

A ROBUST AND DYNAMIC FAULT DETECTION AND CLASSIFICATION OF SEMI-CONDUCTORS USING LOGISTIC REGRESSION

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

by

**SUNKARA SASANKA (37110748)
SURE ANISHA (37110753)**



**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 - 2020



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 **SUNKARA SASANKA (Roll. No. 17SCS0448)** and **SURE ANISHA (Roll. No. 17SCS8095)** who carried out the project entitled "**A ROBUST AND DYNAMIC FAULT DETECTION AND CLASSIFICATION OF SEMI-CONDUCTORS USING LOGISTIC REGRESSION**" under my supervision from **August 2019** to **March 2020**.

Internal Guide

Dr. K.Mohan Prasad, M.Tech., Ph.d.,

Head of the Department

Dr.S.VIGNESHWARI, M.E., Ph.D

Dr. L. Lakshmanan, M.E., Ph.D

Submitted for Viva voce Examination held on _____

Internal Examiner

External Examiner

DECLARATION

We **SUNKARA SASANKA (Roll. No. 17SCS0448)** and **SURE ANISHA (Roll. No. 17SCS8095)** hereby declare that the Project Report entitled “**A ROBUST AND DYNAMIC FAULT DETECTION AND CLASSIFICATION OF SEMI-CONDUCTORS USING LOGISTIC REGRESSION**” is done by me under the guidance of **Dr. K.Mohan Prasad, 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 INSTITUTE OF SCIENCE AND TECHNOLOGY** 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 **Dr. K. Mohan Prasad, M.Tech., Ph.d., Assistant Professor**, 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

Fault detection and classification (FDC) has been recognized as an integral component of the advanced process control (APC) framework in the semiconductor industry, as it helps to improve overall equipment efficiency (OEE). However, some unique characteristics of semiconductor manufacturing processes have posed challenges for FDC applications, such as nonlinearity in most batch processes, and multimodal batch trajectories due to product mix. Historical data are used directly as the reference of normal process operation to determine whether a new measurement is a fault. Therefore, for processes with a large number of variables, it can be computation and storage intensive, and may be difficult for online process monitoring. To address this difficulty, we propose a fast pattern recognition based fault detection method. Based on this measurement reconfiguration, a novel fault detection algorithm is designed for MMCs semiconductor power switches. The open circuit and short circuit faults are detected based on unconformity between modules output voltage and switching signals. Simulations results confirm accurate and fast operation of the proposed algorithm in faulty module diagnosis. Implementation simplicity, no need to extra voltage measurements and multi faults detection ability are some other advantages of the proposed method.

TABLE OF CONTENTS

ABSTRACT	v
LIST OF FIGURES	viii

CHAPTER No.	TITLE	PAGE No.
1.	INTRODUCTION	1
	OVERVIEW	1
	SCOPE OF THE PROJECT	1
	DOMAIN OVERVIEW	1
	MACHINE LEARNING	2
2.	AIM AND SCOPE	7
	LITERATURE SURVERY 1	7
	LITERATURE SURVERY 2	8
	LITERATURE SURVERY 3	10
	LITERATURE SURVERY 4	11
3.	SYSTEM DESIGN & METHODOLOGY	15
	INTRODUCTION	15
	WATERFALL MODEL	16
	RAD MODEL	18
	EXISTING SYSTEM	21
	PROPOSED SYSTEM	22
	DATA SET INFORMATION	23
	REQUIREMENT SPECIFICATION	23

	SOFTWARE DESCRIPTION	24
	ABOUT PYTHON	24
	PANDAS	29
	NUMPY	33
	MATHPLOTLIB	35
	ABOUT IDE	36
4.	SOFTWARE DEVELOPMENT METHODOLOGY	41
	ARCHITECTURE DIAGRAM	41
	SYSTEM IMPLEMENTARION	42
	LIST OF MODULES	42
	SOFTWARE TESTING	47
	TEST DRIVEN DEVELOPMENT	53
5.	RESULTS AND DISCUSSION	66
6.	CONCLUSION AND FUTURE WORK	72
	REFERENCES	73
	APPENDIX	
	A. PAPER ACCEPTANCE MAIL	75
	B. PLAGIARISM REPORT	76
	C. JOURNAL PAPER	77
	D. SOURCE CODE	81

LIST OF FIGURES

FIGURE No.	FIGURE NAME	PAGE No.
	MACHINE LEARNING MODELS	3
	ML DESIGN	5
3.1.1.1	WATERFALL MODEL	16
3.1.2.1	RAD MODEL	18
3.6.1.1	PYTHON OPEN SOURCE	24
4.1.1	BLOCK DIAGRAM	42
4.3.1	UNIT TESTING	51
	INTEGRATION TESTING	57
	SYSTEM TESTING	58
	REGRESSION TESTING	64
	POWER DATA SETS	66
	INFO DATA SET	67
	KIND LINE X,Y TIME,DC_POWER	67
	WEATHER FULL_DATA	69
	IRRADIATION AMBIENT,	70
	MODULE_TEMPERATURE	
	HUE AC_POWER X,Y,	71
	IRRADIATION DC_POWER	

CHAPTER 1

INTRODUCTION

Based on this measurement reconfiguration, a novel fault detection algorithm is designed for MMCs semiconductor power switches. The open circuit and short circuit fault detected based on unconformity between modules output voltage and switching signals. Simulations results confirm accurate and fast operation of the proposed algorithm in faulty module diagnosis. Implementation simplicity, no need to extra voltage measurements and multi faults detection ability are some other advantages of the proposed method

OVERVIEW

Dynamic Voltage and Frequency Scaling (DVFS): As its name states, DVFS refers to the ability of dynamically scaling the voltage and/or frequency of cores. The maximum frequency at which a core can be stably executed is limited by its supply voltage, where higher frequencies can be stably achieved at higher voltages. As shown in the relationship between the supply voltage of a core and the maximum frequency for stable execution can be modelled

SCOPE OF THE PROJECT

The main contributions of this project therefore are:

1. Data Analysis
2. Dataset Pre-processing
3. Training the Model
4. Testing of Dataset

DOMAIN OVERVIEW

Machine Learning is the most popular technique of predicting the future or classifying information to help people in making necessary decisions. Machine Learning algorithms are trained over instances or examples through which they learn from past experiences and also analyse the historical data. Therefore, as it trains over the examples, again and again, it is able to identify patterns in order to make predictions about the future. Data is the core backbone of machine learning algorithms. With the help of the historical data, we are able to create more data by training these machine learning algorithms. For example, Generative Adversarial Networks are an advanced concept of Machine Learning that learns from the historical images through which they are capable of generating more images. This is also applied towards speech and text synthesis. Therefore, Machine Learning has opened up a vast potential for data science applications.

Machine Learning

Machine Learning combines computer science, mathematics, and statistics. Statistics is essential for drawing inferences from the data. Mathematics is useful for developing machine learning models and finally, computer science is used for implementing algorithms.

However, simply building models is not enough. You must also optimize and tune the model appropriately so that it provides you with accurate results. Optimization techniques involve tuning the hyperparameters to reach an optimum result.

The world today is evolving and so are the needs and requirements of people. Furthermore, we are witnessing a fourth industrial revolution of data. In order to derive meaningful insights from this data and learn from the way in which people and the system interface with the data, we need computational algorithms that can churn the data and provide us with results that would benefit us in various ways. Machine Learning has revolutionized industries like medicine, healthcare, manufacturing, banking, and several other industries. Therefore, Machine Learning has become an essential part of modern industry. Data is expanding exponentially and in order to harness the power of this data, added by the massive increase in computation power, Machine Learning has added another dimension to the way we perceive information. Machine Learning is being utilized everywhere. The electronic devices you use, the applications that are part of your everyday life are powered by powerful machine learning algorithms.

With an exponential increase in data, there is a need for having a system that can handle this massive load of data. Machine Learning models like Deep Learning allow the vast majority of data to be handled with an accurate generation of predictions.

Machine Learning has revolutionized the way we perceive information and the various insights we can gain out of it. These machine learning algorithms use the patterns contained in the training data to perform classification and future predictions. Whenever any new input is introduced to the ML model, it applies its learned patterns over the new data to make future predictions. Based on the final accuracy, one can optimize their models using various standardized approaches. In this way, Machine Learning model learns to adapt to new examples and produce better results.

