakers. Deep

learning is getting lots of attention lately and for good reason. It's achieving results that were not possible before.

In deep learning, a computer model learns to perform classification tasks directly from images, text, or sound. Deep learning models can achieve state-of-the-art accuracy, sometimes exceeding

human-level performance. Models are trained by using a large set of labeled data and neural network architectures that contain many layers.

However, the data, which normally is unstructured, is so vast that it could take decades for humans to comprehend it and extract relevant information. Companies realize the incredible potential that can result from unraveling this wealth of information and are increasingly adapting to AI systems for automated support.

1.3.1 DEEP

LEARNING About

Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots.

As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. Series: a pandas Series is a one dimensional data structure (“a one dimensional ndarray”)

that can store values — and for every value it holds a unique index, too.

2. DataFrame: a pandas DataFrame is a two (or more) dimensional data structure – basically a

table with rows and columns. The columns have names and the rows have indexes.

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame

- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.

- a NumPy ndarray, which can be a record or structured

- a two-dimensional ndarray

- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient

- sets in and reshaping of data There are tools available for loading data of different file formats into in-memory data objects.

- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.

- It is very simple to perform pivoting Pandas.

- It also provides indexing, label-based slicing, and sub-setting of large data sets.

We can easily insert and delete columns from a data structure.

Data aggregation and transformations can be done using group by.

High-performance merging and joining of data can be done using Pandas.

It also provides time series functionality.

Inserting and deleting columns in data structures.

Merging and joining data sets.

Reshaping and pivoting data sets.

Aligning data and dealing with missing data.

Manipulating data using integrated indexing for DataFrame objects.

Performing split-apply-combine on data sets using the group by engine.

Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.

Subsetting, fancy indexing, and label-based slicing data sets that are large in size

Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.

Reading from files with CSV, XLSX, TXT, among other formats.

Arranging data in an order ascending or descending.

Filtering data around a condition.

Analyzing time series.

Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays. An array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

The data pointer indicates the memory address of the first byte in the array,

The data type or dtype pointer describes the kind of elements that are contained within the array,

The shape indicates the shape of the array, and

The strides are the number of bytes that should be skipped in memory to go the

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of

multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional

array is also known as a matrix, and is something you should be familiar with. In fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here

- 1) NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions
- 2) Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Mathplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter

was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom. One of the free and open-source Python library which is basically used for technical and scientific computing

is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way.

Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

Deep Learning

Deep learning is a computer software that mimics the network of neurons in a brain. It is a subset of machine learning and is called deep learning because it makes use of deep neural networks.

Deep learning algorithms are constructed with connected layers.

- The first layer is called the Input Layer
- The last layer is called the Output Layer
- All layers in between are called Hidden Layers. The word deep means the network join

neurons in more than two layers.

A deep neural network provides state-of-the-art accuracy in many tasks, from object detection to speech recognition. They can learn automatically, without predefined knowledge explicitly coded by the programmers.

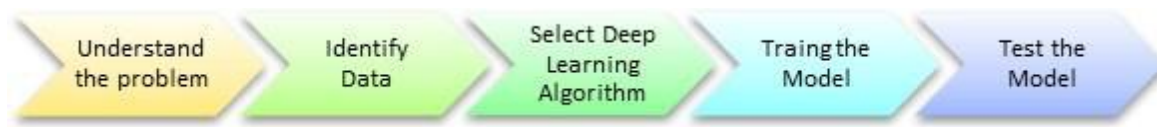


FIG 1.3.1 1 CLASSIFICATION

To grasp the idea of deep learning, imagine a family, with an infant and parents. The toddler points objects with his little finger and always says the word 'cat.' As its parents are concerned about his education, they keep telling him 'Yes, that is a cat' or 'No, that is not a cat.' The infant persists in pointing objects but becomes more accurate with 'cats.' The little kid, deep down, does not know why he can say it is a cat or not. He has just learned how to hierarchies complex features coming up with a cat by looking at the pet overall and continue to focus on details such as the tails or the nose before to make up his mind.

A neural network works quite the same. Each layer represents a deeper level of knowledge, i.e., the hierarchy of knowledge. A neural network with four layers will learn more complex feature than with that with two layers.

The learning occurs in two phases.

1. The first phase consists of applying a nonlinear transformation of the input and create a statistical model as output.
2. The second phase aims at improving the model with a mathematical method known as derivative.

The neural network repeats these two phases hundreds to thousands of time until it has reached a tolerable level of accuracy. The repeat of this two-phase is called an iteration

Classification of Neural Networks

1. Shallow neural network: The Shallow neural network has only one hidden layer between

the input and output.

2. Deep neural network: Deep neural networks have more than one layer. For instance, Google LeNet model for image recognition counts 22 layers.

The computational models in Deep Learning are loosely inspired by the human brain. The multiple layers of training are called Artificial Neural Networks (ANN).

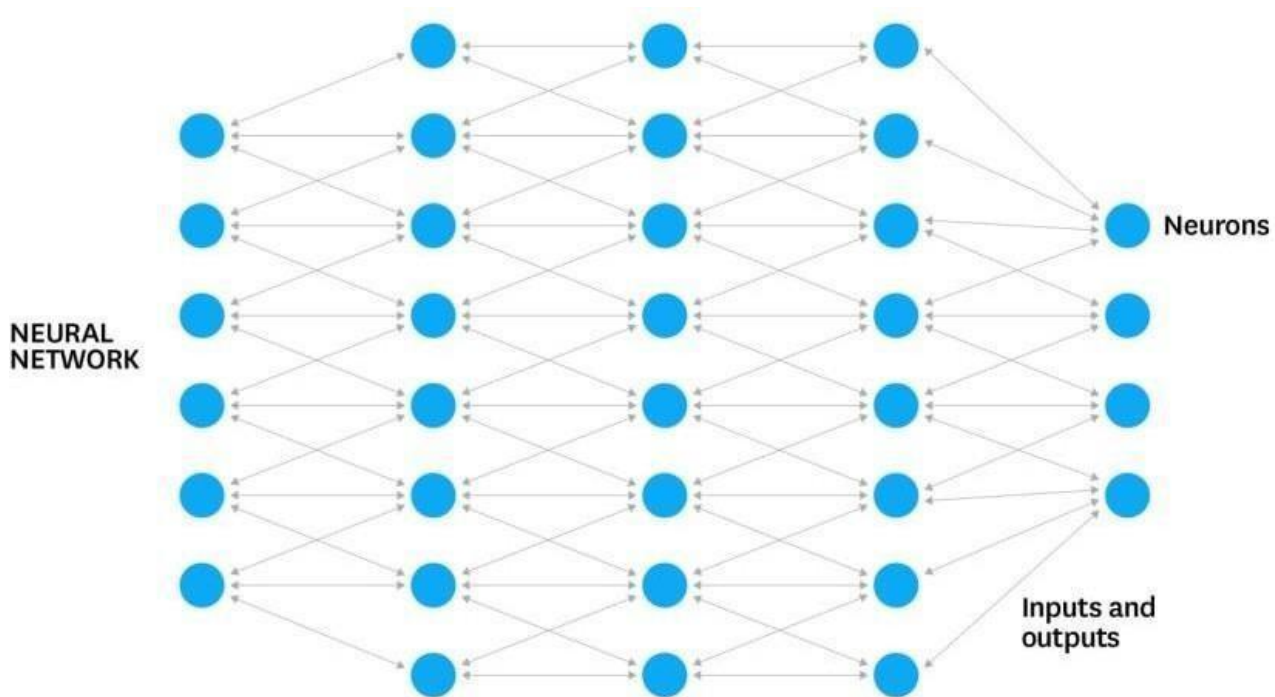


FIG 1.3.1 2 DEEP LEARNING

Neuron

Artificial Neural Networks contain layers of neurons. A neuron is a computational unit that calculates a piece of information based on weighted input parameters. Inputs accepted by the neuron are separately weighted.

Inputs are summed and passed through a non-linear function to produce output. Each layer of neurons detects some additional information, such as edges of things in a picture or tumors in a human body. Multiple layers of neurons can be used to detect additional information about input parameters.

Nodes

Artificial Neural Network is an interconnected group of nodes akin to the vast network of layers of neurons in a brain. Each circular node represents an artificial neuron and an arrow represents a connection from the output of one neuron to the input of another.

Inputs

Inputs are passed into the first layer. Individual neurons receive the inputs, with each of them receiving a specific value. After this, an output is produced based on these values.

Outputs

The outputs from the first layer are then passed into the second layer to be processed. This continues until the final output is produced. The assumption is that the correct output is predefined. Each time data is passed through the network, the end result is compared with the correct one, and tweaks are made to their values until the network creates the correct final output each time.

Some of the commonly used neural networks are as follows:

1. Artificial Neural Network (ANN)
2. Convolutional Neural Network (CNN)
3. Recurrent Neural Network (RNN)
4. Deep Neural Network (DNN)
5. Deep Belief Network (DBN)

Artificial neural networks are one of the main tools used in machine learning. As the “neural” part of their name suggests, they are brain-inspired systems which are intended to replicate the way that we humans learn. Neural networks consist of input and output layers, as well as (in most cases) a hidden layer consisting of units that transform the input into something that the output layer can use. They are excellent tools for finding patterns which are far too complex or numerous for a human programmer to extract and teach the machine to recognize.

Generally, the working of a human brain by making the right connections is the idea behind

ANNs. That was limited to use of silicon and wires as living neurons and dendrites.

Here, neurons, part of human brain. That was composed of 86 billion nerve cells. Also, connected to other thousands of cells by Axons. Although, there are various inputs from sensory organs. That was accepted by dendrites. As a result, it creates electric impulses. That is used to travel through

the artificial neural network. Thus, to handle the different issues, neuron send a message to another neuron.

As a result, we can say that ANNs are composed of multiple nodes. That imitate biological neurons of the human brain. Although, we connect these neurons by links. Also, they interact with each other. Although, nodes are used to take input data.

CHAPTER 2

LITERATURE SURVEY

Modeling for Nonlinear High-speed Links Based on Deep Learning Method

With the nonlinearity degree increasing in high-speed links, the traditional fast time-domain system simulation techniques based on linear time-invariant hypothesis cannot accurately predict the system response. In this paper, a modeling method based on deep learning method is proposed to develop a time-domain model to accurately deal with the nonlinear factor in the high-speed links. Compared with the SBR-based fast timedomain simulation method, the simulated results show that the proposed modeling method based on deep learning can facilitate accurate and fast transient simulation.

Impact of Via Stub Position on High Speed Serial Links

While designing any interface, the impact of channel components like vias, trace, materials, stack- up, connectors, cables, transmitter/receiver packages, and other parasitic effects need to be considered. This paper discusses the impact of via and more importantly the impact of via stub position and length on high speed serial links. Via stub resonance impacts high speed signals but in this paper it is shown that their position also has an impact. Depending on where the via stubs are located on a high speed channel, sometimes their effect could be insignificant (no eye degradation) and sometimes their impact could be significant (major eye degradation).

High-Speed Link Analysis with System Identification Approach

An alternative high-speed link modeling and analysis methodology is suggested based on the system identification approach, instead of using industry standard models. This approach is especially beneficial for the system design/analysis in that the steps are simple and the methodology is well established in other discipline. The modeling of link components like Tx/Redriver/Rx is performed with the identification techniques only using the input/output waveforms. The link analysis is performed with the pre-computed collection of the models, thus significantly reducing the simulation time. The optimal settings of Tx/Rx and the link performance are easily identified and evaluated.

BER Analysis of High Speed Links with Nonlinearity

Quest for higher data rates has led to more complex high speed serial links (SerDes) whose design is more challenging, time consuming, and expensive. As such the ability to accurately predict the performance of the link at the early stages of the design is essential. Bit Error Rate

(BER) is the ultimate metric of performance. Due to the very low BER requirement (10^{-12} or lower), it is not

practical to predict the performance solely through simulation as it requires unacceptably long simulations time. Resorting to accurate analytical methods to augment the simulation is the only viable approach. Analytical methods use knowledge of channel models to calculate probability density function (PDF) of signal (Inter Symbol Interference (ISI), crosstalk, jitter etc.) at the decision point, and use tail probability to predict BER. Such methods based on Linear Time Invariant (LTI) system analysis have been studied extensively in the past. However, it is known that a typical high speed system suffers from significant nonlinearity in the signal path. We present a method to modify the PDF to account for nonlinearity in the path. Once the PDF is correctly modified, similar tail probability methods can be used to determine BER. Static, memory-less nonlinearities are modelled by polynomials in this paper. The methodology can also be adapted to deal with the time varying and frequency dependent nonlinearities.

CHAPTER 3

SYSTEM ANALYSIS

One of the primary choices we face for our project implementations is “Which development methodology should we use?” There are various software development methodologies available in the ICT industry. They are

- Crystal Methods
- Dynamic Systems Development Model (DSDM)
- Extreme Programming (XP)
- Feature Driven Development (FDD)
- Joint Application Development (JAD)
- Lean Development (LD)
- Rapid Application Development (RAD)
- Rational Unified Process (RUP)
- Scrum
- Spiral
- Systems Development Life Cycle (SDLC)
- Waterfall (a.k.a. Traditional)

Among all these methodologies, we have decided to compare the two most popular and widely used methodologies:

- Waterfall Model
- RAD Model

4.2 Waterfall Model

It is called as traditional approach. Waterfall is a linear method (sequential model) for any software application development. Here application development is segregated into a sequence of pre-defined phases.

Waterfall Model (Taken from Google.com)

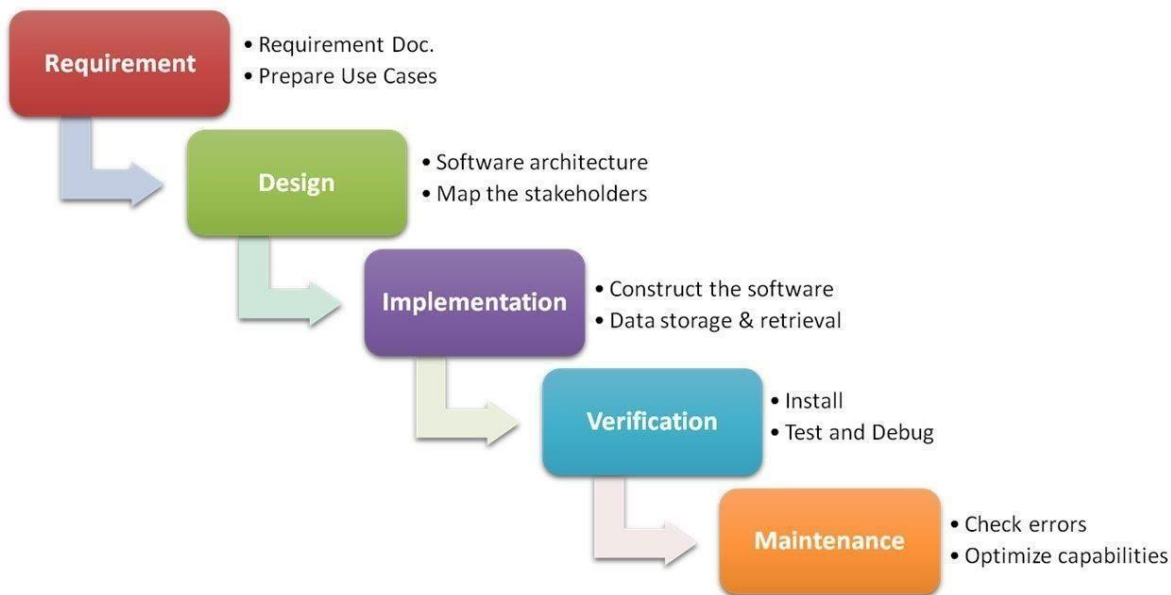


FIG 3- 1 WATERFALL MODEL

(Reference:<https://www.lucidchart.com/blog/waterfall-project-management-methodology>)

1. Requirement gathering and documentation

In this stage, you should gather comprehensive data approximately what this challenge requires. You may gather this data in a diffusion of ways, from interviews to questionnaires to interactive brainstorming. By means of the stop of this section, the task requirements have to be clean, and also you have to have a necessities file that has been allotted on your team.

2. System design

Using the well-known requirements, your team designs the solutions. During this phase, no development will be happening. But the project team starts specification such as programming language or hardware requirements.

3. Implementation

During this phase software development coding will be happening. Web application programmers take data from the previous stage and create a functional product. Web application programmers write source code in small pieces, which are integrated at the end of this phase or the beginning of the next.

4. Testing

Once all coding is done, testing of the product can begin. Testers methodically find and report any problems. If serious issues arise, your project may need to return to phase one for reevaluation.

5. Delivery/deployment

In this phase, the solution is complete, and your project team submits the deliverables to be deployed or released.

6. Maintenance

The final solution has been implemented to the client and is being used. As troubles arises, the project team might also want to create patches and updates might also to deal with them. Again, huge troubles may also necessitate a return to segment one.

4.3 RAD Model

Agile is an iterative, team-based approach to development. This method highlights the speedy delivery of an software application in comprehensive functional components.