Types of Machine Learning

Machine Learning Algorithms can be classified into 3 types as follows –

1. Supervised Learning
2. Unsupervised Learning
3. Reinforcement Learning

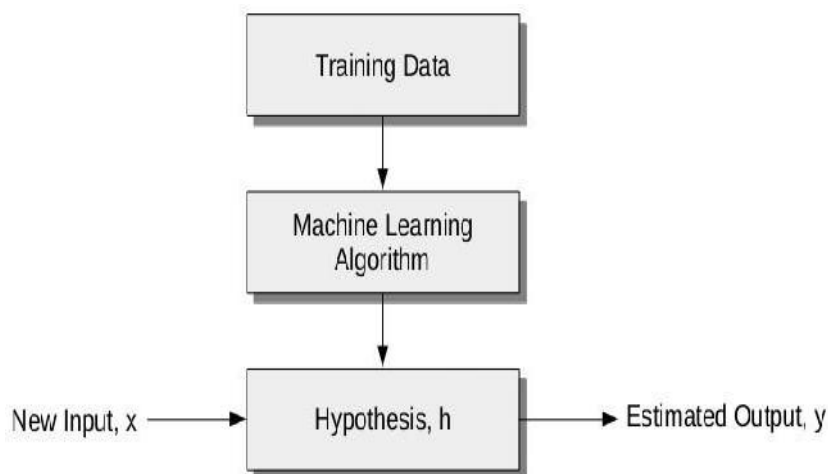


Figure 1.3.1.1 :Machine Learning Models

In the majority of supervised learning applications, the ultimate goal is to develop a finely tuned predictor function $h(x)$ (sometimes called the “hypothesis”). “Learning” consists of using sophisticated mathematical algorithms to optimize this function so that,

given input data x about a certain domain (say, square footage of a house), it will accurately predict some interesting value $h(x)$ (say, market price for said house).

$$h(x_1, x_2, x_3, x_4) = \theta_0 + \theta_1 x_1 + \theta_2 x_3^2 + \theta_3 x_3 x_4 + \theta_4 x_1^3 x_2^2 + \theta_5 x_2 x_3^4 x_4^2$$

This function takes input in four dimensions and has a variety of polynomial terms. Deriving a normal equation for this function is a significant challenge. Many modern machine learning problems take thousands or even millions of dimensions of data to build predictions using hundreds of coefficients. Predicting how an organism's genome will be expressed, or what the climate will be like in fifty years, are examples of such complex problems.

Under supervised ML, two major subcategories are:

- Regression machine learning systems: Systems where the value being predicted falls somewhere on a continuous spectrum. These systems help us with questions of “How much?” or “How many?”.
- Classification machine learning systems: Systems where we seek a yes-or-no prediction, such as “Is this tumor cancerous?”, “Does this cookie meet our quality standards?”, and so on.

In practice, x almost always represents multiple data points. So, for example, a housing price predictor might take not only square-footage (x_1) but also number of bedrooms (x_2), number of bathrooms (x_3), number of floors (x_4), year built (x_5), zip code (x_6), and so forth. Determining which inputs to use is an important part of ML design. However, for the sake of explanation, it is easiest to assume a single input value is used.

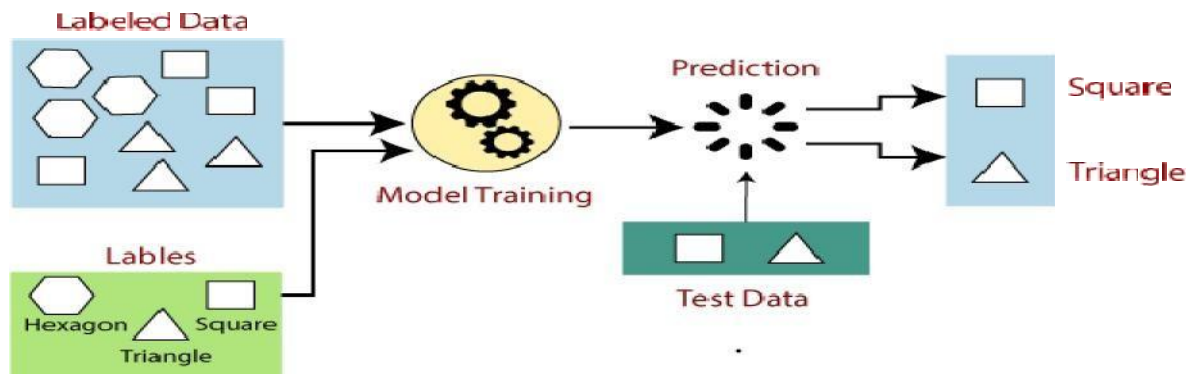


Figure 1.3.1.2 : ML Design

Steps Involved in Supervised Learning:

- First Determine the type of training dataset
- Collect/Gather the labelled training data.
- Split the training dataset into training dataset, test dataset, and validation dataset.
- Determine the input features of the training dataset, which should have enough knowledge so that the model can accurately predict the output.
- Determine the suitable algorithm for the model, such as support vector machine, decision tree, etc.
- Execute the algorithm on the training dataset. Sometimes we need validation sets as the control parameters, which are the subset of training datasets.
- Evaluate the accuracy of the model by providing the test set. If the model predicts the correct output, which means our model is accurate.

Regression:-

Regression algorithms are used if there is a relationship between the input variable and the output variable. It is used for the prediction of continuous variables, such as Weather forecasting, Market Trends, etc. Below are some popular Regression algorithms which come under supervised learning:

1. Linear Regression
2. Regression Trees

3. Non-Linear Regression
4. Bayesian Linear Regression
5. Polynomial Regression

Classification:-

Classification algorithms are used when the output variable is categorical, which means there are two classes such as Yes-No, Male-Female, True-false, etc.

- Spam Filtering,
- Random Forest
- Decision Trees
- Logistic Regression
- Support vector Machines

CHAPTER 2

AIM AND SCOPE

2.1

Literature Survey 1	
Title	Improving Time-Efficiency of Fault-Coverage Simulation for MOS Analog Circuit
Authors	Zhiqiang Liu, Shravan K. Chaganti, Degang Chen
Published Year	2018
Efficiency	• Supports class-imbalance learning techniques • Defective samples available for model training are located • Most defective patterns, even if they only occur at a low frequency
Drawbacks	• Produces a low performance with respect to model verification • No performance improvement in the sampling-based method • F-measure were insignificant
Description	The concept of directed dependency graphs (DDG) has been widely used in general and has been successfully used by Liu et al. [21] to solve log verification issues and achieve excellent results. An EDA tool was developed and implemented in Cadence virtuoso to automatically convert a circuit netlist to a DDG. An important property of the DDG is that it captures the critical dependencies between the voltages and currents of a circuit. For the purpose of decomposing a circuit to sub-circuit, this property is very useful because it allows us to know which voltages and currents are correlated. In this paper, the DDG will

	be defined in a similar manner and demonstrated through the example circuit. A node of a circuit refers to a point where two or more circuit elements meet. For two nodes to be different, their voltages must be
--	---

	different. For example, V_1 and V_o are two nodes of the circuit shown in Fig. 3(a). To avoid confusion, the term “node” will be used only when describing a circuit schematic and “vertex” will be used with reference to a graph throughout the rest of this paper. For a circuit under test(CUT), a directed dependency graph is defined as a directed graph $DDG(V, E)$, where the vertex set V represents all voltages and currents in the circuit and the edge set E stands for the directed dependencies between voltages and currents. A directed graph H is strongly connected if for every pair of vertices in H, say u and v, there is at least one path from u to v and at least one path from v to u. By convention, the trivial graph (with only one vertex) is strongly connected. A Strongly Connected Component (SCC) H of G is a subgraph of G such that H is strongly connected, but if we add any vertices or arcs to H, then it is not strongly connected anymore.
--	---

2.2

Literature Survey 2	
Title	Deep-Structured Machine Learning Model for the Recognition of Mixed-Defect Patterns in Semiconductor Fabrication Processes

Authors	Ghalia Tello, Omar Y. Al-Jarrah, Paul D. Yoo, Yousof Al-Hammadi, Sami Muhaidat, and Uihyoung Lee
Published Year	2018
Efficiency	• Instance-based models would be effective for Fault Detection • Reduce the number of misclassifications of the minority class samples • Maximize model performance

Drawbacks	• There was no sufficient amount of defective data to explain the causes of most defects • it is not possible to apply this method for the detection and isolation of multiple open-circuit faults
Description	Noise reduction is an important pre-processing step to facilitate the learning process of deep learning algorithms. A spatial filter is widely used as a noise reduction technique in image processing applications. In any ML-based model, it is important to select the correct features. Feature extraction not only reduces the complexity of the model, which is significantly influenced by the amount of input data, but also helps to prevent overfitting. The extracted features should be informative enough to assign the samples to single-defect and mixed-defect categories. We extracted 21 different features from the real portion of the dataset, representing single defect and mixed-defect samples. As observed in many real data samples, mixed-defect patterns have more defects than the single-defect patterns, so the first extracted feature is the total number of defects on the wafer. The rest of the features were extracted according to the spatial distribution of defect on the wafer surface. In order to generate these features, the number of defects in different areas of the wafer was counted.

	These features include the number of defects in borders, edges, centre and corners. Moreover, different features were generated each time with different numbers of chips. With mixed-defect patterns, a large number of individual defects is detected in the corners/borders and centre simultaneously, whereas single defects usually cover a smaller area of the wafer surface and appear either in the central region or in the corners/borders. Thus, extra features are generated by concatenating two of the earlier features, such as number of defects in the centre and borders, corners and centre and edges and centre. Splitter Rules: The splitter consists of sequence of rules that follows a tree structure in order to split and categorize data points. Initially, the IG of each feature is calculated to evaluate significance. Each rule contains a feature used to partition the data. The rules start with the feature offering the highest IG, which indicates discrimination ability, i.e., the extent to which the attribute can reduce the degree of ambiguity.
--	--

2.3

Literature Survey 3	
Title	A Convolutional Neural Network for Fault Classification and Diagnosis in Semiconductor Manufacturing Processes
Authors	Ki Bum Lee, Sejune Cheon, and Chang Ouk Kim
Published Year	2017
Efficiency	• Detect and locate an open-circuit faulty power semiconductor device • Works together with the hardware detection methods • Very fast response
Drawbacks	• Ignoring the parameter uncertainties, measurement noise, or other disturbances.

	• Less Secure, High Embedding extraction time. • Poor Application Performance
--	--

Description	A CNN is composed of two parts, as illustrated in Fig. 1. One part is responsible for feature extraction and contains an input layer, convolutional layers, and pooling layers. Convolutional layers and pooling layers, which are stacked layer by layer in the network, are feature extractors. The other part is responsible for classification and contains fully connected layers and an output layer. In this part, fully connected layers receive the features obtained by the last pooling layer as the input and perform classification tasks. The input layer receives data or images to be classified, and then, the convolutional layer detects the local features of the input and stores them as a feature map. As illustrated in Fig. 1, a connection between the input layer and the convolutional layer is established by the receptive field. A receptive field is a square matrix of weights whose size is substantially smaller than the input. A feature map contains nodes, each of which is connected to a certain input area defined by the receptive field.
-------------	---

2.4

Literature Survey 4	
Title	Performance of Machine Learning Algorithms for Class-Imbalanced Process Fault Detection Problems
Authors	Taehyung Lee, Ki Bum Lee, and Chang Ouk Kim

Published Year	2016
Efficiency	• Built-in error handling • Robust Tool for approximating a target function • Speed up the training process
Drawbacks	• Sensitive to noisy and missing data • Does not work well with high dimensionality • Does not work well with large dataset

Description	In the bagging algorithm, source samples have an equal probability of being included in a training sample group due to bootstrap sampling. Conversely, the boosting algorithm is also an iterative algorithm and increases the sampling probability in each iteration for samples that have been misclassified in a previous iteration. AdaBoost is a popular boosting algorithm [15] and was originally designed to use a DT as the weak classifier. However, the boosting algorithm also uses an SVM as a weak classifier. Table III explains the principle of AdaBoost that uses an SVM as a weak classifier. AdaBoost updates the sampling distribution for every iteration . On the first iteration, a classifier is trained with samples drawn with a uniform distribution. The sampling distribution at the next iteration is updated based on the misclassification probability , which increases the probability of the misclassified samples to be selected as training data. A new classifier is then trained with the new training data. FD-KNN: This algorithm is an extension of the standard KNN rule, which determines the k neighbours of a new sample and assigns the representative class of the neighbours as the class of the sample. The FD-KNN algorithm conducts FD only with normal samples (i.e., majority class samples). Specifically, the FD-KNN algorithm assumes that the sum of the squared distances between normal
-------------	--

	samples in a Euclidean space follows a non-central chi-square distribution. For a new sample, the FD-KNN algorithm considers the sum of the squared distances from the sample to the k neighbours to be a test statistic. The FD-KNN algorithm compares the test statistic score with a threshold (e.g., a 95% confidence limit) of the non- central chi-square distribution and classifies the sample as normal if the score is smaller than the threshold. Otherwise, the sample is classified as defective. PC-KNN: This algorithm is an extension of the FD-KNN algorithm and performs FD in a principal component space. Because the FD- KNN algorithm calculates distances to neighbouring samples in the space consisting of input variables, the calculation becomes overloaded when many variables are present in a sample. To reduce the dimensionality, the PC-KNN algorithm calculates distances to neighbours in a principal component space, which reduces both the amount of memory required to perform the KNN algorithm and the computational load. MD-KNN: The FD-KNN and PC-KNN algorithms use the Euclidean distance metric to identify k neighbours, whereas the MD-KNN algorithm uses the Mahalanobis distance metric, which considers the data density when identifying neighbours. When a new sample is introduced, the MD-KNN algorithm first analyses the data density near the sample by finding K neighbours based on the Euclidean distance and estimating the corresponding covariance matrix. Next, the algorithm identifies the k neighbours based on the Mahalanobis distance that reflects the covariance matrix.
--	--

CHAPTER 3

SYSTEM DESIGN & METHODOLOGY

INTRODUCTION

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:-

- Agile Software Development
- 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

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.

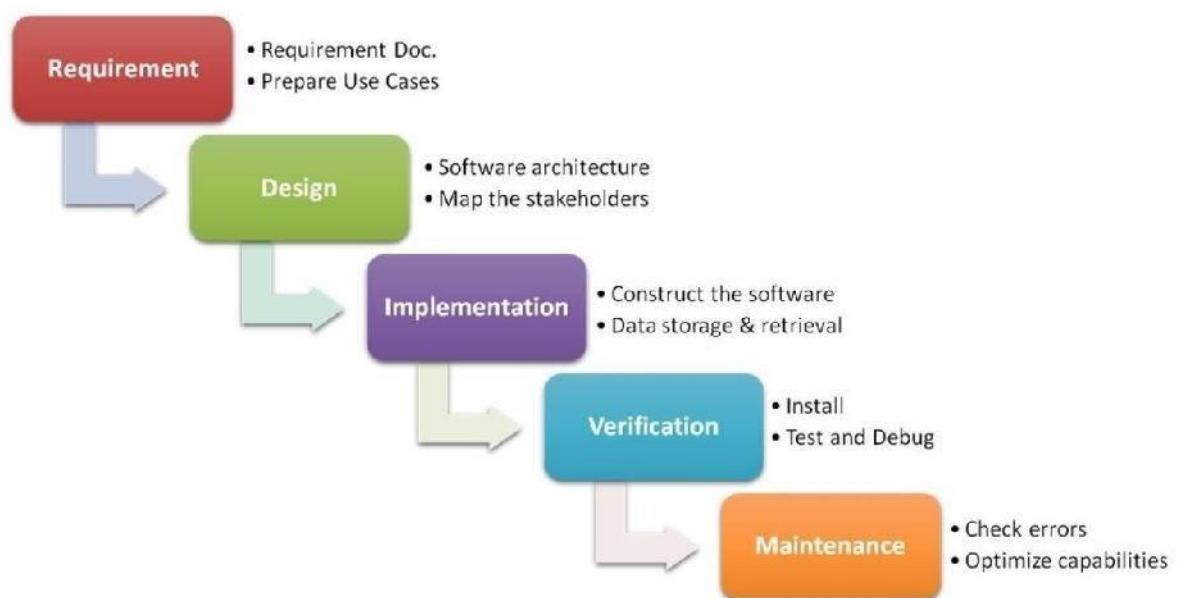


Figure 3.1.1.1 : Waterfall Model

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 re-evaluation.

5. 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.

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.