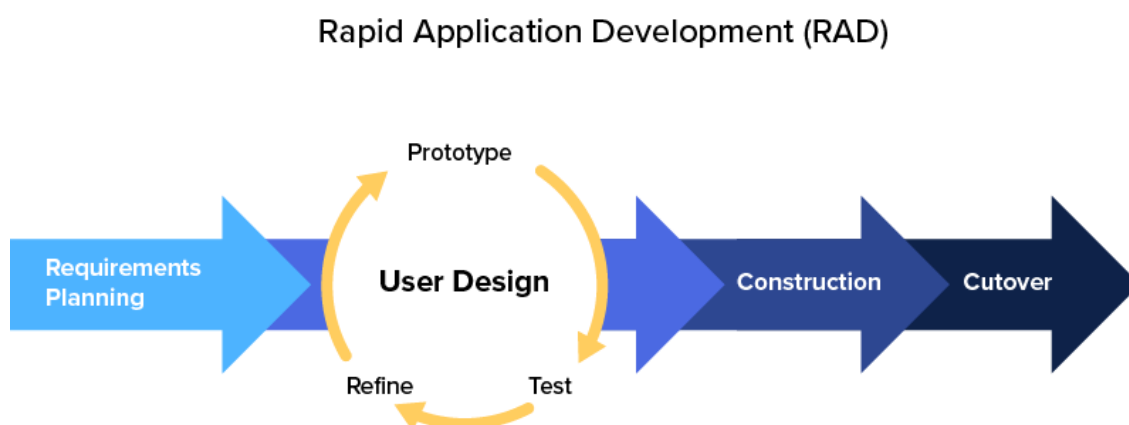


FIG 3- 2 RAD MODEL

RAD Model (Taken from Google.com)

References (<https://www.lucidchart.com/blog/rapid-application-development-methodology>)

Phase 1: Requirements planning

This phase is equivalent to a project scoping meeting. Although the planning phase is condensed compared to other project management methodologies, this is a critical step for the ultimate success of the project.

During this stage, web application programmers, clients (software users), and team members communicate to determine the goals and expectations for the project as well as current and potential issues that would need to be addressed during the build.

A basic breakdown of this stage involves:

Researching the current problem

Defining the requirements for the
project

Finalizing the requirements with each stakeholder's approval

It is important that everyone has the opportunity to evaluate the goals and expectations for the project and weigh in. By getting approval from each key stakeholder and web application programmer, teams can avoid miscommunications and costly change orders down the road.

Phase 2: User design

Once the project is scoped out, it's time to jump right into development, building out the user design through various prototype iterations.

This is the meat and potatoes of the RAD methodology—and what sets it apart from other project management strategies. During this phase, clients work hand in hand with web application programmers to ensure their needs are being met at every step in the design process. It's almost like customizable software development where the users can test each prototype of the product, at each stage, to ensure it meets their expectations.

All the bugs and kinks are worked out in an iterative process. The web application programmer designs a prototype, the client (user) tests it, and then they come together to communicate on

what worked and what didn't.

This method gives web application programmers the opportunity to tweak the model as they go until

they reach a satisfactory design.

Both the software web application programmers and the clients learn from the experience to make sure there is no potential for something to slip through the cracks.

Phase 3: Rapid construction

Phase 3 takes the prototypes and beta systems from the design phase and converts them into the working model.

Because the majority of the problems and changes were addressed during the thorough iterative design phase, web application programmers can construct the final working model more quickly than they could by following a traditional project management approach.

The phase breaks down into several smaller steps:

Preparation for rapid construction

Program and application

development Coding

Unit, integration, and system testing

The software development team of programmers, coders, testers, and web application programmers work together during this stage to make sure everything is working smoothly and that the end result satisfies the client's expectations and objectives.

This third phase is important because the client still gets to give input throughout the process

They can suggest alterations, changes, or even new ideas that can solve problems as they arise.

Phase 4: Cutover

This is the implementation phase where the finished product goes to launch. It includes data conversion, testing, and changeover to the new system, as well as user training.

All final changes are made while the coders and clients continue to look for bugs in the system.

Benefits of RAD methodology

RAD is one of the most successful software development programs available today, with numerous benefits for both software development teams as well as their clients.

Here are just a few advantages:

RAD lets you break the project down into smaller, more manageable tasks.

The task-oriented structure allows project managers to optimize their team's efficiency by assigning tasks according to members' specialties and experience.

Clients get a working product delivered in a shorter time frame.

Regular communication and constant feedback between team members and stakeholders increases the efficiency of the design and build process.

3.1 Existing system

The existing system presents a pioneering study on how to exploit deep learning for significant performance gain in wireless network optimization. Analysis on the flow constrained optimization problems suggests the possibility that a smaller-sized problem can be solved while sharing equally optimal solutions with the original problem, by excluding the potentially unused links from the problem formulation. To this end, we design a deep learning framework to find the latent relationship between flow information and link usage by learning from past computation experience.

For a fixed network topology (physical locations of nodes), the main factor that determines whether a link will be scheduled in the optimal solution is the flow information, which includes the locations of source and destination nodes and flow demand for each commodity. Then the link usage prediction problem is to perform link evaluation based on the flow information. Equivalently, the problem is to find the relationships between the input flow information and the output link evaluations. Generally, flow information affects link usefulness in an indirect way and there is no explicit relationship between them. One approach is to leverage the knowledge obtained from solutions of sample computations, based on which we can predict the usage of links for a new problem, i.e., new input flow information

To give an intuitive example, given node locations and a flow demand with both source and destination nodes located in the upper part of the network, then links in the upper half of the network are more likely to be used than the lower part. The information "upper half of network should be given more consideration" can be derived through problem inputs, and such intermediate information, or features are what we hope to extract. In engineering practice such heuristic approaches are widely used for manual optimization, however with learning techniques it is possible to discover them in a systematic manner.

3.2 Drawbacks of Existing System

Increases network cost

less effectiveness

Predicted future values are much lower than the actual values

Cannot learn the change trend of the traffic

Cannot handle the impact of the sudden jitter in the traffic

3.3 Proposed System

The proposed system main content includes: the processing of raw data, the segmenting of training data, the training of model, the prediction of future traffic. First, a normalization process needs to be performed to convert network traffic into a series of values acceptable for KNN network.

At the beginning of each period, it is necessary to add the latest data to continue training the KNN network. In this way, with the arrival of new data, a series of parameters of the KNN network can be updated to meet the real-time requirements of the prediction. The updating method is to remove the oldest data and add the latest data, both of which have a length of 7 to ensure that the length of traffic.

Initial state, comprehended as a series of parameters related to memory characteristics, is generally filled with zero to initialize the KNN Cell at the first of time steps when training the KNN network. With the updating of the traffic, the initial state of KNN Cell at the beginning of a new round of training is taken the same value as the final state of KNN Cell at the end of the previous round of training.

Excessive training may lead to the occurrence of overfitting. This paper suggests that the times of training in each round should be appropriately reduced without affecting the accuracy of prediction model. It is not difficult to find that the training of the model uses multiple adjacent periods of traffic as the contrast. Theoretically, the modified model can not only learn the change trend of historical traffic, but also avoid the impact of sudden jitter to some extent, so as to ensure the stability of the prediction model.

3.4 Advantages of Proposed System

Reduces complexity of solving network optimization problems

- ✓ Simplifies the service deployment
- ✓ Enhanced the robustness of the model
- ✓ Ensure the stability of the prediction
- ✓ Prediction at different time scales

CHAPTER 4

REQUIREMENT SPECIFICATION

4.1 Introduction

Clearly defined requirements are essential signs on the road that leads to a successful project. They establish a formal agreement between a client and a provider that they are both working to reach the same goal. High-quality, detailed requirements also help mitigate financial risks and keep the project on a schedule. According to the Business Analysis Body of Knowledge definition, requirements are a usable representation of a need.

Creating requirements is a complex task as it includes a set of processes such as elicitation, analysis, specification, validation, and management. In this article, we'll discuss the main types of requirements for software products and provide a number of recommendations for their use.

4.2. Hardware & Software Specification

4.2.1 Hardware Specification

Hardware Requirement

Hardware Minimum Requirement

Disk Space 32 GB or more, 10 GB or more for Foundation

Edition Processor 1.4 GHz 64 bit

Memory 512 MB

Display (800 × 600) Capable video adapter and monitor

4.2.2 Software Specification

Backend Technologies

- ✓ Python
- ✓ Numpy
- ✓ Sci-learn
- ✓ Eclipse IDE

Frontend Technologies

- ✓ Web Technologies
- ✓ Bootstrap

4.3 Software Description

4.3.1 About Python

Python is a free, open-source programming language. Therefore, all you have to do is install Python once, and you can start working with it. Not to mention that you can contribute your own code to the community. Python is also a cross-platform compatible language. So, what does this mean? Well, you can install and run Python on several operating systems. Whether you have a Windows, Mac or Linux, you can rest assure that Python will work on all these operating systems

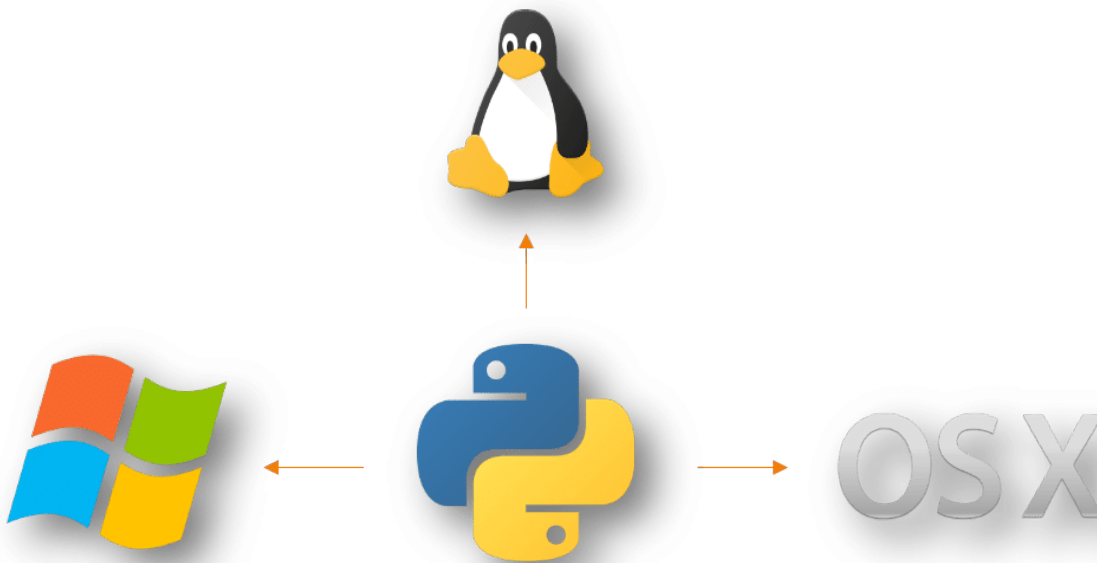


FIG 4.3.1 – 1 PYTHON

Python is also a great visualization tool. It provides libraries such as Matplotlib, seaborn and bokeh to create stunning visualizations.

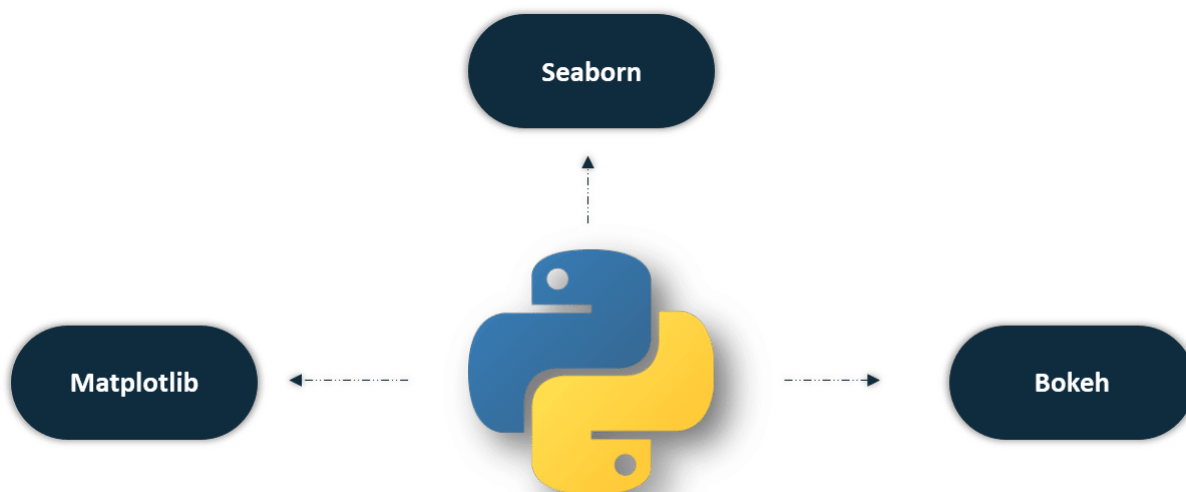


FIG 4.3.1 – 2 PYTHON IMPLEMENTING MACHINE LEARNING

In addition, Python is the most popular language for machine learning and deep learning. As a matter of fact, today, all top organizations are investing in Python to implement machine learning in the back-end.

Python Line Structure

Python coding style comprises physical lines as well as logical lines or statements. A physical line in a Python program is a sequence of characters, and the end of the line terminates the line sequence as opposed to some other languages, such as C and C++ where a semi-colon is used to mark the end of the statement. A logical line, on the other hand, is composed of one or more physical lines. The use of a semi-colon is not prohibited in Python, although it's not mandatory. The NEWLINE token denotes the end of the logical line. A logical line that only contains spaces, comments, or tabs are called blank lines and they are ignored by the interpreter.

As we saw that in Python, a new line simply means that a new statement has started. Although, Python does provide a way to split a statement into a multiline statement or to join multiple statements into one logical line. This can be helpful to increase the readability of the statement.

Following are the two ways to split a line into two or more lines:

Explicit Line Joining

In explicit line joining, we use a backward slash to split a statement into a multiline statement. Implicit Line Joining

Statements that reside inside [], {}, or () parentheses can be broken down into two or more physical lines without using a back slash.

Multiple Statements on a Single Line

In Python, it is possible to club multiple statements in the same line using a semi-colon; however, most programmers do not consider this to be a good practice as it reduces the readability of the code.

Whitespaces and Indentation

Unlike most of the programming languages, Python uses indentation to mark a block of code.

According to Python coding style guideline or PEP8, we should keep an indent size of four.

Most of the programming languages provide indentation for better code formatting and don't enforce to have it. But in Python it is mandatory. This is why indentation is so crucial in Python.

Comments in any programming language are used to increase the readability of the code. Similarly, in Python, when the program starts getting complicated, one of the best ways to maintain the readability of the code is to use Python comments. It is considered a good practice to include documentations and notes in the python syntax since it makes the code way more readable and understandable to other programmers as well, which comes in handy when multiple programmers are simultaneously working on the same project.

The code can only explain how it does something and not why it does that, but Python comments can do that. With Python comments, we can make documentations for various explanations in our code itself. Comments are nothing but tagged lines of codes which increase the readability of a code and make it self-explanatory. There are different ways of creating comments depending on the type of comment we want to include in our code.

Following are different kinds of comments that can be included in our Python program:

1. Single Line Comments
2. Docstring Comments
3. Multiline Comments

Single line Python comments are marked with # character. These comments end at the end of the physical line, which means that all characters starting after the # character (and lasts till the end of the line) are part of the comment.

Python has the documentation strings (or docstrings) feature which is usually the first statement included in functions and modules. Rather than being ignored by the Python Interpreter like regular comments, docstrings can actually be accessed at the run time using the dot operator.

It gives programmers an easy way of adding quick notes with every Python module, function, class, and method. To use this feature, we use triple quotes in the beginning of the

documentation string or comment and the closing triple quotes at the end of the documentation comment. Docstrings can be one-liners as well as multi-liners.

Unlike some programming languages that support multiline comments, such as C, Java, and more, there is no specific feature for multiline comments in Python. But that does not mean that it is totally impossible to make multiline comments in Python. There are two ways we can include comments that can span across multiple lines in our Python code.

Python Block Comments: We can use several single line comments for a whole block. This type of comment is usually created to explain the block of code that follows the Block comment. Python Block comment is the only way of writing a real comment that can span across multiple lines. It is supported and preferred by Python's PEP8 style guide since Block comments are ignored by Python interpreter or parser. However, nothing is stopping programmers from using the second 'non-real' way of writing multiline comments in Python which is explained below.

Using Docstrings: Docstrings are largely used as multiline comments in Python by many programmers since it is the closest thing to having a multiline comment feature in Python. While it is not wrong to use docstrings when we need to make multiline comments, it is important to keep in mind that there is a significant difference between docstrings and comments. Comments in Python are totally ignored by the Python Interpreter, while docstrings, when used inside the Python function, can be accessed at the run time.

Python Data Types

One of the most crucial part of learning any programming language is to understand how data is stored and manipulated in that language. Users are often inclined toward Python because of its ease of use and the number of versatile features it provides. One of those features is dynamic typing.

In Python, unlike statically typed languages like C or Java, there is no need to specifically declare the data type of the variable. In dynamically typed languages such as Python, the interpreter itself predicts the data type of the Python Variable based on the type of value assigned to that variable.

Advantages of Python

Universal Language Construct

Support both High Level and Low Level Programming

Language Interoperability

Fastest Development life cycle therefore more productive coding environment

Less memory used because a single container hold multiple data types and each type doesn't require its own function

Learning Ease and open source development

Speed and user-friendly data structure

Extensive and extensible libraries.