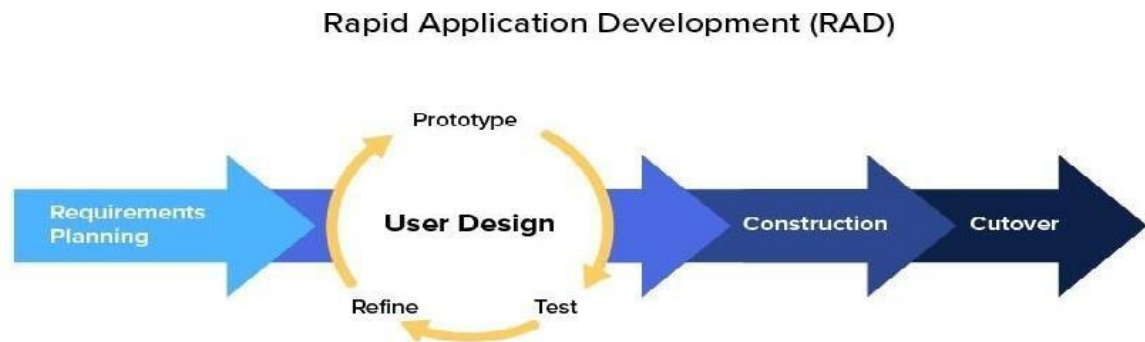


Figure 3.1.2.1 : RAD Model

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:

1. Researching the current problem.

2. Defining the requirements for the project.

3. 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:

- 1.Preparation for rapid construction.
- 2.Program and application development.
- 3.Coding.
- 4.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.

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.

EXISTING SYSTEM

The existing system presents an overview of several research efforts that propose to use machine learning techniques for power and thermal management on single-core and multi-core processors. Traditional power and thermal management techniques rely on a certain a-priori knowledge of the chip's thermal model, as well as information of the workloads/applications to be executed (e.g., transient and average power consumption). Nevertheless, these a-priori information is not always available, and even if it is, it cannot reflect the spatial and temporal uncertainties and variations that come from the environment, the hardware, or from the workloads/applications. Contrarily, techniques based on machine learning can potentially adapt to varying system conditions and workloads, learning from past events in order to improve themselves as the environment changes, resulting in improved management decisions.

Dynamic Power Management (DPM): DPM refers to the dynamic selection of the power states of individual cores. Namely, cores can be individually set to execution/active mode, or they can be individually set to some low-power mode (e.g., idle/clock-gated, sleep, power-gated, etc.). The specific low-power modes that are available depend on the chip, and every different low-power mode has an associated power consumption, as well as different latencies for transitioning across power modes.

For DPM, an undesirable situation is when a core is put to sleep only to be awakened immediately (e.g., when new service requests arrive earlier than expected), such that the system pays for the extra energy and latency of waking up the core to process the new requests. Contrarily, if a core remains idle and no service requests arrive in that period, then the core wastes energy by not entering a low-power mode.

PROPOSED SYSTEM

In the semiconductor industry, process shift and drift are primarily due to the following reasons: preventive and corrective maintenances, differences in the incoming materials, aging of the etcher over a clean cycle, and drift in the process monitoring sensors. For the etch data set, the three experiments were carried out several weeks apart. Due to the process shift and drift, data from different experiments have different means and somewhat different covariance structure.

Data pre-processing is an important aspect of multivariate statistical analysis and can have a significant impact on the overall sensitivity and robustness of the method. For batch process monitoring, data pre-processing usually includes trajectory shift, trajectory alignment, data filtering, unfolding and scaling. Other pre-processing steps may be needed depending on the specific process. Although all these data pre-processing techniques are powerful for improving the effectiveness of the fault detection method, they are not desirable in an automated manufacturing environment. This is because most of the pre-processing steps (e.g., dynamic time warping (DTW) and filtering) are process specific and require human interactions which are difficult to automate.

This is done for all training, validation and test data. Once equal length batch records are obtained, the second step is to scale each variable to zero mean and unit variance for each wafer in the training data set and scale the validation and test data accordingly using the mean and variance values obtained from the training data. Note that the above two-step data pre-processing can be done automatically in the production environment. The data is further unfolded batch-wise to obtain a two-dimensional array.

DATA SET INFORMATION

A complex modern semi-conductor manufacturing process is normally under consistent surveillance via the monitoring of signals/variables collected from sensors and or process measurement points. However, not all of these signals are equally valuable in a specific monitoring system. The measured signals contain a combination of useful information, irrelevant information as well as noise. It is often the case that useful information is buried in the latter two. Engineers typically have a much larger number of signals than are actually required. If we consider each type of signal as a feature, then feature selection may be applied to identify the most relevant signals. The Process Engineers may then use these signals to determine key factors contributing to yield excursions downstream in the process. This will enable an increase in process throughput, decreased time to learning and reduce the per unit production costs.

REQUIREMENT SPECIFICATION

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.

Hardware Specification:-

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

Software Specification:-

Backend Technologies

Python
Numpy
Sci-learn
Eclipse IDE

Frontend Technologies

Web Technologies
Bootstrap

SOFTWARE DESCRIPTION

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.

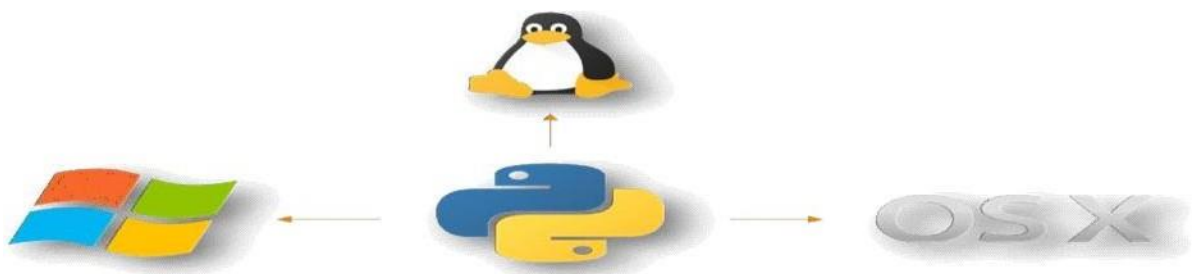


Figure 3.6.1.1 : Python Open Source

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

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:

1. Explicit Line Joining:-

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

2. Implicit Line Joining:-

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

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.

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.

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

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 Data Frame 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 Data Frame.

Data Frames 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 modelling, 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' Data Frames 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 Data Frames.

1. Series: a pandas Series is a one dimensional data structure that can store values — and for every value it holds a unique index, too.
2. Data Frame: a pandas Data Frame 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, Data Frames in Python are very similar: they come with the Pandas library, and they are defined as two-dimensional data structures with columns of potentially different types.

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

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

It provides Data Frame 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.
- Reshaping and pivoting data sets.
- Aligning data and dealing with missing data.
- Manipulating data using integrated indexing for Data Frame 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.
- Sub setting, fancy indexing, and label-based slicing data sets that are large in size.
- Generating data range, converting frequency, date shifting, lagging, and other timeseries functionality.
- Reading from files with CSV, XLSX, TXT, among other formats.
- Arranging data in an order ascending or descending.
- Filtering data around a condition.
- Analysing 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.
- 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 2dimensional 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, error charts, scatterplots, etc, with just a few lines of code."

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.

1. Komodo
2. NetBeans
3. PyCharm
4. Python for Visual Studio.NET
5. PyDev
6. PyStudio
7. 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.

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.

"notebook" or "notebook documents" denote documents that contain both code and rich text elements, such as figures, links, equations, ... Because of the mix of code and text elements, these documents are the ideal place to bring together an analysis description, and its results, as well as, they can be executed perform the data analysis in real time.

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 4

SOFTWARE DEVELOPMENT METHODOLOGY

A UML diagram is a diagram based on the UML (Unified Modelling 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 modelling 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 modelling workflows as well as a wider range of features to improve readability and efficiency.

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.

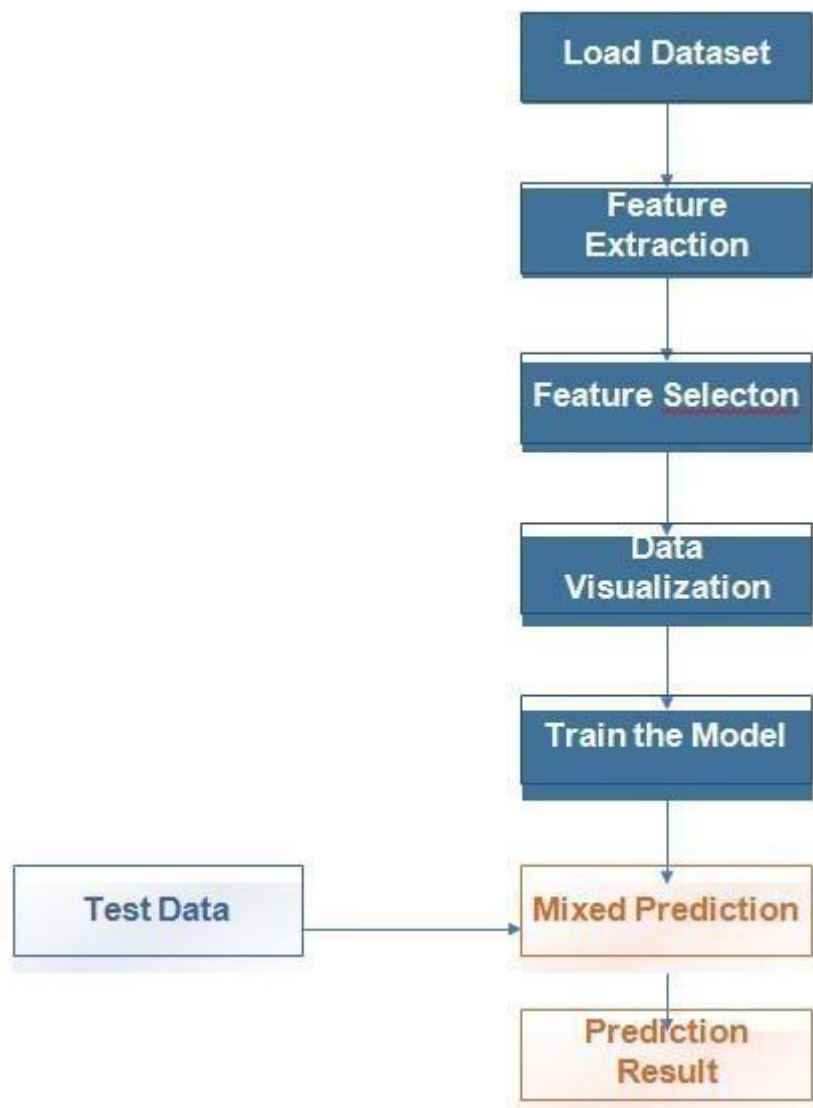


Figure 4.1.1 : Block Diagram

SYSTEM IMPLEMENTARION

List of Modules:-

Module 1 : Exploratory Data Evaluation

Module 2 : Pre-processing

Module 3 : Feature Engineering

Module 4 : Prediction

Module 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.

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 pre-processing. 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`

Module 3 : Feature Engineering:-

Filter methods are generally used as a pre-processing 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-efficients.

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

4.2.1.4. 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. This methodology is more commonly known as learning decision tree from data and above tree is called Classification tree as the target is to classify passenger as survived or died.

Regression trees are represented in the same manner, just they predict continuous values like price of a house. In general, Decision Tree algorithms are referred to as CART or Classification and Regression Trees.

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.
- 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.



Figure 4.3.1 : Unit Testing

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.

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

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.

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

Integration Testing:-

Integration testing is a level of software testing where individual units are combined and tested as a group. The purpose 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.

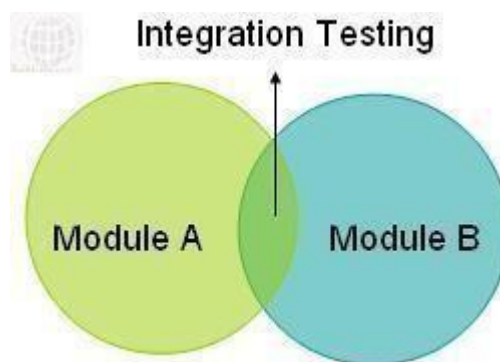


Figure 4.4.1 : Integration Testing

System testing:-

This 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.

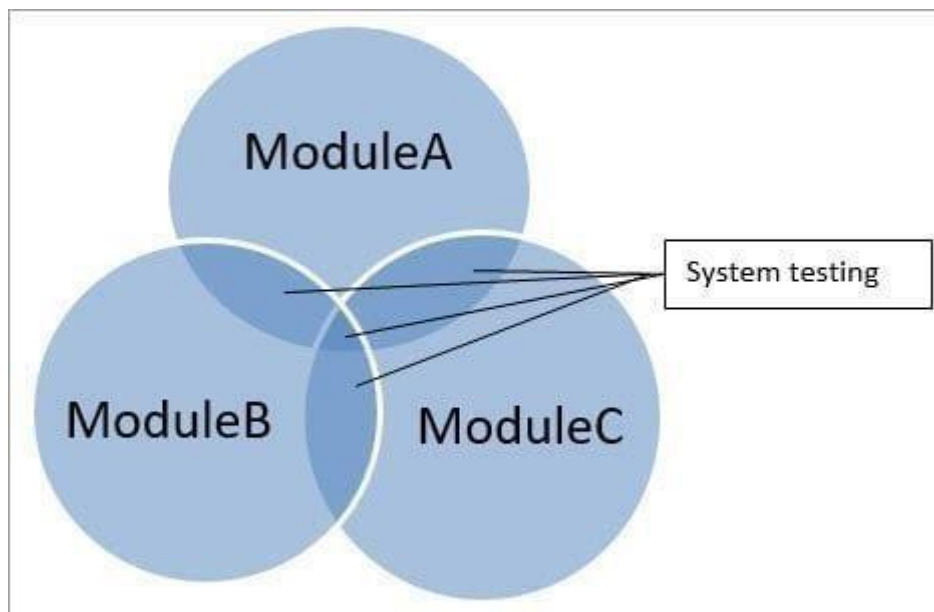


Figure 4.4.2 : System Testing

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”.

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 help 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.

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.



Figure 4.4.3 : Regression Testing

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.

CHAPTER 5

RESULTS AND DISCUSSION

As a result of this a fast pattern recognition based fault detection method is proposed. The new method makes use of the Logistic Regression to explicitly account for some unique characteristics of most semiconductor processes, namely nonlinearity and multimodal trajectories. Compared to Regression, a fault detection method that uses the Logistic Regression rule in the high-dimensional original variable space, Logistic Regression reduces computational complexity and memory requirements by applying the Logistic Regression rule to the low-dimensional principal component Logistic Regression.

We have four Data Sets two datasets are power and two are weather. We need to Load dataset and we have to do extraction of feature from dataset then we need to select a feature what we need. Data Visualization it records the clear image of the feature then we need to Train the model after that we have to send Mixed prediction their we need to test the feature with the Test Data then we will get Prediction Result if the feature do not have fault then that will be in classification if not fault detection appears.

	DATE_TIME	DATACENTER_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	1155031	1EN3M6L0836L7	0.0	0.0	0.0	8256502K
1	15-05-2020 00:00	1155031	1F5E64743053V	0.0	0.0	0.0	8180345K
2	15-05-2020 00:00	1155031	3F04636C596C8D	0.0	0.0	0.0	8087703K
3	15-05-2020 00:00	1155031	7F7A27A1E3F464	0.0	0.0	0.0	7922360K
4	15-05-2020 00:00	1155031	846E206C6a787Ca	0.0	0.0	0.0	7150360K

datacenter_id.head()

	DATE_TIME	DATACENTER_ID	SOURCE_KEY	DC_POWER	AC_POWER	DAILY_YIELD	TOTAL_YIELD
0	15-05-2020 00:00	4155031	4177946B777464e	0.0	0.0	0.0	5425.070000
1	15-05-2020 00:00	4155031	8B4B747718E7245	0.0	0.0	0.0	5102.774400
2	15-05-2020 00:00	4155031	814646464646464	0.0	0.0	0.0	5075.370000
3	15-05-2020 00:00	4155031	774646464646464	0.0	0.0	0.0	4885.070000
4	15-05-2020 00:00	4155031	000074777777777	0.0	0.0	0.0	2177.070000

Figure 5.1 :Power Data Sets

In Exploratory Data Analysis It reads the power Datasets and Weather Datasets. In dataset we have parameter DATE_TIME, DATACENTER_ID, SOURCE_KEY, DC_POWER, AC_POWER, DAILY_YIELD, TOTAL_YIELD. Again It reads the power Datasets and Power Datasets. In dataset we have parameters DATE_TIME, DATACENTER_ID, SOURCE_KEY, AMBIENT_TEMPERATURE, MODULE_TEMPERATURE, IR_RADIATION.