Simple & support IoT

And many more

. Based on the 3 languages that are described above, we decided to use Python as programming language for developing this e-Voting web based application. The important motives are

Easy to learn, even non experienced programmers can use it. Ex: spacing and tabbing instead of extra syntax

Interactive mode

Large and comprehensive standard libraries

Python programs resemble to that of pseudo-code. This makes it a basis and a must have for beginner programmers due to its extreme ease a difficulty when compared to C++ Java, Perl, and so forth

About Pandas

Pandas is a popular Python package for data science, and with good reason: it offers powerful, expressive and flexible data structures that make data manipulation and analysis easy, among many other things. The DataFrame is one of these structures.

Pandas is a high-level data manipulation tool developed by Wes McKinney. It is built on the

Numpy package and its key data structure is called the DataFrame. DataFrames allow you to store and manipulate tabular data in rows of observations and columns of variables.

Pandas is built on top of the NumPy package, meaning a lot of the structure of NumPy is used or replicated in Pandas. Data in pandas is often used to feed statistical analysis in SciPy, plotting functions from Matplotlib, and machine learning algorithms in Scikit-learn.

Jupyter Notebooks offer a good environment for using pandas to do data exploration and modeling, but pandas can also be used in text editors just as easily.

Jupyter Notebooks give us the ability to execute code in a particular cell as opposed to running the entire file. This saves a lot of time when working with large datasets and complex transformations. Notebooks also provide an easy way to visualize pandas' DataFrames and plots. As a matter of fact, this article was created entirely in a Jupyter Notebook.

There are two types of data structures in pandas: Series and DataFrames.

1. **Series:** a pandas Series is a one dimensional data structure ("a one dimensional ndarray") that can store values — and for every value it holds a unique index, too.
2. **DataFrame:** a pandas DataFrame is a two (or more) dimensional data structure – basically a table with rows and columns. The columns have names and the rows have indexes

Those who are familiar with R know the data frame as a way to store data in rectangular grids that can easily be overviewed. Each row of these grids corresponds to measurements or values of an instance, while each column is a vector containing data for a specific variable. This means that a data frame's rows do not need to contain, but can contain, the same type of values: they can be numeric, character, logical, etc.

Now, DataFrames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional labeled data structures with columns of potentially different types.

In general, you could say that the Pandas DataFrame consists of three main components: the

data, the index, and the columns.

Firstly, the DataFrame can contain data that is:

- a Pandas DataFrame
- a Pandas Series: a one-dimensional labeled array capable of holding any data type with axis labels or index. An example of a Series object is one column from a DataFrame.
- a NumPy ndarray, which can be a record or structured
- a two-dimensional ndarray
- dictionaries of one-dimensional ndarray's, lists, dictionaries or Series.

Some of the key features of Python Pandas are as follows:

It provides DataFrame objects with default and customized indexing which is very fast and efficient.

- There are tools available for loading data of different file formats into in-memory data objects.
- It is easy to perform data alignment and integrated handling of missing data in Python Pandas.
- It is very simple to perform pivoting and reshaping of data sets in Pandas.
- It also provides indexing, label-based slicing, and sub-setting of large data sets.
- We can easily insert and delete columns from a data structure.
- Data aggregation and transformations can be done using group by.
- High-performance merging and joining of data can be done using Pandas.
- It also provides time series functionality.
- Inserting and deleting columns in data structures.
- Merging and joining data sets.

- Reshaping and pivoting data sets.

- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for DataFrame objects.
- Performing split-apply-combine on data sets using the group by engine.
- Manipulating high-dimensional data in a data structure with a lower dimension using hierarchical axis indexing.
- Subsetting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other time-series functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending
- Filtering data around a condition.
- Analyzing time series.
- Iterating over a data set.

Numpy

Numpy is the core library for scientific computing in Python. It provides a high-performance multidimensional array object, and tools for working with these arrays. If you are already familiar with MATLAB, you might find this tutorial useful to get started with Numpy.

A numpy array is a grid of values, all of the same type, and is indexed by a tuple of nonnegative integers. The number of dimensions is the rank of the array; the shape of an array is a tuple of integers giving the size of the array along each dimension.

NumPy is, just like SciPy, Scikit-Learn, Pandas, etc. one of the packages that you just can't miss when you're learning data science, mainly because this library provides you with an array data

structure that holds some benefits over Python lists, such as: being more compact, faster access in reading and writing items, being more convenient and more efficient.

NumPy is a Python library that is the core library for scientific computing in Python. It contains a collection of tools and techniques that can be used to solve on a computer mathematical models of problems in Science and Engineering. One of these tools is a high-performance multidimensional array object that is a powerful data structure for efficient computation of arrays and matrices. To work with these arrays, there's a vast amount of high-level mathematical functions operate on these matrices and arrays.

an array is basically nothing but pointers. It's a combination of a memory address, a data type, a shape, and strides:

- The data pointer indicates the memory address of the first byte in the array,
- The data type or dtype pointer describes the kind of elements that are contained within the array,
- The shape indicates the shape of the array, and
- The strides are the number of bytes that should be skipped in memory to go to the next element. If your strides are (10,1), you need to proceed one byte to get to the next column and 10 bytes to locate the next row.

Or, in other words, an array contains information about the raw data, how to locate an element and how to interpret an element.

With NumPy, we work with multidimensional arrays. We'll dive into all of the possible types of multidimensional arrays later on, but for now, we'll focus on 2-dimensional arrays. A 2-dimensional array is also known as a matrix, and is something you should be familiar with. In

fact, it's just a different way of thinking about a list of lists. A matrix has rows and columns. By specifying a row number and a column number, we're able to extract an element from a matrix.

We can create a NumPy array using the `numpy.array` function. If we pass in a list of lists, it will automatically create a NumPy array with the same number of rows and columns. Because we want all of the elements in the array to be float elements for easy computation, we'll leave off the header row, which contains strings. One of the limitations of NumPy is that all the elements in an array have to be of the same type, so if we include the header row, all the elements in the array will be read in as strings. Because we want to be able to do computations like find the average quality of the wines, we need the elements to all be floats.

NumPy has several advantages over using core Python mathematical functions, a few of which are outlined here:

1. NumPy is extremely fast when compared to core Python thanks to its heavy use of C extensions.
2. Many advanced Python libraries, such as Scikit-Learn, Scipy, and Keras, make extensive use of the NumPy library. Therefore, if you plan to pursue a career in data science or machine learning, NumPy is a very good tool to master.

NumPy comes with a variety of built-in functionalities, which in core Python would take a fair bit of custom code.

Matplotlib

Plotting of data can be extensively made possible in an interactive way by Matplotlib, which is a plotting library that can be demonstrated in Python scripts. Plotting of graphs is a part of data visualization, and this property can be achieved by making use of Matplotlib.

Matplotlib makes use of many general-purpose GUI toolkits, such as wxPython, Tkinter, QT, etc., in order to provide object-oriented APIs for embedding plots into applications. John D. Hunter was the person who originally wrote Matplotlib, and its lead developer was Michael Droettboom.

One of the free and open-source Python library which is basically used for technical and scientific computing

is Python SciPy. Matplotlib is widely used in SciPy as most scientific calculations require plotting of graphs and diagrams.

Matplotlib is a plotting library like GNUplot. The main advantage towards GNUplot is the fact that Matplotlib is a Python module. Due to the growing interest in python the popularity of matplotlib is continually rising as well.

Another reason for the attractiveness of Matplotlib lies in the fact that it is widely considered to be a perfect alternative to MATLAB, if it is used in combination with Numpy and Scipy. Whereas MATLAB is expensive and closed source, Matplotlib is free and open source code. It is also object-oriented and can be used in an object oriented way. Furthermore it can be used with general-purpose GUI toolkits like wxPython, Qt, and GTK+. There is also a procedural "pylab", which designed to closely resemble that of MATLAB. This can make it extremely easy for MATLAB users to migrate to matplotlib.

Matplotlib can be used to create publication quality figures in a variety of hardcopy formats and interactive environments across platforms.

Another characteristic of matplotlib is its steep learning curve, which means that users usually make rapid progress after having started. The official website has to say the following about this: "matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, errorcharts, scatterplots, etc, with just a few lines of code."

4.3.2 About IDE

IDE – Integrated Development Environment. It is software suite. It contains different instruments for web application software engineers to compose and test applications. Programming Web application software engineers utilize wide scope of devices habitually all through application

source code creation, building and application testing. Programming Development apparatuses frequently incorporate word processors for source code creation, code libraries for reuse, compilers and test stages. An Integrated Development Environment (IDE) is a product application that gives a

programming domain to streamline creating and troubleshooting programming. Instead of playing out every one of the means required to make an executable program as disconnected individual assignments, it brings every one of the devices required into one application and workspace. Every one of the devices has a familiarity with the earth, and they cooperate to display a consistent improvement set for the designer.

An IDE typically has an editor and a compiler window. There, we can write our code, run it, and compile it. Hence, it becomes much easier to develop applications using an IDE. The major reasons for using an IDE are given below:

We can easily write our code in its text editor window.

To store the resource files of a project, we have a project editor window.

It makes debugging of the code easier.

An IDE's autosuggest feature helps us in writing efficient code.

We can import other projects for integrating with the existing one.

Without an IDE, a product web application software engineer must choose, send, incorporate and deal with these instruments independently. An IDE brings huge numbers of those improvement related instruments all together system, application or administration.

The incorporated toolset is intended to streamline programming advancement and can distinguish and limit coding missteps and grammatical errors.

- ✓ Komodo
- ✓ NetBeans
- ✓ PyCharm
- ✓ Python for Visual Studio.NET
- ✓ PyDev
- ✓ PyStudio

✓ LiClipse (Eclipse)

LiClipse is at the core for all Python based application development activities. It's been available for many years. LiClipse is a set of plugins to enhance Eclipse. It improves the overall Eclipse experience. It is best IDE tool to develop Python Applications.

Advantages of LiClipse

- Perfect Debugger
- Error checking features
- An enhanced ability to analyze the code for quality and security concerns
- Auto complete feature

Based on the various IDE that are described above, we decided to use LiClipse as IDE (Integrated Development Environment) for developing this e-Voting web based application.

Important Features That Make LiClipse our IDE of Choice

- Rich Reusable Libraries are available
- Model Driven Development
- Enterprise Python Toolin

IDE is a program that is used to write applications (other programs) in a variety of programming languages. It allows you to complete many distinct program development steps in one place (program window). A program which allows you to do this is known as an integrated development environment, or IDE. The program is free and easy to import onto your home computer.

LiClipse is used for writing programs in Python. There are several steps that must be completed to set up LiClipse and further steps to start up a new Python project (set of Python components which work together to make up an application).

PyCharm is an Integrated Development Environment (IDE) that provides tools for writing efficient code. It supports various frameworks such as Django, Flask, and Pyramid. PyCharm is majorly used by developers for application development with Python language.

Let's suppose, we are making a project using Python programming language. We know that Python is extensively popular among the developer's community, as well as in industries, due to its following features:

Simple to code: Python is a very simple language compared to other programming languages. Also, it is a developer-friendly language.

Object-oriented language: Python is an object-oriented language that uses concepts of classes and objects.

GUI programming support: We can create graphical user interfaces using Python Tkinter library. Besides, we can use modules such as PyQt5, PyQt4, and wxPython to create a GUI application.

High-level language: Python is a high-level programming language. We do not need to take care of managing memory or the system architecture while writing Python codes.

Portability: Python is a portable programming language, i.e., we can run Python codes on any platform. For example, we can run a code written for Windows in Linux, UNIX, and Mac, without changing it.

Integrated language: Python is an integrated programming language. We can easily integrate C, C++, etc. with Python.

Interpreted language: Python is an interpreted programming language as well. Python's source code is changed to bytecode. Due to this, it is directly executed without compiling.

Large set of standard libraries: Python consists of a rich set of modules, packages, and libraries. It helps in various stages of application development.

The Jupyter Notebook App produces these documents.

As a server-client application, the Jupyter Notebook App allows you to edit and run your notebooks via a web browser. The application can be executed on a PC without Internet access, or it can be installed on a remote server, where you can access it through the Internet.

Its two main components are the kernels and a dashboard.

A kernel is a program that runs and introspects the user's code. The Jupyter Notebook App has a

kernel for Python code, but there are also kernels available for other programming languages.

The dashboard of the application not only shows you the notebook documents that you have made and can reopen but can also be used to manage the kernels: you can which ones are running and shut them down if necessary.

CHAPTER 5

SYSTEM DESIGN

5.1 Introduction

A UML diagram is a diagram based on the UML (Unified Modeling Language) with the purpose of visually representing a system along with its main actors, roles, actions, artifacts or classes, in order to better understand, alter, maintain, or document information about the system.

Mainly, UML has been used as a general-purpose modeling language in the field of software engineering. However, it has now found its way into the documentation of several business processes or workflows. For example, activity diagrams, a type of UML diagram, can be used as a replacement for flowcharts. They provide both a more standardized way of modeling workflows as well as a wider range of features to improve readability and efficiency.

5.2 Architecture Diagram

An architecture diagram is a graphical representation of a set of concepts, that are part of an architecture, including their principles, elements and components.

What is a diagram? What are the types of diagrams for architecture? The Dragon1 open EA Method makes it very clear: if a diagram does not show a concept, principle or part of a principle, it is NOT an architecture diagram, because it does not show (a part of) the architecture.

There are many kinds of architecture diagrams, like a software architecture diagram, system architecture diagram, application architecture diagram, security architecture diagram, etc.

For system developers, they need system architecture diagrams to understand, clarify, and communicate ideas about the system structure and the user requirements that the system must support. It's a basic framework can be used at the system planning phase helping partners understand the architecture, discuss changes, and communicate intentions clearly.

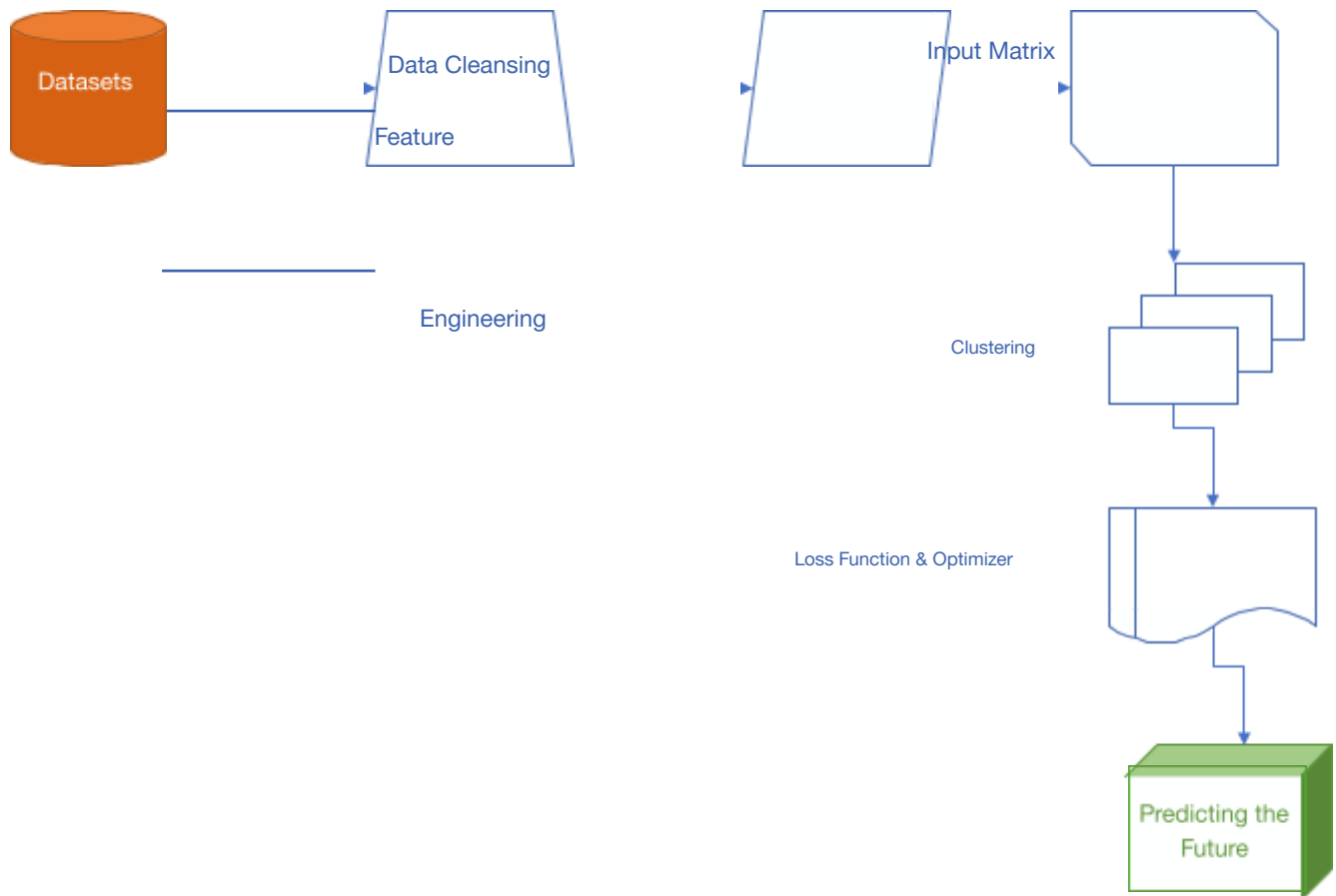


FIG 5.1 – 1 ARCHITECTURE DIAGRAM

SYSTEM IMPLEMENTATION**6.1 LIST OF MODULES**

Module 1 : Exploratory Data

Evaluation Module 2 : Pre-processing

Module 3 : Feature

Engineering Module 4 :

Prediction

6.1.1 Moduel 1 : Exploratory Data Evaluation

Exploratory Data Analysis (EDA) is the first step in your data analysis process. Here, you make sense of the data you have and then figure out what questions you want to ask and how to frame them, as well as how best to manipulate your available data sources to get the answers you need. You do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in your existing data, using visual and quantitative methods to get a sense of the story this tells.

Exploratory Data Analysis is valuable to data science projects since it allows to get closer to the certainty that the future results will be valid, correctly interpreted, and applicable to the desired business contexts. Such level of certainty can be achieved only after raw data is validated and checked for anomalies, ensuring that the data set was collected without errors. EDA also helps to find insights that were not evident or worth investigating to business stakeholders and data scientists but can be very informative about a particular business.

EDA is performed in order to define and refine the selection of feature variables that will be used for machine learning. Once data scientists become familiar with the data set, they often have to return to feature engineering step, since the initial features may turn out not to be serving their intended purpose. Once the EDA stage is complete, data scientists get a firm feature set they

need for supervised and unsupervised machine learning.

6.1.2 Module 2 Pre-Processing

Sometimes you may find some data are missing in the dataset. We need to be equipped to handle the problem when we come across them. Obviously you could remove the entire line of data but

what if you are unknowingly removing crucial information? Of course we would not want to do that. One of the most common idea to handle the problem is to take a mean of all the values of the same column and have it to replace the missing data.

The library that we are going to use for the task is called Scikit Learn preprocessing. It contains a class called Imputer which will help us take care of the missing data.

Sometimes our data is in qualitative form, that is we have texts as our data. We can find categories in text form. Now it gets complicated for machines to understand texts and process them, rather than numbers, since the models are based on mathematical equations and calculations. Therefore, we have to encode the categorical data.

Now we need to split our dataset into two sets — a Training set and a Test set. We will train our machine learning models on our training set, i.e our machine learning models will try to understand any correlations in our training set and then we will test the models on our test set to check how accurately it can predict. A general rule of the thumb is to allocate 80% of the dataset to training set and the remaining 20% to test set. For this task, we will import `test_train_split` from `model_selection` library of `scikit`.

6.1.3 Module 3 : Feature Engineering

Filter methods are generally used as a preprocessing step. The selection of features is independent of any machine learning algorithms. Instead, features are selected on the basis of their scores in various statistical tests for their correlation with the outcome variable. The correlation is a subjective term here. For basic guidance, you can refer to the following table for defining correlation co- efficient.

Pearson's Correlation: It is used as a measure for quantifying linear dependence between two continuous variables X and Y. Its value varies from -1 to +1.

LDA: Linear discriminant analysis is used to find a linear combination of features that characterizes or separates two or more classes (or levels) of a categorical variable.

ANOVA: ANOVA stands for Analysis of variance. It is similar to LDA except for the fact that it is operated using one or more categorical independent features and one continuous dependent

feature. It provides a statistical test of whether the means of several groups are equal or not.

Chi-Square: It is a statistical test applied to the groups of categorical features to evaluate the likelihood of correlation or association between them using their frequency distribution.

6.1.4 Module 4: Prediction

Once training is complete, it's time to see if the model is any good, using Evaluation. This is where that dataset that we set aside earlier comes into play. Evaluation allows us to test our model against data that has never been used for training. This metric allows us to see how the model might perform against data that it has not yet seen. This is meant to be representative of how the model might perform in the real world.

A good rule of thumb I use for a training-evaluation split somewhere on the order of 80/20 or 70/30. Much of this depends on the size of the original source dataset. If you have a lot of data, perhaps you don't need as big of a fraction for the evaluation dataset.

Once you've done evaluation, it's possible that you want to see if you can further improve your training in any way. We can do this by tuning our parameters. There were a few parameters we implicitly assumed when we did our training, and now is a good time to go back and test those assumptions and try other values.

A tree has many analogies in real life, and turns out that it has influenced a wide area of machine learning, covering both classification and regression. In decision analysis, a decision tree can be used to visually and explicitly represent decisions and decision making. As the name goes, it uses a tree-like model of decisions. A decision tree is drawn upside down with its root at the top. In the image on the left, the bold text in black represents a condition/internal node, based on which the tree splits into branches/ edges. The end of the branch that doesn't split anymore is the decision/leaf, in this case, whether the passenger died or survived, represented as red and green text respectively. Although, a real dataset will have a lot more features and this will just be a branch in a much bigger tree, but you can't ignore the simplicity of this algorithm. The feature importance is clear and relations can be viewed easily.

CHAPTER 7

SOFTWARE

TESTING

Testing documentation is the documentation of artifacts that are created during or before the testing of a software application. Documentation reflects the importance of processes for the customer, individual and organization. Projects which contain all documents have a high level of maturity. Careful documentation can save the time, efforts and wealth of the organization.

If the testing or development team gets software that is not working correctly and developed by someone else, so to find the error, the team will first need a document. Now, if the documents are available then the team will quickly find out the cause of the error by examining documentation. But, if the documents are not available then the tester need to do black box and white box testing again, which will waste the time and money of the organization. More than that, Lack of documentation becomes a problem for acceptance.

Benefits of using Documentation

- Documentation clarifies the quality of methods and objectives.
- It ensures internal coordination when a customer uses software application.
- It ensures clarity about the stability of tasks and performance.
- It provides feedback on preventive tasks.
- It provides feedback for your planning cycle.
- It creates objective evidence for the performance of the quality management system.

The test scenario is a detailed document of test cases that cover end to end functionality of a software application in liner statements. The liner statement is considered as a scenario. The test scenario is a high-level classification of testable requirements. These requirements are grouped on the basis of the functionality of a module and obtained from the use cases.

In the test scenario, there is a detailed testing process due to many associated test cases. Before performing the test scenario, the tester has to consider the test cases for each scenario.

In the test scenario, testers need to put themselves in the place of the user because they test the software application under the user's point of view. Preparation of scenarios is the most critical part, and it is necessary to seek advice or help from customers, stakeholders or developers to prepare the scenario.

As per the IEEE Documentation describing plans for, or results of, the testing of a system or component, Types include test case specification, test incident report, test log, test plan, test procedure, test report. Hence the testing of all the above mentioned documents is known as documentation testing.

This is one of the most cost effective approaches to testing. If the documentation is not right: there will be major and costly problems. The documentation can be tested in a number of different ways to many different degrees of complexity. These range from running the documents through a spelling and grammar checking device, to manually reviewing the documentation to remove any ambiguity or inconsistency.

Documentation testing can start at the very beginning of the software process and hence save large amounts of money, since the earlier a defect is found the less it will cost to be fixed.

the most popular testing documentation files are test reports, plans, and checklists. These documents are used to outline the team's workload and keep track of the process. Let's take a look at the key requirements for these files and see how they contribute to the process.

Test strategy. An outline of the full approach to product testing. As the project moves along, developers, designers, product owners can come back to the document and see if the actual performance corresponds to the planned activities.

Test data. The data that testers enter into the software to verify certain features and their outputs. Examples of such data can be fake user profiles, statistics, media content, similar to files that would be uploaded by an end-user in a ready solution.

Test plans. A file that describes the strategy, resources, environment, limitations, and schedule of the testing process. It's the fullest testing document, essential for informed planning. Such a

document is distributed between team members and shared with all stakeholders.

Test scenarios. In scenarios, testers break down the product's functionality and interface by modules and provide real-time status updates at all testing stages. A module can be described by a single statement, or require hundreds of statuses, depending on its size and scope.

Test cases. If the test scenario describes the object of testing (what), a scenario describes a procedure (how). These files cover step-by-step guidance, detailed conditions, and current inputs of a testing task. Test cases have their own kinds that depend on the type of testing — functional, UI, physical, logical cases, etc. Test cases compare available resources and current conditions with desired outcomes and determine if the functionality can be released or not.

Traceability Matrix. This software testing documentation maps test cases and their requirements. All entries have their custom IDs — team members and stakeholders can track the progress of any tasks by simply entering its ID to the search.

External documentation collects information from inner documentation but also emphasize on providing a visual data representation — graphs, diagrams, etc.

External reports — these documents collect information on test results and can describe an entire project or a particular piece of functionality.

Test summary report — the file with final test results and findings, presented to stakeholders.

Bug reports — such files keep track of newly encountered bugs and their fixes. We prefer to keep our bug documentation numbered, so it's easier to mention them in further documentation. Reports are concise and focus on offering tangible solutions. Sometimes, bug reports can only include issue description, if the team hasn't yet found the best approach to fixing the problem.

The combination of internal and external documentation is the key to a deep understanding of all testing processes. Although stakeholders typically have access to the majority of documentation, they mostly work with external files, since they are more concise and tackle tangible issues and results. Internal files, on the other hand, are used by team members to optimize the testing process. Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

But as a curiosity or Out of the box thinking, have you ever wondered how developers test their own

code? What method do they use to test before releasing code for testing? How is dev-testing

important in an agile process? The answer to all this is Unit Testing. I want to educate you on the importance of Unit Testing so that development and testing teams can work more collaboratively to design, test and release an excellent application.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.



FIG 7 -1 IMPLEMENTATION

Unit Testing is used to design robust software components that help maintain code and eliminate the issues in code units. We all know the importance of finding and fixing defects in the early stage of the software development cycle. Unit Testing serves the same purpose.

Unit Testing is an integral part of the agile software development process. When a nightly build run unit test suite should run and report should be generated. If any of the unit tests have failed

then the QA team should not accept that build for verification.

If we set this as a standard process, many defects would be caught in the early development cycle, saving much testing time.

I know many developers hate to write unit tests. They either ignore or write bad unit test cases due to tight scheduled or lack of seriousness (yea they write empty unit tests, so 100% of them pass successfully ;-)). It's important to write good unit tests or don't write them at all. It's even more important to provide sufficient time and a supportive environment for real benefits.

- Testing can be done in the early phases of the software development lifecycle when other modules may not be available for integration
- Fixing an issue in Unit Testing can fix many other issues occurring in later development and testing stages
- Cost of fixing a defect found in Unit Testing is very less than the one found in the system or acceptance testing
- Code coverage can be measured
- Fewer bugs in the System and Acceptance testing
- Code completeness can be demonstrated using unit tests. This is more useful in the agile process. Testers don't get the functional builds to test until integration is completed. Code completion cannot be justified by showing that you have written and checked in the code. But running Unit tests can demonstrate code completeness.
- Expect robust design and development as developers write test cases by understanding the specifications first.
- Easily identify who broke the build
- Saves development time: Code completion may take more time but due to decreased defect count overall development time can be saved.

7.1 General

Unit Testing frameworks are mostly used to help write unit tests quickly and easily. Most of the programming languages do not support unit testing with the inbuilt compiler. Third-party open source and commercial tools can be used to make unit testing even more fun.

List of popular Unit Testing tools for different programming languages:

Java framework – JUnit

PHP framework – PHPUnit

C++ frameworks – UnitTest++ and Google C++

.NET framework – NUnit

Python framework – py.test

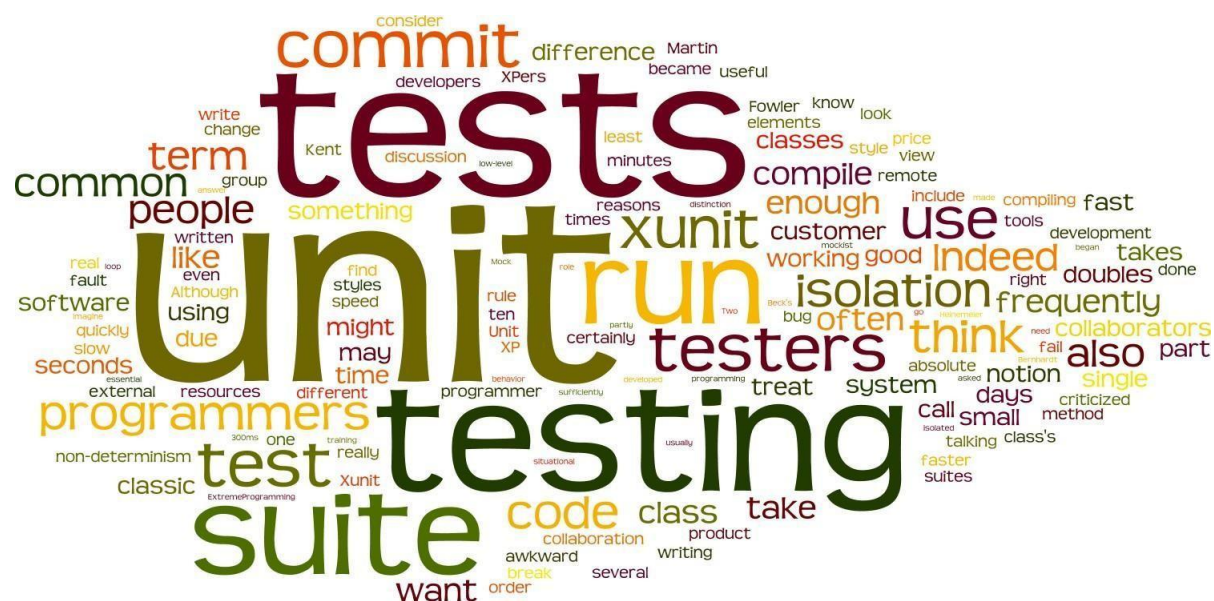


FIG 7-2 BENEFITS OF TESTING

Functional Testing is a type of black box testing whereby each part of the system is tested against functional specification/requirements. For instance, seek answers to the following questions,

Are you able to login to a system after entering correct credentials?

Does your payment gateway prompt an error message when you enter incorrect card number?

Does your “add a customer” screen adds a customer to your records successfully?

Well, the above questions are mere samples to perform full-fledged functional testing of a system.

7.2 Test Driven Development

Test Driven Development, or TDD, is a code design technique where the programmer writes a test before any production code, and then writes the code that will make that test pass. The idea is that with a tiny bit of assurance from that initial test, the programmer can feel free to refactor and refactor some more to get the cleanest code they know how to write. The idea is simple, but like most simple things, the execution is hard. TDD requires a completely different mind set from what most people are used to and the tenacity to deal with a learning curve that may slow you down at first.

Functional Testing types include:

1. Unit Testing
2. Integration Testing
3. System Testing
4. Sanity Testing
5. Smoke Testing
6. Interface Testing
7. Regression Testing
8. Beta/Acceptance Testing

9. Non-functional Testing types include:

Performance Testing

- Load Testing
- Stress Testing
- Volume Testing
- Security Testing

- Compatibility Testing
- Install Testing

- Recovery Testing
- Reliability Testing
- Usability Testing
- Compliance Testing
- Localization Testing

7.3 Unit Testing

UNIT TESTING is a level of software testing where individual units/ components of a software are tested. The purpose is to validate that each unit of the software performs as designed. A unit is the smallest testable part of any software. It usually has one or a few inputs and usually a single output. In procedural programming, a unit may be an individual program, function, procedure, etc. In object- oriented programming, the smallest unit is a method, which may belong to a base/ super class, abstract class or derived/ child class. (Some treat a module of an application as a unit. This is to be discouraged as there will probably be many individual units within that module.) Unit testing frameworks, drivers, stubs, and mock/ fake objects are used to assist in unit testing. A unit can be almost anything you want it to be -- a line of code, a method, or a class. Generally though, smaller is better. Smaller tests give you a much more granular view of how your code is performing. There is also the practical aspect that when you test very small units, your tests can be run fast; like a thousand tests in a second fast.

Black Box testers don't care about Unit Testing. Their main goal is to validate the application against the requirements without going into the implementation details.

Unit Testing is not a new concept. It's been there since the early days of programming. Usually, developers and sometimes White box testers write Unit tests to improve code quality by verifying each and every unit of the code used to implement functional requirements (aka test drove development TDD or test-first development).

Most of us might know the classic definition of Unit Testing –

“Unit Testing is the method of verifying the smallest piece of testable code against its purpose.” If the purpose or requirement failed then the unit test has failed.

In simple words, Unit Testing means – writing a piece of code (unit test) to verify the code (unit) written for implementing requirements.

7.4 Blackbox Testing

During functional testing, testers verify the app features against the user specifications. This is completely different from testing done by developers which is unit testing. It checks whether the code works as expected. Because unit testing focuses on the internal structure of the code, it is called the white box testing. On the other hand, functional testing checks app’s functionalities without looking at the internal structure of the code, hence it is called black box testing. Despite how flawless the various individual code components may be, it is essential to check that the app is functioning as expected, when all components are combined. Here you can find a detailed comparison between functional testing vs unit testing.

7.5 Integration Testing

INTEGRATION TESTING is a level of software of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

integration testing: Testing performed to expose defects in the interfaces and in the interactions between integrated components or systems. See also component integration testing, system integration testing.

component integration testing: Testing performed to expose defects in the interfaces and interaction between integrated components. System integration testing: Testing the integration of systems and packages; testing interfaces to external organizations (e.g. Electronic Data Interchange, Internet). Integration tests determine if independently developed units of software

work correctly when they are connected to each other. The term has become blurred even by the diffuse standards of the

software industry, so I've been wary of using it in my writing. In particular, many people assume

integration tests are necessarily broad in scope, while they can be more effectively done with a narrower scope.