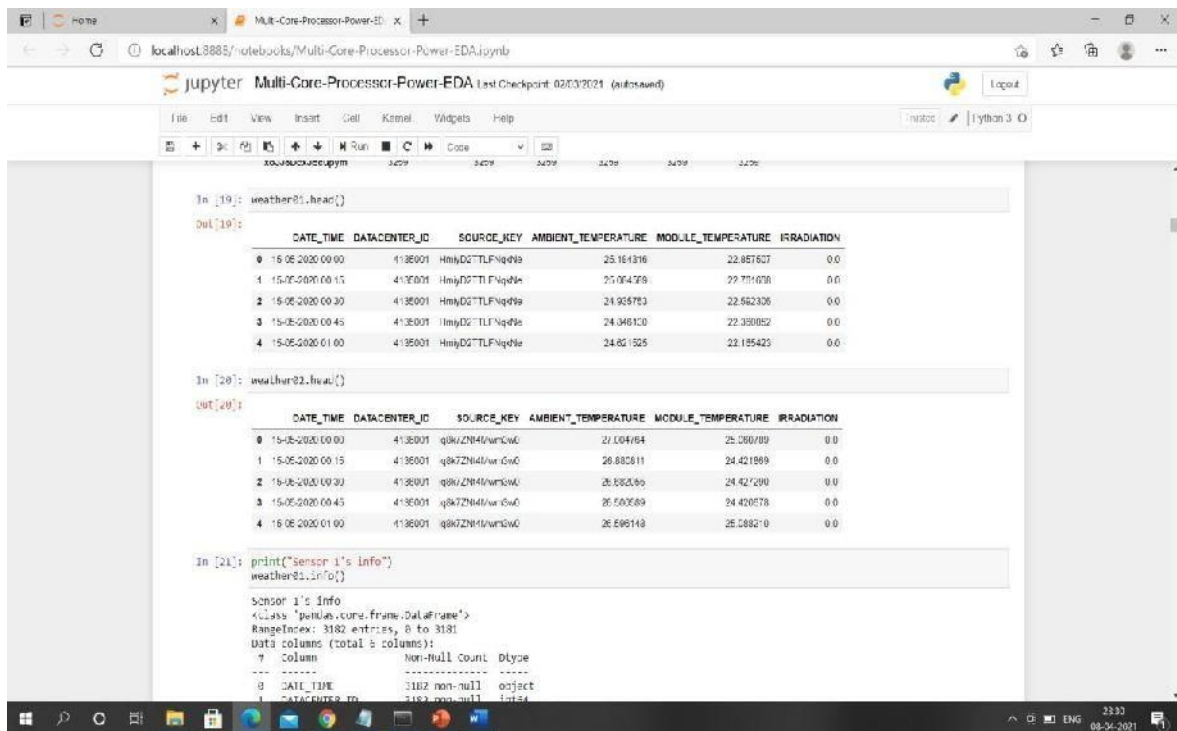


Figure 5.1.1 Info Data Set

2)

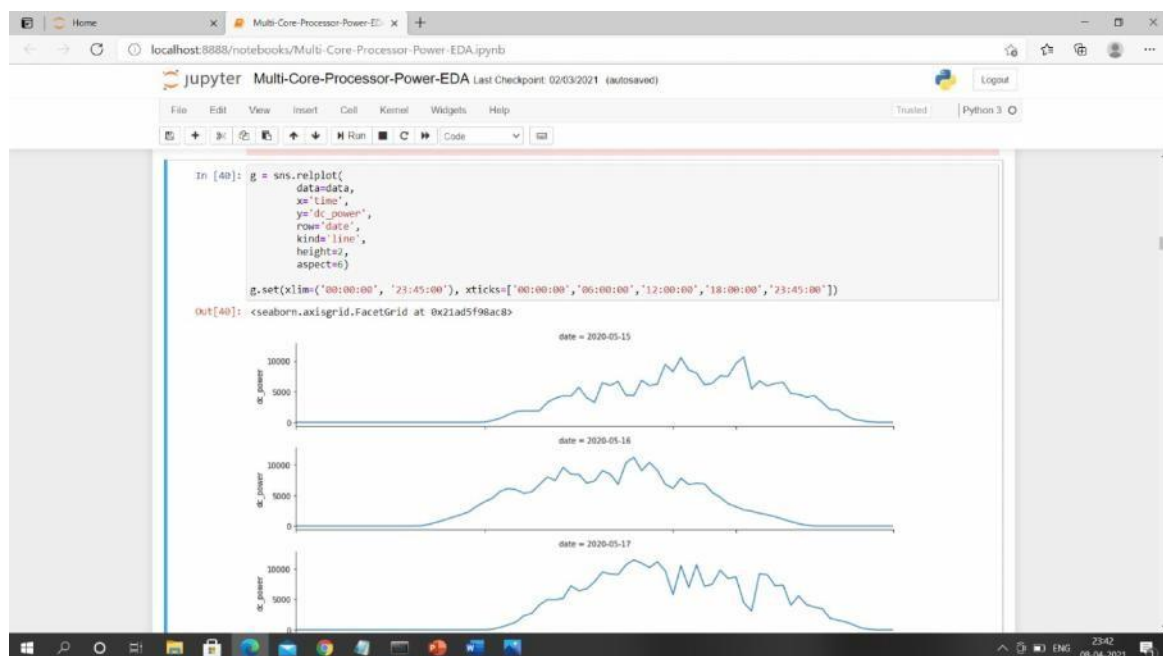


Figure 5.1.2 : Kind Line x,y time,dc_power

In Preprocessing we will get info of power datasets it check the Null Value of two Datasets.

We have inverters in datasets. In Dataset 1 we have 22 inverters and in Dataset 2 we have 22 inverters will take SOURCE_KEY for two Datasets.

From Wheather Dataset we will get info of Wheather dataset and we have Sensor 1 has 1 source key Sensor 2 has 1 source key.check the Null and count of the Paramaters. In dataset1 we will seprate Date and Time From DATA_TIME.In Dataset1 `g = sns.relplot(data=data,x='time',y='dc_power',row='date'kind='line',height=2,aspect=6)``g.set(xlim=('00:00:00','23:45:00'),xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])` Same way but now 'kind='Scatter'.Next it will show dotied form `g = sns.relplot(data=data,x='time',y='daily_yield',row='date',kind='scatter',height=2,aspect=6)``g.set(xlim=('00:00:00','23:45:00'),xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])` same as before but now `y='total_yield'`.In Dataset 1 3264 rows × 8 columns indexs by heatmap we will check in dataset how many NULL and sum of the parameters in dataset.`early_morning=datacenter01_inv_0.between_time('00:00:00','05:45:00')``afternoon=datacenter01_inv_0.between_time('06:00:00','18:30:00')``night=datacenter01_inv_0.between_time('18:45:00','23:45:00')``data=datacenter01_inv_0data['time']=data['time'].astype(str)``sns.set(font_scale=1.5)``g=sns.relplot(data=data,x='time',y='dc_power',col='date',kind='scatter',height=2,aspect=3,col_wrap)``g.set(xlim=('00:00:00','23:45:00'),xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])`.Again same before but now `y='daily_yield'`,.We need to check nullvaluefordataset1`g=sns.relplot(data=data,x='time',y='dc_power',col='date',kind='scatter',height=2,aspect=3,col_wrap=3)``g.set(xlim=('00:00:00','23:45:00'),xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])`same Again but now `y='daily_yield'`,.After these all we have to checkthe null value and show total in one graphical from 2020-05-15 00:00:00 to 2020-06-17 00:00:00.In the same way we need to another dataset of Wheater we need separate data time

```
from DATE_TIME.data = weather01data['time'] =
data['time'].astype(str)sns.set(font_scale =1.5)g =
sns.relplot(data=data,x='time',y='ambient_temperature',col='date',
kind='scatter',height=3,aspect=3,col_wrap=3)g.set(xlim=('00:00:00',
'23:45:00'),
xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00']).same
as before but
```