As often with these things, it's best to start with a bit of history. When I first learned about integration testing, it was in the 1980's and the waterfall was the dominant influence of software development thinking. In a larger project, we would have a design phase that would specify the interface and behavior of the various modules in the system. Modules would then be assigned to developers to program. It was not unusual for one programmer to be responsible for a single module, but this would be big enough that it could take months to build it. All this work was done in isolation, and when the programmer believed it was finished they would hand it over to QA for testing.

Integration testing tests integration or interfaces between components, interactions to different parts of the system such as an operating system, file system and hardware or interfaces between systems. Integration testing is a key aspect of software testing.

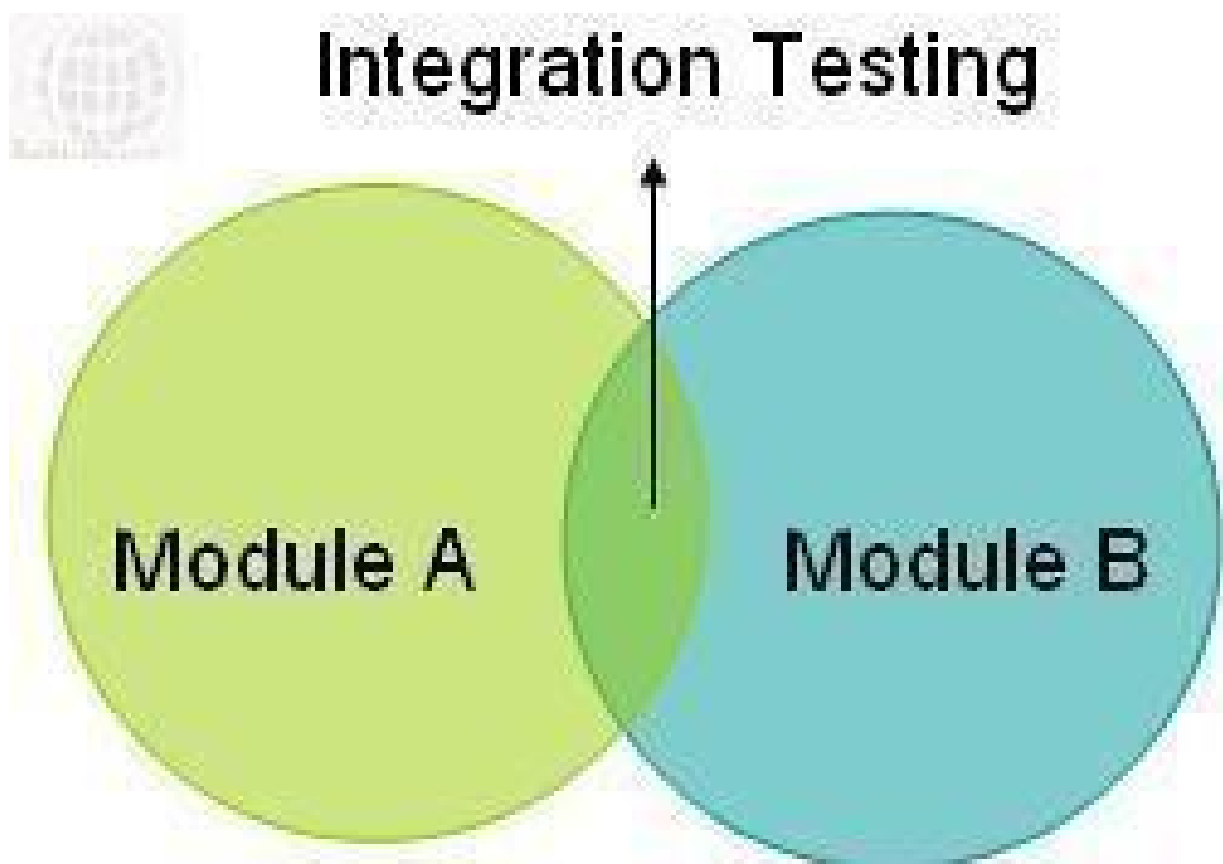


FIG 7 -3 INTEGRATION TESTING

7.6 System Testing

System Testing is a level of software testing where a complete and integrated software is tested. The purpose of this test is to evaluate the system's compliance with the specified requirements. System Testing means testing the system as a whole. All the modules/components are integrated in order to verify if the system works as expected or not. System Testing is done after Integration Testing. This plays an important role in delivering a high-quality product.

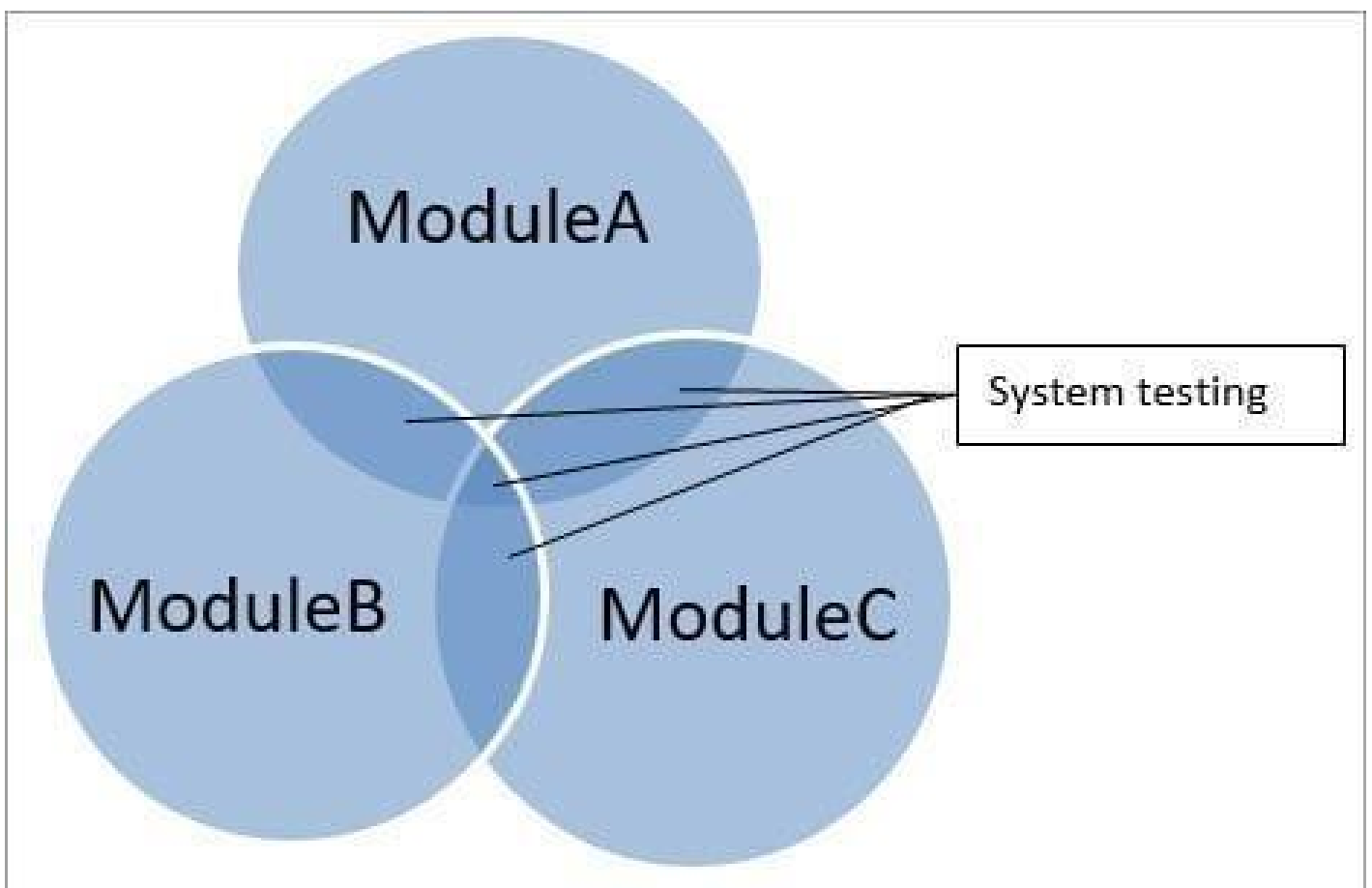


FIG 7 -4 SYSTEM TESTING

System testing is a method of monitoring and assessing the behaviour of the complete and fully-integrated software product or system, on the basis of pre-decided specifications and functional requirements. It is a solution to the question "whether the complete system functions in accordance to its pre-defined requirements?"

It's comes under black box testing i.e. only external working features of the software are evaluated during this testing. It does not requires any internal knowledge of the coding, programming, design, etc., and is completely based on users-perspective.

A black box testing type, system testing is the first testing technique that carries out the task of testing a software product as a whole. This System testing tests the integrated system and validates whether it meets the specified requirements of the client.

System testing is a process of testing the entire system that is fully functional, in order to ensure the system is bound to all the requirements provided by the client in the form of the functional specification or system specification documentation. In most cases, it is done next to the Integration testing, as this testing should be covering the end-to-end system's actual routine. This type of testing requires a dedicated Test Plan and other test documentation derived from the system specification document that should cover both software and hardware requirements. By this test, we uncover the errors. It ensures that all the system works as expected. We check System performance and functionality to get a quality product. System testing is nothing but testing the system as a whole. This testing checks complete end-to-end scenario as per the customer's point of view. Functional and Non-Functional tests also done by System testing. All things are done to maintain trust within the development that the system is defect-free and bug-free. System testing is also intended to test hardware/software requirements specifications. System testing is more of a limited type of testing; it seeks to detect both defects within the "inter-assemblages".

7.7 Sanity Testing

Sanity Testing is done when as a QA we do not have sufficient time to run all the test cases, be it Functional Testing, UI, OS or Browser Testing. Sanity testing is a subset of regression testing. After receiving the software build, sanity testing is performed to ensure that the code changes introduced are working as expected .This testing is a checkpoint to determine if testing for the

build can proceed or not.

The main purpose of this testing is to determine that the changes or the proposed functionality are working as expected. If the sanity test fails, the build is rejected by the testing team to save time

and money. It is performed only after the build has cleared the smoke test and been accepted by the Quality Assurance team for further testing. The focus of the team during this testing process is to validate the functionality of the application and not detailed testing. Let's assume that in an e-commerce project, there are mainly five modules- like login page, home page, user profile page, new user creation, etc. There is a defect in the login page when the password field accepts less than four alpha numeric characters and the requirement mentions that this password field should not be below eight characters. So, the bug is reported by the testing team to the development team to resolve it. When the development team fixes the defect and passes it to the testing team. Then the testing team checks if the changes done are working fine. It also checks if it does not have an impact on other related functionalities. Now there is a functionality to update the password in the user profile page. As part of the sanity check, we would need to validate the login page as well as the profile page to ensure that the checks are working fine at both the places. Sanity tests are generally performed on build where the production deployment is required immediately - like a critical bug fix.

Smoke Testing is done to make sure if the build we received from the development team is testable or not. It is also called as "Day 0" check. It is done at the "build level".

It helps not to waste the testing time to simply testing the whole application when the key features don't work or the key bugs have not been fixed yet. Here our focus will be on primary and core application work flow.

To conduct smoke testing, we don't write test cases. We just pick the necessary test cases from already written test cases.

Do we really write test cases for all testing types? Here in this article, , we have given clear idea on choosing testing types to write test cases.

As mentioned earlier, here in Smoke Testing, our main focus will be on core application work flow. So we pick the test cases from our test suite which cover major functionality of the application. In general, we pick minimal number of test cases that won't take more than half an

hour to execute. After receiving a Software build with the minor issues fixes in code or functionality, Sanity testing is carry out to check whether the bugs reported in previous build are fixed & there is regression

introduced due to these fixes i.e. not breaking any previously working functionality. The main aim of Sanity testing to check the planned functionality is working as expected. Instead of doing whole regression testing the Sanity testing is perform.

Sanity tests helps to avoid wasting time and cost involved in testing if the build is failed. Tester should reject the build upon build failure.

After completion of regression testing the Sanity testing is started to check the defect fixes & changes done in the software application is not breaking the core functionality of the software. Typically this is done nearing end of SDLC i.e. while releasing the software. You can say that sanity testing is a subset of acceptance testing. We can also say Tester Acceptance Testing for Sanity testing.

7.8 Regression Testing

Regression Testing is a type of testing that is done to verify that a code change in the software does not impact the existing functionality of the product. This is to make sure the product works fine with new functionality, bug fixes or any change in the existing feature. Previously executed test cases are re-executed in order to verify the impact of change.

Regression Testing is a Software Testing type in which test cases are re-executed in order to check whether the previous functionality of the application is working fine and the new changes have not introduced any new bugs.

This test can be performed on a new build when there is a significant change in the original functionality that too even in a single bug fix.

For regression testing to be effective, it needs to be seen as one part of a comprehensive testing methodology that is cost-effective and efficient while still incorporating enough variety—such as well-designed frontend UI automated tests alongside targeted unit testing, based on smart risk prioritization—to prevent any aspects of your software applications from going unchecked. These days, many Agile work environments employing workflow practices such as XP (Extreme Programming), RUP (Rational Unified Process), or Scrum appreciate regression testing as an

essential aspect of a dynamic, iterative development and deployment schedule.

But no matter what software development and quality-assurance process your organization uses, if you take the time to put in enough careful planning up front, crafting a clear and diverse testing strategy with automated regression testing at its core, you can help prevent projects from going over budget, keep your team on track, and, most importantly, prevent unexpected bugs from damaging your products and your company's bottom line.

Performance testing is the practice of evaluating how a system performs in terms of responsiveness and stability under a particular workload. Performance tests are typically executed to examine speed, robustness, reliability, and application size. The process incorporates "performance" indicators such as:

- Browser, page, and network response times
- Server request processing times
- Acceptable concurrent user volumes
- Processor memory consumption; number and type of errors that might be encountered with app

Performance testing gathers all the tests that verify an application's speed, robustness, reliability, and correct sizing. It examines several indicators such as a browser, page and network response times, server query processing time, number of acceptable concurrent users architected, CPU memory consumption, and number/type of errors which may be encountered when using an application.

Performance testing is the testing that is performed to ascertain how the components of a system are performing under a certain given situation.

Resource usage, scalability, and reliability of the product are also validated under this testing. This testing is the subset of performance engineering, which is focused on addressing performance issues in the design and architecture of a software product.



FIG 7 -5 PERFORMANCE TESTING

Software Performance testing is type of testing perform to determine the performance of system to major the measure, validate or verify quality attributes of the system like responsiveness, Speed, Scalability, Stability under variety of load conditions. The system is tested under a mixture of load conditions and check the time required responding by the system under varying workloads. Software performance testing involves the testing of application under test to ensure

that application is

working as expected under variety of load conditions. The goal of performance testing is not only find the bugs in the system but also eliminate the performance bottlenecks from the system.

Load Testing is type of performance testing to check system with constantly increasing the load on the system until the time load is reaches to its threshold value. Here Increasing load means increasing number of concurrent users, transactions & check the behavior of application under test. It is normally carried out underneath controlled environment in order to distinguish between two different systems. It is also called as “Endurance testing” and “Volume testing”. The main purpose of load testing is to monitor the response time and staying power of application when system is performing well under heavy load. Load testing comes under the Non Functional Testing & it is designed to test the non-functional requirements of a software application.

Load testing is perform to make sure that what amount of load can be withstand the application under test. The successfully executed load testing is only if the specified test cases are executed without any error in allocated time.

Simple examples of load testing:

- Testing printer by sending large job.
- Editing a very large document for testing of word processor
- Continuously reading and writing data into hard disk.
- Running multiple applications simultaneously on server.
- Testing of mail server by accessing thousands of mailboxes
- In case of zero-volume testing & system fed with zero load..

CHAPTER 8

CONCLUSION & FUTURE ENHANCEMENT

8.1 Conclusion

It is more often used for evaluating patients with suspected acute stroke which represents high cost and low value endeavor. We have developed an algorithm for helping clinicians determine which patients evaluated for acute stroke need a classifiers. This algorithm has superior diagnostic accuracy in comparison to the algorithm, which is currently the most accurate and widely used algorithm for determining which acute stroke patient needs a classifiers.

Using readily available clinical information, our algorithm is capable of helping clinicians avoid nearly two times the number of acute stroke that can be avoided when the algorithm is utilized. Clinical implementation of this algorithm can help decrease financial costs associated with evaluations for traumatic brain injury, while decreasing patient exposure to avoidable ionizing radiation. Although the algorithm currently utilizes only easily obtainable clinical data

8.2 Future Enhancement

In our future work, we want to consider also time characteristics of the traffic by using the time series methods. Moreover, we plan to predict also other parameters that describe demands in dynamic optical networks, namely, demand volume and holding time.

REFERENCES

- L. Velasco and L. Gifre, "iONE: A workflow-oriented ABNO implementation," in Proc. of the Photonics in Switching Conf., 2015.
- L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco, "Big data analytics in support of virtual network topology adaptability," in Optical Fiber Communication Conf. (OFC), 2016.
- F. Morales, M. Ruiz, and L. Velasco, "Virtual network topology reconfiguration based on big data analytics for traffic prediction," in Optical Fiber Communication Conf. (OFC), 2016.
- M. R. Rufuie, E. Gad, M. Nakhla, and R. Achar, "Generalized Hermite polynomial chaos for variability analysis of macromodels embedded in nonlinear circuits," IEEE Trans. Compon. Packag. Manuf. Technol., vol. 4, no. 4, pp. 673–684, Apr. 2014.
- D. Spina, D. De Jonghe, D. Deschrijver, G. Gielen, L. Knockaert, and T. Dhaene, "Stochastic macromodeling of nonlinear systems via polynomial chaos expansion and transfer function trajectories," IEEE Trans. Microw. Theory Techn., vol. 62, no. 7, pp. 1454–1460, Jul. 2014.
- P. Manfredi, D. Vande Ginste, D. De Zutter, and F. G. Canavero, "Stochastic modeling of nonlinear circuits via SPICE-compatible spectral equivalents," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 7, pp. 2057–2065, Jul. 2014.
- P. Manfredi, D. Vande Ginste, D. De Zutter, and F. G. Canavero, "Generalized decoupled polynomial chaos for nonlinear circuits with many random parameters," IEEE Microw. Wireless Compon. Lett., vol. 25, no. 8, pp. 505–507, Aug. 2015.
- M. Ahadi and S. Roy, "Sparse linear regression (SPLINER) approach for efficient multidimensional uncertainty quantification of high-speed circuits," IEEE Trans. Comput.-Aided Des. Integr. Circuits Syst., vol. 35, no. 10, pp. 1640–1652, Oct. 2016.
- M. Larbi, I. S. Stievano, F. G. Canavero, and P. Besnier, "Variability impact of many design parameters: The case of a realistic electronic link," IEEE Trans. Electromagn. Compat., vol. 60, no. 1 pp. 34–41, Feb. 2018.
- F. Gao. "Network traffic prediction based on neural network," Proceedings of 2015 International Conference on Intelligent Transportation, Big Data and Smart City, pp. 527-530, 2016.
- B. S. Mi, C. G. Song. "Prediction models for network multi-source dissemination of information based on multivariate chaotic time series," 2017 3rd IEEE International Conference on Computer and Communications, ICC 2017, vol. 2018-January, pp. 767-771, 2018.
- C. Xue, H. B. Cao, W. W. Xu, X. Gu, J. Wang. "A short-term power forecasting model for photovoltaic plants based on data mining," 2017 3rd IEEE International Conference on Computer and Communications, ICC 2017, vol. 2018-January, pp. 2873-2878, 2018.
- K. Zhang, Y. Chai, X. A. Fu. "A network traffic prediction model based on recurrent wavelet neural network," Proceedings of 2nd International Conference on Computer Science and Network Technology, pp. 1630- 1633, 2012.
- X. Wang, C. Zhang, S. Zhang. "Modified Elman neural network and its application to network traffic prediction," Proceedings of 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence

Systems, vol. 2, pp. 629-633, 2013.

Y. Tian, L. Pan. "Predicting short-term traffic flow by Long ShortTerm Memory recurrent neural network," Proceedings - 2015 IEEE International Conference on Smart City, Held Jointly with 8th IEEE International Conference on Social Computing and Networking, 5th IEEE International Conference on Sustainable Computing and Communications, pp. 153-158, 2015.

H. Shao, B. H. Soong. "Traffic flow prediction with Long Short-Term Memory networks (KNNs)," IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 2986-2989, 2017.

Q. Zhuo. Z, Q. M. Li, H. Yan, Y. Qi. "long short-term memory neural network for network traffic prediction," proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering, vol. 2018-January, pp. 1-6, 2018.

APPENDIX

A. PLAGIARISM REPORT



Don't Worry! This report is 100% safe & secure. It's not available publically and it's not accessible by search engines (Google, Yahoo,

Bing, etc)

Sentence

An Unified & Reliable Methodology for High Speed Link Analysis using Vector Quantization *A Deep Learning Approach K. Janaki Rami Reddy¹, K. Satya Srinivas², S. L. Jany Shabu³ ¹ Student, Sathyabama Institute of Science and Technology, Chennai ² Student, Sathyabama Institute of Science and Technology, Chennai ³ Assistant Professor, Sathyabama Institute of Science and Technology, Chennai Abstract— For any kinds of network, a vital execution constraint is its connection network. Studying the energy restriction and the vibrant ecosystem, engineers typically needs to maintain the network being completely linked maximum possible time, so as to guarantee a greatest potential coverage. Be that as it may, most methodologies in the current writing zeroed in on connect utilization in network advancement. The vital attributes of metropolitan transportation frameworks were not thought of and the idea of criticality of connections was stirred up with the idea of weakness. By recognizing the basic parts of the conveyance network, fitting methodologies can be created to improve unwavering quality, remedial activity, a severe fix methodology for basic focuses, and so forth Keywords— Deep Learning, Input Output Layer, Optimization. A General System Wireless sensor networks are made out of little and cheap sensor hubs that have obliged assets. Such networks are required to be utilized in an assortment of uses. In certain applications, the network is conveyed in a threatening climate, and sensors might be under various dangers. Contrasting and other potential dangers, sending a wormhole assault is simpler and may make more noteworthy harm the network. In this manner, how to distinguish the presence of wormholes turns into an indispensable issue for sensor networks. Software Engineering starts with a need dependent on a genuine or saw inadequacy where this converts into characterizing the issue. Installed inside this cycle is the criticism circle. It keeps experts from building up an unnecessary arrangement on the grounds that the criticism circle compels them to consistently screen the cycle to guarantee they address the client's issue. This paper characterizes the criticality of connections from two points of view, i.e., the weakness and potential, in light of which a novel technique for recognizing basic connections in a metropolitan transportation network is proposed. This tale procedure incorporates a positioning technique and a novel mesoscopic model to analyse the metropolitan transportation network execution. Profound learning is a specific subfield of Artificial Intelligence another translation of taking in depictions from data that puts an emphasis on learning reformist layers of continuously critical depictions. The some place down in significant learning isn't a reference to such a more significant course of action achieved by the approach; rather, it addresses this considered reformist layers of depictions The number of layers add to a model of the information is known as the profundity of the model. Other fitting names for the field might have been layered portrayals learning and various levelled portrayals learning. Present day profound adapting frequently includes tens or even many progressive layers of portrayals—and they're totally gained consequently from openness to preparing information. In the meantime, different ways to

deal with AI will in general zero in on learning just a couple of layers of portrayals of the information; henceforth, they're some of the time called shallow learning. Neural Network Artificial Neural

Network An artificial neural network attempts to mirror the mind at its most essential level, i.e., that of the neuron. An artificial neuron has a comparative construction to that of a human neuron and contains the accompanying areas: Info layer: This layer resembles dendrites and takes contribution from various organizations/neurons. Summation layer: This layer functions like the soma of neurons. It totals the input signal got. Activation layer: This layer is additionally like a soma, and it takes the collected data and flames a sign in particular if the accumulated input passes a specific boundary regard. Else, it doesn't fire Output layer: This layer is like axon terminals in that it very well may be associated with different neurons/networks or go about as a last output layer (for forecasts).. The above figure shows a simple Artificial Neural Network. In the earlier picture, $X_1, X_2, X_3, \dots, X_n$ are the sources of info provided to the counterfeit neural organization. $W_1, W_2, W_3, \dots, W_n$ are the loads related with the data sources, and Y is the last prediction. Numerous activation functions can be utilized in the activation layer, to change over every one of the direct subtleties created at the input and make the summation layer nonlinear. This assists clients with gaining more insights regarding the input information that would not be conceivable if this were a straight capacity. In this way, the activation layer assumes a significant part in forecasts.

Probably the most natural sorts of activation functions are sigmoid, ReLU, and softmax. Linear Dependency and Period To examine the number of arrangements the condition has, we can consider the sections of A indicating various bearings we can go from the root, and decide the number of ways there are of arriving at b . In this view, every component of x determines how far we should go in every one of these headings, with x_i indicating how far to move toward section i . This sort of activity is called as Linear Combination. Think about a bunch of vectors $\{v(1), v(2), \dots, v(n)\}$. Increase every vector $v(i)$ by a coefficient which yields the accompanying outcome: Officially, this sort of excess is known as direct reliance. A bunch of vectors is directly autonomous if no vector in the set is a straight blend of different vectors. On the off chance that we add a vector to a set that is a straight blend of different vectors in the set, the new

vector doesn't add any focuses to the set's range. This implies that for the section space of the lattice to include all of R_m , the framework should contain at any rate one bunch of m directly free segments.

Norms Now and again, we need to quantify the size of a vector. In AI, we as a rule measure the size of vectors utilizing a capacity called a norm. A norm is any function f that fulfills the subsequent characteristics

Details of Proposed Operations The general design of a Neural Network is created utilizing the model article in Keras. This gives a basic method to make a heap of layers by adding new layers in a steady progression. The least demanding approach to describe a model is by using the progressive model, which grants straightforward development of a straight store of layers. A layer in the Neural Network is described as a social occasion of neurons or a shrewdly secluded assembling in a different leveled network structure. As DL ended up being progressively notable, there were a couple of tests drove with network designs to improve execution for an arrangement of usage cases. The use cases dependent on typical oversaw counts like request and backslide, PC vision tests, expanding DL for trademark language getting ready and comprehension, talk affirmation, and mixes of different spaces. To deal with the model improvement measure, Keras gives us a couple of sorts of layers and diverse means to interface them..

An engineering chart is a graphical portrayal of a bunch of ideas, that are important for a design, including their standards, components and parts. What is an outline? What are the sorts of outlines for engineering? The Dragon1 open EA Method makes it exceptionally understood: if an outline doesn't show an idea, guideline or part of a rule, it's anything but an engineering chart, since it doesn't show (a piece of) the design. There are numerous sorts of design charts, similar to a product engineering graph, framework engineering outline, application design graph, security design graph, and so forth

For system fashioners, they need structure designing outlines to fathom, clarify, and bestow considerations in regards to the system structure and the customer necessities that the structure ought to maintain. It's an essential construction can be used at the system orchestrating stage helping assistants with understanding the plan, look at changes, and grant assumptions undeniably.

Thick Layer: A thick layer is an ordinary Neural Network layer that relates every neuron in the portrayed layer to every neuron in the past layer. For instance, if Layer 1 has 5 neurons and Layer 2 (thick layer) has 3 neurons, the hard and fast number of relationship between Layer 1 and Layer 2 would be 15 (5×3). Since it obliges each possible relationship between the layers, it is known as a "thick" layer. We similarly need to describe the info shape for the Keras layer. The info shape ought to be portrayed for simply the chief layer. Coming about layers basically need the amount of neurons described. We can use the `input_dim` attribute to describe the quantity of estimations the information has. For instance, if we have a table with 10 features and 1000 models, we need to give the `input_dim` as 10 to the layer to appreciate the condition of information data.

Experiment Results conclusion In outline, this novel approach was utilized to build up a technique and numerical structure for deciding an interstate frameworks' most critical links. There are two principle benefits to this strategy and system. The first is that it is straightforward. There is no compelling reason to learn or build up a confounded reproduction that distinguishes the thruway's most critical links.

references L. Velasco and L. Gifre, "iONE: A work process situated ABNO execution," in Proc. of the Photonics in Switching Conf., 2015 L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco, "Big data analytics in support of virtual network topology adaptability," in Optical Fiber Communication Conf. (OFC), 2016. F. Morales, M. Ruiz, and L. Velasco, "Virtual network topology reconfiguration based on big data analytics for traffic prediction," in Optical Fiber Communication Conf. (OFC), 2016. M. R. Rufuie, E. Gad, M. Nakhla, and R. Achar, "Generalized Hermite polynomial chaos for variability analysis of macromodels embedded in nonlinear circuits," IEEE Trans.

Compon. Packag. Manuf.

Technol., vol. 4, no. 4, pp. 673–684, Apr. 2014. D. Spina, D. De Jonghe, D. Deschrijver, G. Gielen, L. Knockaert, and T. Dhaene, "Stochastic macromodeling of nonlinear systems via polynomial chaos expansion and transfer function trajectories," IEEE Trans. Microw. Theory Techn., vol. 62, no. 7, pp. 1454–1460, Jul. 2014. P. Manfredi, D. Vande Ginste,

D. De Zutter, and F. G. Canavero, "Stochastic modeling of nonlinear circuits via SPICE-compatible spectral equivalents," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 7, pp. 2057– 2065, Jul. 2014..

F. Gao. "Organization traffic forecast dependent on neural organization," Proceedings of 2015 International Conference on Intelligent Transportation, Big Data and Smart City, pp. 527-530, 2016. B. S.

Mi, C. G. Melody. "Forecast models for network multi-source spread of data dependent on multivariate turbulent time arrangement," 2017 third IEEE International Conference on Computer and Communications, ICCC 2017, vol. 2018-January, pp. 767-771, 2018. C. Xue, H. B. Cao, W. W. Xu, X. Gu, J. Wang. "A transient force guaging model for photovoltaic plants dependent on information mining," 2017 third IEEE International Conference on Computer and Communications, ICCC 2017, vol. 2018- January, pp. 2873-2878, 2018. H. Shao, B. H. Soong. "Traffic stream forecast with Long Short Term Memory organizations (KNNs)," IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 2986-2989, 2017. Q. Zhuo. Z, Q. M. Li, H. Yan, Y. Qi. "long transient memory neural organization for network traffic forecast," procedures of the 2017 twelfth International Conference on Intelligent Systems and Knowledge Engineering, vol. 2018-January, pp. 1-6, 2018.

Report Title:	Plagrism report identify critical links
Report Link: (Use this link to send report to anyone)	https://www.check-plagiarism.com/plag-report/23493218c3b1c58c6e07eb4792346a415f95e1617804872
Report Generated Date:	07 April, 2021
Total Words:	1916

Total Characters:	12172
Keywords/Total Words Ratio:	81.06%
Excluded URL:	No
Unique:	88%
Matched:	12%

Sentence wise detail:

an unified reliable methodology for high speed link analysis using vector quantization a deep learning approach k. janaki rami reddy1 k. satya srinivas2 s. l.

jany shabu3 1 student sathyabama institute of science and technology chennai 2 student sathyabama institute of science and technology chennai

3 assistant professor sathyabama institute of science and technology chennai abstract for any kinds of network a vital execution constraint is its connection network. (0)

studying the energy restriction and the vibrant ecosystem engineers typically needs to maintain the network being completely linked maximum possible time so as to guarantee a greatest potential coverage.

be that as it may most methodologies in the current writing zeroed in on connect utilization in network advancement.

the vital attributes of metropolitan transportation frameworks were not thought of and the idea of criticality of connections was stirred up with the idea of weakness.

by recognizing the basic parts of the conveyance network fitting methodologies can be created to improve unwavering quality remedial

activity a severe fix methodology for basic focuses and so forth keywords deep learning input output layer optimization. a general system wireless sensor networks are made out of little and cheap sensor hubs that have obliged assets.

such networks are required to be utilized in an assortment of uses.

in certain applications the network is conveyed in a threatening climate and sensors might be under various dangers.

contrasting and other potential dangers sending a wormhole assault is simpler and may make more noteworthy harm the network.

in this manner how to distinguish the presence of wormholes turns into an indispensable issue for sensor networks.

software engineering starts with a need dependent on a genuine or saw inadequacy where this converts into characterizing the issue.

installed inside this cycle is the criticism circle.

it keeps experts from building up an unnecessary arrangement on the grounds that

the criticism circle compels them to consistently screen the cycle to guarantee they

address the client s issue. this paper characterizes the criticality of connections

from two points of view i. e.

the weakness and potential in light of which a novel technique for recognizing basic connections in a metropolitan transportation network is proposed.

this tale procedure incorporates a positioning technique and a novel mesoscopic model to analyse

the metropolitan transportation network execution.

profound learning is a specific subfield of artificial intelligence another translation of taking in depictions from data that puts an emphasis on learning reformist layers of continuously critical depictions. the some place down in significant learning isn't a reference to such a more significant

course of action achieved by the approach rather it addresses this considered reformist layers of depictions the number of layers add to a model of the information is known as the profundity of the model. (1)

other fitting names for the field might have been layered portrayals learning and various levelled portrayals learning.

present day profound adapting frequently includes tens or even many progressive layers of portrayals and they're totally gained consequently from openness to preparing information. (2)

in the meantime different ways to deal with ai will in general zero in on learning just a couple of layers of portrayals of the information henceforth they're some of the time called shallow learning.

neural network artificial neural network an artificial neural network attempts to mirror the mind at its most essential level

i. e.
that of the neuron.

an artificial neuron has a comparative construction to that of a human neuron and contains the accompanying areas info layer this layer resembles dendrites and takes contribution from various organizations neurons.

summation layer this layer functions like the soma of neurons.

it totals the input signal got.

activation layer this layer is additionally like a soma and it takes the collected data and flames a sign in particular if the accumulated input passes a specific boundary regard. (3) else it doesn't fire output layer this layer is like axon terminals in that it very well

may be associated with different neurons networks or go about as a last output layer for forecasts .. the above figure shows a simple artificial neural network.

in the earlier picture x_1 x_2 x_3 x_n are the sources of info provided to the counterfeit neural organization. w_1 w_2 w_3 w_n are the loads related with the data sources and y is the last prediction.

cnumerous activation functions can be utilized in the activation layer to change over every one of the direct subtleties created at the input and make the summation layer nonlinear.

this assists clients with gaining more insights regarding the input information that would not be conceivable if this were a straight capacity.