`y='module_temperature'`, again `y='irradiation'`,.After this by heatmap how many null values for Wheater 1.`weather01['date'] = weather01.index.date`

`weather01['time'] = weather01.index.time` ,`weather02['date'] = weather02.index.date` ,`weather02['time'] = weather02.index.time`.

```
data = weather01
data['time'] = data['time'].astype(str)
sns.set(font_scale=1.5)
g = sns.relplot(data=data,
x='time',y='module_temperature', col='date',kind='scatter',
height=3,aspect=3,col_wrap=3)
g.set(xlim=('00:00:00', '23:45:00'),
xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00']).
```

The screenshot shows a Jupyter Notebook interface with the following code and output:

```
In [90]: full_data = pd.concat([datacenter01.with_weather01, datacenter02.with_weather02], ignore_index=True)
In [91]: full_data
Out[91]:
```

	source_key	dc_power	ps_power	daily_yield	total_yield	date	time	date_time	source_key	ambient temperature	module temperature
0	0	0.0	0.0	0.0	6259659.0	2020-09-16	00:00:00	2020-09-16 00:00:00	HmtyD2TTLFvq04e	25.184376	32.657607
1	0	0.0	0.0	0.0	6259659.0	2020-09-16	00:15:00	2020-09-16 00:15:00	HmtyD2TTLFvq04e	25.1584589	32.761668
2	0	0.0	0.0	0.0	6259659.0	2020-09-16	00:30:00	2020-09-16 00:30:00	HmtyD2TTLFvq04e	24.535753	32.592306
3	0	0.0	0.0	0.0	6259659.0	2020-09-16	00:45:00	2020-09-16 00:45:00	HmtyD2TTLFvq04e	24.846730	32.386862
4	0	0.0	0.0	0.0	6259659.0	2020-09-16	01:00:00	2020-09-16 01:00:00	HmtyD2TTLFvq04e	24.021025	32.165423
...	...	...	...	...	...	...	...	...	...	...	...
143611	21	0.0	0.0	4396.0	209035741.0	2020-09-17	22:45:00	2020-09-17 22:45:00	iq9k7Zt6t8Mm3w0	23.511703	32.648001
143612	21	0.0	0.0	4396.0	209035741.0	2020-09-17	23:00:00	2020-09-17 23:00:00	iq9k7Zt6t8Mm3w0	22.982282	32.711190
143613	21	0.0	0.0	4396.0	209035741.0	2020-09-17	23:15:00	2020-09-17 23:15:00	iq9k7Zt6t8Mm3w0	22.354743	32.492245
143614	21	0.0	0.0	4396.0	209035741.0	2020-09-17	23:30:00	2020-09-17 23:30:00	iq9k7Zt6t8Mm3w0	22.291048	32.573609
...	...	...	...	...	...	...	...	...	...	...	...

Figure 5.1.3 : Weather full_data

3)

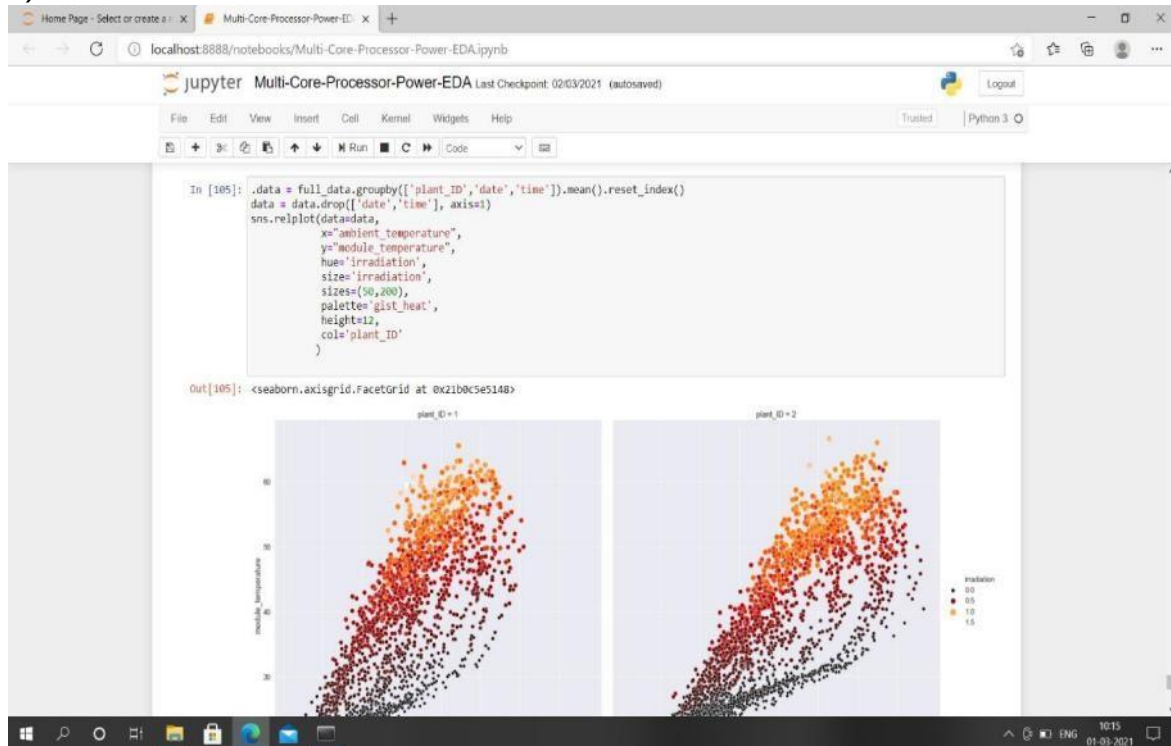


Figure 5.1.4 : Irradiation Ambient,Module_temperature

In Prediction : Prediction Logistic Regression measures the relationship between the dependent variable (our label, what we want to predict)

and the one or more independent variables (our features), by estimating probabilities using it's underlying logistic function. By heatmap datasets Power 1 and Wheather 1

will be show in one graphical of all parameters and heatmap datasets Power 2 and Wheather 2 will be show in one graphical of all parameters.data

```
full_data.groupby(['plant_ID','date','time']).mean().reset_index()
```

```
data = data.drop(['date','time'], axis=1)
sns.relplot(data=data,
            x="ambient_temperature",
            y="module_temperature",
            hue='irradiation',size='irradiation',sizes=(50,200),palette='gist_heat',
            height=12,
```

```
col='plant_ID')anddata=full_data.groupby(['plant_ID','date','time']).me
an().reset_index()data = data.drop(['date','time'], axis=1)
```

```
sns.relplot(data=data, x="irradiation", y="dc_power",
hue='ac_power',size='irradiation',sizes=(50,200),
palette='gist_heat',height=12,col='plant_ID')
```