in this way the activation layer assumes a significant part in forecasts.
probably the most natural sorts of activation functions are sigmoid relu and softmax.
linear dependency and period to examine the number of arrangements the condition has we can consider the sections of an

indicating various bearings we can go from the root and decide the number of ways there are of arriving at b.

in this view every component of x determines how far we should go in every one of these headings with x_i indicating how far to move toward section i .

this sort of activity is called as linear combination. think about a bunch of vectors

v_1 v_2 v_n .

increase every vector v_i by a coefficient which yields the accompanying outcome officially this sort of excess is known as direct reliance.

a bunch of vectors is directly autonomous if no vector in the set is a straight blend of different vectors. on the off chance that we add a vector to a set that is a straight blend of different vectors in the set the new vector doesn't add any focuses to the set's range.

this implies that for the section space of the lattice to include all of $\mathbb{R}^m$ the framework should contain at any rate one bunch of m directly free segments. norms now and again we need to quantify the size of a vector. (4)

in ai we as a rule measure the size of vectors utilizing a capacity called a norm
. a norm is any function f that fulfills the subsequent characteristics details of

proposed operations the general design of a neural network is created

utilizing the model article in keras.

this gives a basic method to make a heap of layers by adding new layers in a steady progression. the least demanding approach to . (5)

describe a model is by using the progressive model which grants straightforward development of a straight store of layers.

a layer in the neural network is described as a social occasion of neurons or a shrewdly secluded assembling in a different leveled network structure.

as dl ended up being progressively notable there were a couple of tests drove with network designs to improve execution for an arrangement of usage cases.

the use cases dependent on typical oversaw counts like request and backslide pc vision tests expanding dl for trademark language getting ready and comprehension talk affirmation and mixes of different spaces.

to deal with the model improvement measure keras gives us a couple of sorts of layers and diverse means to interface them..

an engineering chart is a graphical portrayal of a bunch of ideas that are important for a design including their standards components and parts.

what is an outline what are the sorts of outlines for engineering the dragon1

open ea method makes it exceptionally understood if an outline doesn't show

an idea guideline or part of a rule it's anything but an engineering chart since it doesn't show a piece of the design. (6)

there are numerous sorts of design charts similar to a product engineering graph framework engineering outline application

design graph security design graph and so forth for system fashioners they need structure designing outlines to fathom

clarify and bestow considerations in regards to the system structure and the customer necessities that the structure ought to maintain. (7)

it's an essential construction can be used at the system orchestrating stage helping assistants with understanding the plan look at changes and grant assumptions undeniably. (8) thick layer a thick layer is an ordinary neural network layer that relates every neuron in the portrayed layer to every neuron in the past layer.

for instance if layer 1 has 5 neurons and layer 2 thick layer has 3 neurons the hard and fast number of relationship between layer 1 and layer 2 would be 15 5 3 .

since it obliges each possible relationship between the layers it is known as

a thick layer. we similarly need to describe the info shape for the keras layer.

the info shape ought to be portrayed for simply the chief layer.

coming about layers basically need the amount of neurons

described.

we can use the input dim attribute to describe the quantity of estimations the information has.

for instance if we have a table with 10 features and 1000 models we need to give the input dim as 10 to the layer to appreciate the condition of information data.

experiment results conclusion in outline this novel approach was utilized to build up a technique and numerical structure for deciding an interstate frameworks most critical

links. (9) there are two principle benefits to this strategy and system.

the first is that it is straightforward.

there is no compelling reason to learn or build up a confounded reproduction that distinguishes the thruway's most critical links. references l. velasco and l.

gífrone a work process situated above execution in proc.

of the photonics in switching conf. 2015 l. gífrone l. m. contreras v. lopez and l.

velasco big data analytics in support of virtual network topology adaptability in optical fiber communication conf. ofc 2016. f. morales m. ruiz and l. velasco virtual network topology reconfiguration based on big data analytics for traffic prediction (10)

velasco big data analytics in support of virtual network topology adaptability in optical fiber communication conf. ofc 2016. f. morales m. ruiz and l. in optical fiber communication conf. ofc 2016.

m. r. rufuie e. gad m. nakhla (10)

velasco big data analytics in support of virtual network topology adaptability in optical fiber communication conf. ofc 2016. f. morales m. ruiz and l. and r. (12)

achar generalized hermite polynomial chaos for variability analysis of macromodels embedded in nonlinear circuits *ieee trans. compon. packag. manuf. technol.* vol. 4 no. 4 pp. 673 684 apr. 2014. d. spina d. de jonghe d. beschrijver g. gielen l. knockaert and t. (13)

dhaene stochastic macromodeling of nonlinear systems via polynomial chaos expansion and transfer function trajectories *ieee trans.*

microw. theory techn. vol. 62 no. 7 pp. 1454 1460 jul. 2014. p. manfredi d. vande ginste d. de zutter and f. g. (14)

microw. theory techn. vol. 62 no. 7 pp. 1454 1460 jul. 2014. p. manfredi d. vande ginste canavero stochastic modeling of nonlinear circuits via spice compatible spectral equivalents *ieee trans. circuits syst. i reg. papers* vol. 61 no. 7 pp. (15)

2057 2065 jul. 2014.. f. gao.

organization traffic forecast dependent on neural organization proceedings of 2015 international conference on intelligent transportation big data and smart

city pp. 527 530 2016.

b. s. mi c. g. melody.

forecast models for network multi source spread of data dependent on multivariate turbulent time arrangement 2017 third ieee international conference on computer and communications iccc 2017 vol. 2018 january pp. 767 771 2018. c. xue h. b. cao w. w. xu x. gu j. wang. a transient force guaging model for photovoltaic plants dependent on information mining 2017 third ieee international conference on computer and communications iccc 2017 vol. 2018 january pp. 2873 2878 2018. h. shao b. h. soong. traffic stream forecast with long short term memory organizations knns ieee region 10 annual international conference proceedings tencon pp. 2986 2989 2017. q. zhuo. z q. m. li h. yan y. qi.

long transient memory neural organization for network traffic forecast procedures of the 2017 twelfth international conference on intelligent systems and knowledge engineering vol. 2018 january pp. 1 6 2018. (15)

Match Urls:

- 0: <https://digital.hbs.edu/platform-digit/submission/upwork-can-value-creation-spur-value-capture/>
- 1: <https://www.merriam-webster.com/dictionary/model>
- 2: <https://www.merriam-webster.com/dictionary/information>
- 3: <https://www.grammarly.com/blog/regards/>
- 4: <http://fourier.eng.hmc.edu/e161/lectures/algebra/node11.html>
- 5: <https://infinityworldnews.com/latest-tech-news/2021/03/11/qled-tv-versus-neo-qled-how-samsung-tvs-are-evolving/>
- 6: <https://99designs.com/>
- 7: <https://www.merriam-webster.com/dictionary/maintain>
- 8: <https://www.merriam-webster.com/thesaurus/undeniably>
- 9: <https://linksinc.org/members/>
- 10: <https://www.osapublishing.org/abstract.cfm?URI=ofc-2016-Th3I.5>
- 11: <https://www.jrcigars.com/>
- 12: <https://biblio.ugent.be/publication/5730955>
- 13: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.717.9650>
- 14: [https://www.scribd.com/document/360998805/Stochastic-Modeling-of-Nonlinear-Circuits-via-SPI CE- Compatible-Spectral-E equivalents](https://www.scribd.com/document/360998805/Stochastic-Modeling-of-Nonlinear-Circuits-via-SPI-CE- Compatible-Spectral-E equivalents)
- 15: https://www.researchgate.net/publication/338366086_Comparative_analysis_on_Prediction_Models_with_various_Data_Processings_in_the_Prognosis_of_Cervical_Cancer

Keywords Density		
One Word	2 Words	3 Words
layer 3.81%	neural network 0.67%	science technology chennai 0.29%
work 2.95%	international conference 0.48%	vol 2018 january 0.29%
network 2.48%	thick layer 0.38%	sathyabama institute science 0.29%
sign 1.24%	layer layer 0.38%	communications iccc 2017 0.19%
model 1.24%	institute science 0.29%	student sathyabama institute 0.19%

Plagiarism Report

By check-plagiarism.com

B.JOURNAL PAPER

An Unified & Reliable Methodology for High Speed Link Analysis using Vector Quantization

*A Deep Learning Approach

K. Satya Srinivas¹, K. Janaki Rami Reddy², S. L. Jany Shabu³

¹ Student, Sathyabama Institute of Science and Technology, Chennai

² Student, Sathyabama Institute of Science and Technology, Chennai

³ Assistant Professor, Sathyabama Institute of Science and Technology, Chennai

Abstract— For any kinds of network, a vital execution constraint is its connection network. Studying the energy restriction and the vibrant ecosystem, engineers typically needs to maintain the network being completely linked maximum possible time, so as to guarantee a greatest potential coverage. Be that as it may, most methodologies in the current writing zeroed in on connect utilization in network advancement. The vital attributes of metropolitan transportation frameworks were not thought of and the idea of criticality of connections was stirred up with the idea of weakness. By recognizing the basic parts of the conveyance network, fitting methodologies can be created to improve unwavering quality, remedial activity, a severe fix methodology for basic focuses, and so forth

Keywords— Deep Learning, Input Output Layer, Optimization.

into characterizing the issue. Installed inside this cycle is the criticism circle. It keeps experts from building up an unnecessary arrangement on the grounds that the criticism circle compels them to consistently screen the cycle to guarantee they address the client's issue.

This paper characterizes the criticality of connections from two points of view, i.e., the weakness and potential, in light of which a novel technique for recognizing basic connections in a metropolitan transportation network is proposed. This tale procedure incorporates a positioning technique and a novel mesoscopic model to analyse the metropolitan transportation network execution.

1.INTRODUCTION

A GENERAL SYSTEM

Wireless sensor networks are made out of little and cheap sensor hubs that have obliged assets. Such networks are required to be utilized in an assortment of uses. In certain applications, the network is conveyed in a threatening climate, and sensors might be under various dangers. Contrasting and other potential dangers, sending a wormhole assault is simpler and may make more noteworthy harm the network. In this manner, how to distinguish the presence of wormholes turns into an indispensable issue for sensor networks.

Software Engineering starts with a need dependent on a genuine or saw inadequacy where this converts

Profound learning is a specific subfield of Artificial Intelligence another translation of taking in depictions from data that puts an emphasis on learning reformist layers of continuously critical depictions. The some place down in significant learning isn't a reference to such a more significant course of action achieved by the approach; rather, it addresses this considered reformist layers of

depictions

The number of layers add to a model of the information is known as the profundity of the model. Other fitting names for the field might have been layered portrayals learning and various levelled portrayals learning. Present day profound adapting frequently includes tens or even many progressive layers of portrayals—and they're totally gained consequently from openness to preparing information. In the meantime, different ways to deal with AI will in general zero in on learning just a couple of layers of portrayals of the information; henceforth, they're some of the time called shallow learning.

Neural Network

Artificial Neural Network

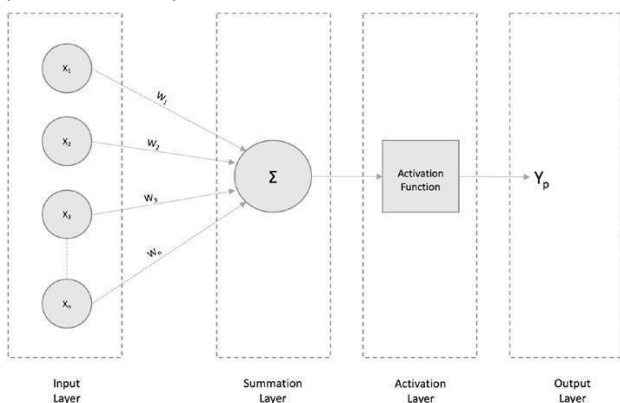
An artificial neural network attempts to mirror the mind at its most essential level, i.e., that of the neuron. An artificial neuron has a comparative construction to that of a human neuron and contains the accompanying areas:

Info layer: This layer resembles dendrites and takes contribution from various organizations/neurons.

Summation layer: This layer functions like the soma of neurons. It totals the input signal got.

Activation layer: This layer is additionally like a soma, and it takes the collected data and flames a sign in particular if the accumulated input passes a specific boundary regard. Else, it doesn't fire

Output layer: This layer is like axon terminals in that it very well may be associated with different neurons/networks or go about as a last output layer (for forecasts)..



The above figure shows a simple Artificial Neural Network.

In the earlier picture, $x_1, x_2, x_3, \dots, x_n$ are the sources of info provided to the counterfeit neural organization. $w_1, w_2, w_3, \dots, w_n$ are the loads related with the data sources, and y is the last prediction.

Numerous activation functions can be utilized in the activation layer, to change over every one of the direct subtleties created at the input and make the summation layer nonlinear. This assists clients with gaining more insights regarding the input information that would not be conceivable if this were a straight capacity. In this way, the activation layer assumes a significant part in forecasts. Probably the most natural sorts of activation functions are sigmoid, ReLU, and softmax.

Linear Dependency and Period

To examine the number of arrangements the condition has, we can consider the sections of An indicating various bearings we can go from the root, and decide the number of ways there are of arriving at b . In this view, every component of x determines how far we should go in every one of these headings, with x_i indicating how far to move toward section i .

$$Ax = \sum_i x_i A_{:,i}.$$

This sort of activity is called as Linear Combination. Think about a bunch of vectors $\{v(1), v(2), \dots, v(n)\}$. Increase every vector $v(i)$ by a coefficient which yields the accompanying outcome:

$$\sum_i c_i v^{(i)}.$$

Officially, this sort of excess is known as direct reliance. A bunch of vectors is directly autonomous if no vector in the set is a straight blend of different vectors. On the off chance that we add a vector to a set that is a straight blend of different vectors in the set, the new vector doesn't add any focuses to the set's range. This implies that for the section space of the lattice to include all of R_m , the framework should contain at any rate one bunch of m directly free segments.

Norms

Now and again, we need to quantify the size of a vector. In AI, we as a rule measure the size of vectors utilizing a capacity called a norm .

$$||x||_p = \left(\sum_i |x_i|^p \right)^{\frac{1}{p}}$$

A norm is any function f that fulfills the subsequent characteristics

$$f(x) = 0 \Rightarrow x = 0$$

$$f(x + y) \leq f(x) + f(y) \text{ (the triangle inequality)}$$

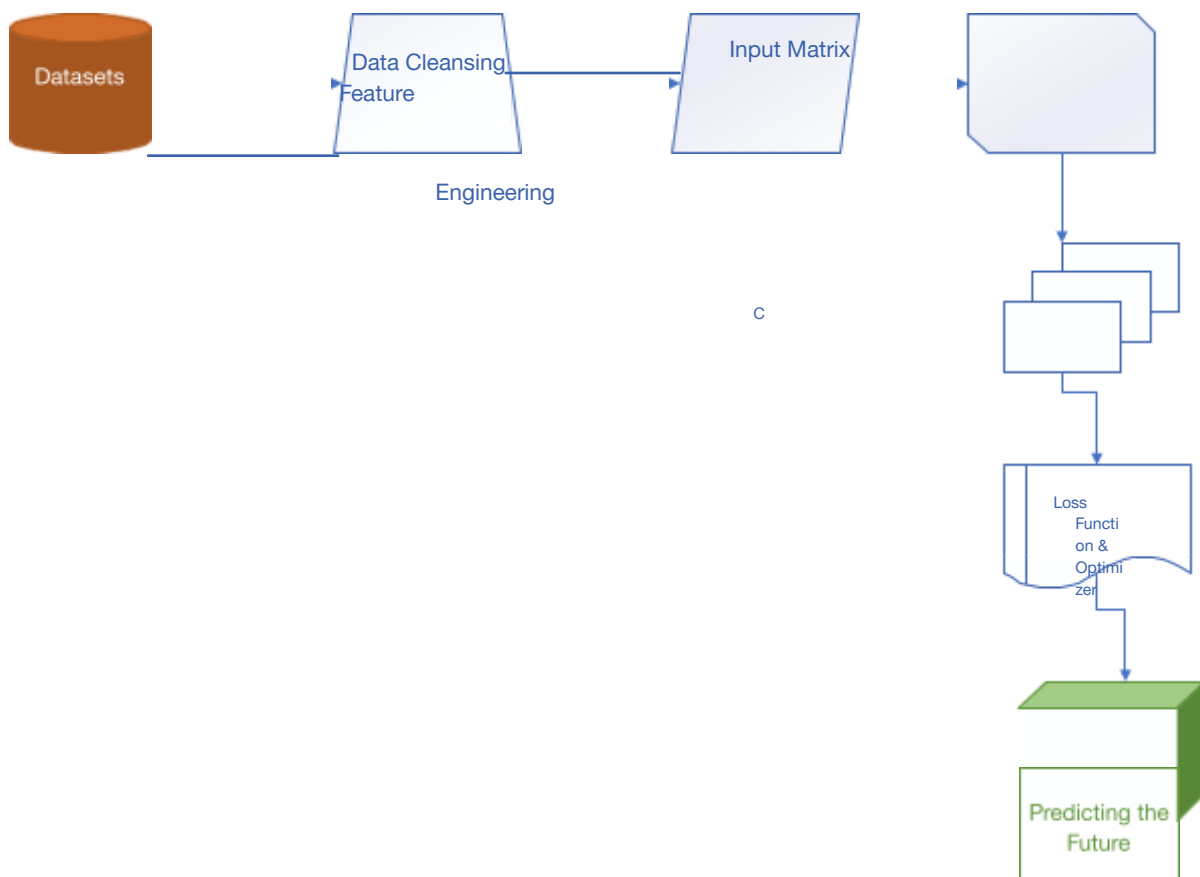
$$\forall \alpha \in \mathbb{R}, f(\alpha x) = |\alpha|f(x)$$

Details of Proposed Operations

The general design of a Neural Network is created utilizing the model article in Keras. This gives a basic method to make a heap of layers by adding new

layers in a steady progression. The least demanding approach to .describe a model is by using the progressive model, which grants straightforward development of a straight store of layers.

A layer in the Neural Network is described as a social occasion of neurons or a shrewdly secluded assembling in a different leveled network structure. As DL ended up being progressively notable, there were a couple of tests drove with network designs to improve execution for an arrangement of usage cases. The use cases dependent on typical oversaw counts like request and backslide, PC vision tests, expanding DL for trademark language getting ready and comprehension, talk affirmation, and mixes of different spaces. To deal with the model improvement measure, Keras gives us a couple of sorts of layers and diverse means to interface them..



An engineering chart is a graphical portrayal of a bunch of ideas, that are important for a design, including their standards, components and parts.

What is an outline? What are the sorts of outlines for engineering? The Dragon1 open EA Method makes it exceptionally understood: if an outline doesn't show an idea, guideline or part of a rule, it's anything but an engineering chart, since it doesn't show (a piece of) the design.

There are numerous sorts of design charts, similar to a product engineering graph, framework engineering outline, application design graph, security design graph, and so forth

For system fashioners, they need structure designing outlines to fathom, clarify, and bestow considerations in regards to the system structure and the customer necessities that the structure ought to maintain. It's an essential construction can be used at the system orchestrating stage helping assistants with understanding the plan, look at changes, and grant assumptions undeniably.

Thick Layer:

A thick layer is an ordinary Neural Network layer that relates every neuron in the portrayed layer to every neuron in the past layer. For instance, if Layer 1 has 5 neurons and Layer 2 (thick layer) has 3 neurons, the hard and fast number of relationship between Layer 1 and Layer 2 would be 15 (5×3). Since it obliges each possible relationship between the layers, it is known as a "thick" layer. We similarly need to describe the info shape for the Keras layer. The info shape ought to be portrayed for simply the chief layer. Coming about layers basically need the amount of neurons described. We can use the input_dim attribute to describe the quantity of estimations the information has. For instance, if we have a table with 10 features and 1000 models, we need to give the input_dim as 10 to the layer to appreciate the condition of information data.

Experiment Results



	precision	recall	f1-score	support
-1	0.88	1.00	0.93	7
0	1.00	0.80	0.89	10
1	0.92	1.00	0.96	11
accuracy			0.93	28
macro avg	0.93	0.93	0.93	28
weighted avg	0.94	0.93	0.93	28

Conclusion

In outline, this novel approach was utilized to build up a technique and numerical structure for deciding an interstate frameworks' most critical links. There are two principle benefits to this

strategy and system. The first is that it is straightforward. There is no compelling reason to learn or build up a confounded reproduction that distinguishes the thruway's most critical links.

References

- [1] L. Velasco and L. Gifre, "iONE: A work process situated ABNO execution," in Proc. of the Photonics in Switching Conf., 2015
- [2] L. Gifre, L. M. Contreras, V. Lopez, and L. Velasco, "Big data analytics in support of virtual network topology adaptability," in Optical Fiber Communication Conf. (OFC), 2016.
- [3] F. Morales, M. Ruiz, and L. Velasco, "Virtual network topology reconfiguration based on big data analytics for traffic prediction," in Optical Fiber Communication Conf. (OFC), 2016.
- [4] M. R. Rufuie, E. Gad, M. Nakhla, and R. Achar, "Generalized Hermite polynomial chaos for variability analysis of macromodels embedded in nonlinear circuits," IEEE Trans. Compon. Packag. Manuf. Technol., vol. 4, no. 4, pp. 673–684, Apr. 2014.
- [5] D. Spina, D. De Jonghe, D. Deschrijver, G. Gielen, L. Knockaert, and T. Dhaene, "Stochastic macromodeling of nonlinear systems via polynomial chaos expansion and transfer function trajectories," IEEE Trans. Microw. Theory Techn., vol. 62, no. 7, pp. 1454–1460, Jul. 2014.
- [6] P. Manfredi, D. Vande Ginste, D. De Zutter, and F. G. Canavero, "Stochastic modeling of nonlinear circuits via SPICE-compatible spectral equivalents," IEEE Trans. Circuits Syst. I, Reg. Papers, vol. 61, no. 7, pp. 2057–2065, Jul. 2014.
- [7] C. Xue, H. B. Cao, W. W. Xu, X. Gu, J. Wang, "A short-term power forecasting model for photovoltaic plants based on data mining," 2017 3rd IEEE International Conference on Computer and Communications, ICC3 2017, vol. 2018-January, pp. 2873-2878, 2018.
- [8] K. Zhang, Y. Chai, X. A. Fu, "A network traffic prediction model based on recurrent wavelet neural network," Proceedings of 2 nd International Conference on Computer Science and Network Technology, pp. 1630-1633, 2012.
- [9] X. Wang, C. Zhang, S. Zhang, "Modified Elman neural network and its application to network traffic prediction," Proceedings of 2012 IEEE 2nd International Conference on Cloud Computing and Intelligence Systems, vol. 2, pp. 629-633, 2013.
- [10] Y. Tian, L. Pan, "Predicting short-term traffic flow by Long ShortTerm Memory recurrent neural network," Proceedings - 2015 IEEE International Conference on Smart City, Held Jointly with 8th IEEE International Conference on Social Computing and Networking, 5th IEEE International Conference on Sustainable Computing and Communications, pp. 153-158, 2015.
- [11] H. Shao, B. H. Soong, "Traffic flow prediction with Long Short-Term Memory networks (KNNs)," IEEE Region 10 Annual International Conference, Proceedings/TENCON, pp. 2986-2989, 2017.
- [12] Q. Zhuo, Z. Q. M. Li, H. Yan, Y. Qi, "long short-term memory neural network for network traffic prediction," proceedings of the 2017 12th International Conference on Intelligent Systems and Knowledge Engineering, vol. 2018-January, pp. 1-6, 2018.

C.SOURCE CODE

```
import numpy as np
import pandas as
pd
import matplotlib.pyplot as
plt import seaborn as sns
from datetime import date
import warnings
warnings.filterwarnings('ignore'
)
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import classification_report
from sklearn.linear_model import
LogisticRegression from sklearn.metrics import
classification_report from sklearn.neighbors import
KNeighborsClassifier sns.set_style('darkgrid')
data =
pd.read_csv('Dataset.csv')
data.head()
data.info()
print("The shape of the data is
:",data.shape) data.describe()
plt.figure(figsize=(12,7))
f = sns.heatmap(data.isnull(), cbar = False, cmap =
'viridis')
f.set_xticklabels(labels=f.get_xticklabels(),
fontdict={'fontsize':13})
plt.title("Heatmap of Missing Values", fontsize =
15) plt.show()
```

```
data['tripping_reason'].value_counts()
data['tripping_reason'].fillna(value='transient fault',
inplace
= True)
data['other_circuit'].value_counts()
data.drop('other_line_status', inplace = True, axis =1)
```

```

data.drop('observation', inplace = True, axis =1)
data.drop('other_circuit', inplace = True, axis =1)
data['repair_carried'].value_counts()
data['repair_carried'].fillna(value = 'nil', inplace= True)
data['trip_year'] =
pd.to_datetime(data['date_of_trip'], dayfirst= True
).dt.year
data['restore_year'] = pd.to_datetime(data['date
of restoration'], dayfirst= True ).dt.year
data['trip_month'] =
pd.to_datetime(data['date_of_trip'], dayfirst= True
).dt.month
data['restore_month'] = pd.to_datetime(data['date
of restoration'], dayfirst= True ).dt.month
data['trip_hour']=pd.to_datetime(data['time_of_trip']).dt.h
o ur
data['restore_hour']=pd.to_datetime(data['time_of_restor
ation']).dt.hour
data.head(10)
data['trip_month']=data['trip_month'].map({1:'January',
2:'February', 3: 'March', 4: 'April', 5: 'May', 6: 'June', 7:
'July', 8: 'August', 9: 'September', 10: 'October', 11:
'November', 12: 'December'})
data['restore_month']=data['restore_month'].map({1:'Jan
u ary', 2:'February', 3: 'March', 4: 'April', 5: 'May', 6:
'June', 7:
'July', 8: 'August', 9: 'September', 10: 'October', 11:
'November', 12: 'December'})
data['weekday'] =
pd.to_datetime(data['date_of_trip']).dt.weekday
data['weekday'] = data['weekday'].map({0: 'Monday', 1:
'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5:
'Saturday', 6: 'Sunday'})
data['weekday'] = data['weekday'].map({0: 'Monday', 1:
'Tuesday', 2: 'Wednesday', 3: 'Thursday', 4: 'Friday', 5:
'Saturday', 6: 'Sunday'})
plt.figure(figsize= (12,7))
plt.title('Trips on Weekdays',fontsize = 15)
f = sns.barplot(x = temp['index'], y = temp['weekday'],

```

```
palette = 'hls')
f.set_xticklabels(labels          =
f.get_xticklabels(), fontdict={'fontsize':13})
plt.xlabel('Weekdays', fontsize = 16)
plt.yticks(list(range(max(temp['weekday']))))
plt.show()
```

```
temp = data['trip_year'].value_counts()
temp =
data['trip_month'].value_counts()
```

```
plt.figure(figsize= (12,7)) plt.title('Trips
in Months',fontsize = 15)
f = sns.barplot(x = temp.index[:-1], y = temp.values[:-1],
palette = 'rainbow')
f.set_xticklabels(labels =
f.get_xticklabels(), fontdict={'fontsize':13})
plt.xlabel('Months', fontsize =
16)
plt.yticks(list(range(max(temp))))
plt.show()
temp = data['weather'].value_counts()
```

```
plt.figure(figsize= (12,7))
plt.title('Trips in Weather',fontsize = 15)
f = sns.barplot(x = temp.index, y = temp.values, palette
= 'inferno')
f.set_xticklabels(labels =
f.get_xticklabels(), fontdict={'fontsize':13})
plt.xlabel('Weather', fontsize = 16)
plt.yticks(list(range(0,max(temp)+2)))
plt.show()
temp = data['line_trip'].value_counts().reset_index()
```

```
plt.figure(figsize=(9,9))
f = plt.pie(x = temp['line_trip'],labels =
['Yes','No'], colors=('lightblue','orange'), autopct=
"%1.1f%%") plt.title('Line Trips at other End', fontsize =
15)
plt.show()
temp = data['tripping_reason'].value_counts()
```

```
plt.figure(figsize= (12,7))
plt.title('Trips Reasons',fontsize =
15)
f = sns.barplot(x = temp.index, y = temp.values, palette
= 'autumn')
f.set_xticklabels(labels =
f.get_xticklabels(), fontdict={'fontsize':13}, rotation = 30)
plt.xlabel('Reasons', fontsize =
16) plt.yticks(range(max(temp)+2))
plt.show()
```

```
plt.figure(figsize= (15,10))
```

```
plt.subplot(3,3,1)
```

```
sns.distplot(data['voltage'], rug = True, kde =
```

```
False) plt.xlabel('Voltage in KiloVolts', fontsize =
```

```
12) plt.title('Distribution of Voltage',fontsize = 15)
```

```
plt.subplot(3,3,2)
sns.distplot(data['load_of_line'], color= 'green', rug =
True, kde = False)
plt.title('Distribution of Load of Line',fontsize = 15)
plt.xlabel('Load on line in Amperes', fontsize = 12)
```

```
plt.subplot(3,3,3)
sns.distplot(data['frequency'], rug= True, color= 'orange',
kde = False)
plt.xlabel('Voltage in KiloVolts', fontsize = 12)
plt.title('Distribution of Frequency',fontsize = 15)
```

#Kde Plots

```
plt.subplot(3,3,4
)
sns.kdeplot(data['voltage'], shade = True)
plt.xlabel('Voltage in KiloVolts', fontsize =
12) plt.title('Distribution of Voltage',fontsize
= 15)
```

```
plt.subplot(3,3,5)
sns.kdeplot(data['load_of_line'], shade = True, color =
'g') plt.title('Distribution of Load of Line',fontsize = 15)
plt.xlabel('Load on line in Amperes', fontsize = 12)
```

```
plt.subplot(3,3,6)
sns.kdeplot(data['frequency'],shade= True, color
= 'Orange')
plt.title('Distribution of Frequency',fontsize = 15)
```

#Box Plots

```
plt.subplot(3,3,7
)
sns.boxplot(x = data['voltage'], orient = 'v',color= 'b',
boxprops=dict(alpha=.5))
plt.subplot(3,3,8)
sns.boxplot(x = data['load_of_line'], orient = 'v', color=
'g', boxprops=dict(alpha=.5))
plt.subplot(3,3,9)
sns.boxplot(x = data['frequency'], orient = 'v', color=
'Orange', boxprops=dict(alpha=.5))
```

```
plt.tight_layout()
```

```
plt.show()  
sns.jointplot(x = data['load_of_line'], y = data['voltage'],  
kind = 'reg', color= 'g')
```

```
plt.show()
```

```

sns.jointplot(x = data['load_of_line'], y =
data['frequency'], kind = 'reg', color= 'darkorange')
plt.show()
sns.jointplot(x = data['voltage'], y = data['frequency'], kind
= 'reg', color =
'blue') plt.show()
temp =
data['trip_hour'].value_counts()
plt.figure(figsize= (10,10))

```

```

plt.subplot(2,1,1)
sns.pointplot(x = temp.index, y = temp.values ,palette=
'Reds')
sns.pointplot(x = temp.index, y = temp.values ,join=
True, color = 'r', markers = '')
plt.title('Trips on Hours', fontsize = 15)
plt.xlabel('Trip Hours of Day', fontsize =
12) plt.ylabel('Number of Hours', fontsize
= 12) plt.yticks([0,1,2,3,4])

```

```

temp =
data['restore_hour'].value_counts()
plt.subplot(2,1,2)
sns.pointplot(x = temp.index, y = temp.values ,palette=
'Greens')
sns.pointplot(x = temp.index, y = temp.values ,join=
True, color='g', markers = '')
plt.title('Restoration on Hours', fontsize = 15)
plt.xlabel('Restore Hours of Day', fontsize =
12) plt.ylabel('Number of Hours', fontsize = 12)
plt.yticks([0,1,2,3,4])
plt.show()
for i in range(0,len(data['repair_carried'])):
if data['repair_carried'][i] == 'nil':
data['repair_carried'][i] = 'None'
temp =
data['repair_carried'].value_counts()
plt.figure(figsize= (12,7))
plt.title('Repairs Carried', fontsize = 15)
f = sns.barplot(x = temp.index, y = temp.values, palette
= 'Set1')
f.set_xticklabels(labels =
f.get_xticklabels(), fontdict={'fontsize':13})
plt.xlabel('Repair Types', fontsize = 15)
plt.yticks(range(max(temp)+2))

```

```
plt.show()
plt.figure(figsize= (9,7))
plt.title('Nature of the fault',fontsize = 15)
f = sns.countplot(data['nature'], palette= 'hls')
f.set_xticklabels(labels      =
f.get_xticklabels(),
```

```

fontdict={'fontsize':13})
plt.xlabel("Types", fontsize=
14) plt.show()
data['line_trip'] = data['line_trip'].map({'no':0, 'yes':1})
data['type_of_fault'] = data['type_of_fault'].map({'low':-1,
'medium':0, 'high': 1})
data.head()
X_full = data.iloc[:, [3,4,6]].values
y_full =
data['type_of_fault'].values sc =
StandardScaler()
sc.fit(X_full)
X_full = sc.transform(X_full)
X_train, X_test, y_train, y_test = train_test_split(X_full,
y_full, test_size = 0.25, random_state = 1)
classifier = KNeighborsClassifier()
classifier.fit(X_train, y_train)
y_test_pred = classifier.predict(X_test)
print(classification_report(y_train,
classifier.predict(X_train)))
cr = classification_report(y_test,
y_test_pred) print(cr)
t = input("Enter Trip\t")
l = float(input("Enter load of
line\t")) f = float(input("Enter
Frequency\t")) if t == 'yes':
t = 1
elif t == 'no':
t = 0
samp = np.array([[int(t), int(l), float(f)]])
samp = sc.transform(samp)
res =
classifier.predict(samp)
print("\n-----Output \n")
if res == -1:
print("High
Performance") elif res ==
0:
print("Medium
Performance") else:
print("Low Performance")
classifier = LogisticRegression(solver=
'newton-cg',multi_class= 'multinomial')
classifier.fit(X_train, y_train)

```

```
y_test_pred = classifier.predict(X_test)
print(classification_report(y_train,
classifier.predict(X_train)))
cr = classification_report(y_test,
y_test_pred) print(cr)
t = input("Enter Trip\t")
```

```

l = float(input("Enter load of
line\t")) f = float(input("Enter
Frequency\t")) if t == 'yes':
t = 1
elif t == 'no':
t = 0
samp = np.array([[int(t), int(l), float(f)]])
samp = sc.transform(samp)
res =
classifier.predict(samp)
print("\n-----Output  \n")
if res == -1:
print("High
Perform") elif res ==
0:
print("Medium
Performance") else:
print("Low Performance")

```