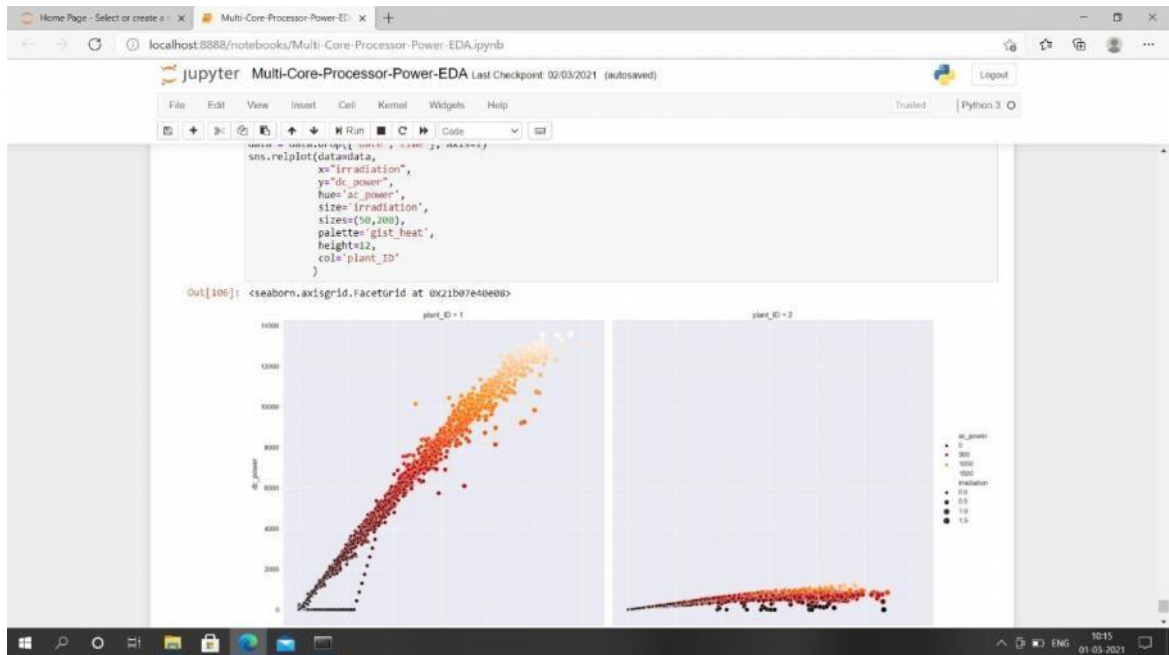


Figure 5.1.5 : hue ac_power x,y,irradiation dc_power

CHAPTER 6

CONCLUSION AND FUTURE WORK

CONCLUSION

In this paper, a fast pattern recognition based fault detection method is proposed. The new method makes use of the Logistic Regression to explicitly account for some unique characteristics of most semiconductor processes, namely nonlinearity and multimodal trajectories. Compared to Regression, a fault detection method that uses the Logistic Regression rule in the high-dimensional original variable space, Logistic Regression reduces computational complexity and memory requirements by applying the Logistic Regression rule to the low-dimensional principal component Logistic Regression

FUTURE WORK

FDC research will be required to correctly classify—as a new class—any novel type of fault encountered in the other fields.

REFERENCES

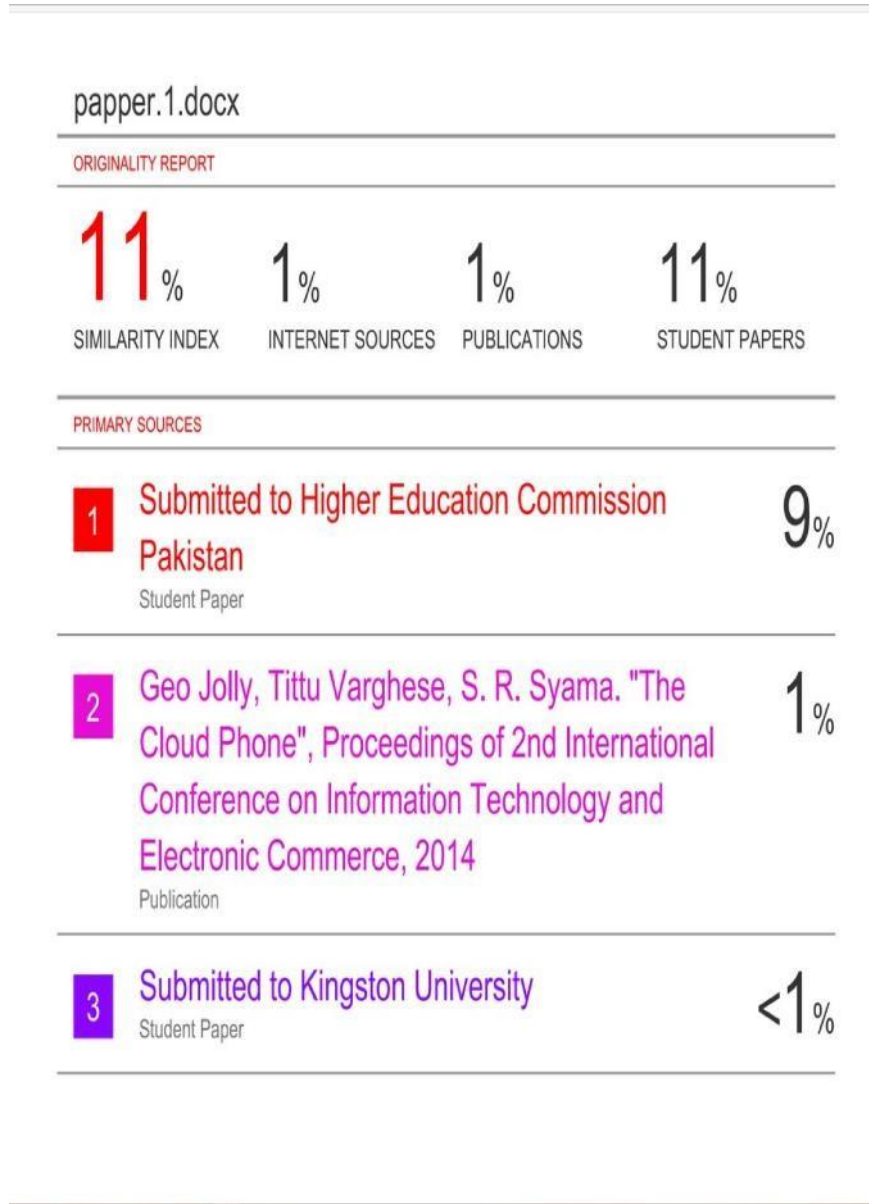
- [1] J. M. Nawaz, M. Z. Arshad, and S. J. Hong, "Fault diagnosis in semiconductor etch equipment using Bayesian networks," *J. Semicond. Tech. Sci.*, vol. 14, no. 2, pp. 252–261, Apr. 2014.
- [2] C.-F. Chien, C.-Y. Hsu, and P.-N. Chen, "Semiconductor fault detection and classification for yield enhancement and manufacturing intelligence," *Flex. Serv. Manuf. J.*, vol. 25, no. 3, pp. 367–388, Sep. 2013.
- [3] Z. Chen, C. Li, and R.-V. Sanchez, "Gearbox fault identification and classification with convolutional neural networks," *Shock Vib.*, vol. 2015, article ID 390134, 2015.
- [4] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. L. Zhao, "Time series classification using multichannels deep convolutional neural networks," in *Proc. WAIM 2014*, Macau, China, 2014, pp. 298–310.
- [5] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
- [6] F. Adly et al., "Randomized general regression network for identification of defect patterns in semiconductor wafer maps," *IEEE Trans. Semicond. Manuf.*, vol. 28, no. 2, pp. 145–152, May 2015, doi: 10.1109/TSM.2015.2405252
- [7] J. Li et al., "The mathematical model and novel final test system for wafer-level packaging," *IEEE Trans. Ind. Informat.*, vol. 13, no. 4, pp. 1817–1824, Aug. 2017, doi: 10.1109/TII.2016.2643694.
- [8] J. Li et al., "New applications of an automated system for high-power LEDs," *IEEE/ASME Trans. Mechatronics*, vol. 21, no. 2, pp. 1035–1042, Apr. 2016, doi: 10.1109/TMECH.2015.2487507.
- [9] J. Li, W. Wang, Y. Xia, H. He, and W. Zhu, "The soft-landing features of a micro- magnetorheological fluid damper," *Appl. Phys. Lett.*, vol. 106, no. 1, 2015, Art. no. 014104, doi: 10.1063/1.4903924.
- [10] J. Li et al., "Structural design and control of a small-MRF damper under 50 N soft- landing applications," *IEEE Trans. Ind. Informat.*, vol. 11, no. 3, pp. 612–619, Jun. 2015, doi: 10.1109/TII.2015.2413353.

- [11] C. Xiao, H. Liao, Y. Wang, J. Li, and W. Zhu, "A novel automated heatpipe cooling device for high-power LEDs," *Appl. Thermal Eng.*, vol. 111, pp. 1320–1329, Jan. 2017, doi: 10.1016/j.applthermaleng.2016.10.041.
- [12] J. Li et al., "An electromechanical model and simulation for test process of the wafer probe," *IEEE Trans. Ind. Electron.*, vol. 64, no. 2, pp. 1284–1291, Feb. 2017, doi: 10.1109/TIE.2016.2615273.
- [13] S. Sunter, "Closing the loop between analog design and test," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, May 2016, pp. 894–897.
- [14] S. Sunter, K. Jurga, and A. Laidler, "Using mixed-signal defect simulation to close the loop between design and test," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 12, pp. 2313–2322, Dec. 2016.
- [15] Coyette, G. Gielen, R. Vanhooren, and W. Dobbelaere, "Optimization of analog fault coverage by exploiting defect-specific masking," in *Proc. 19th IEEE Eur. Test Symp. (ETS)*, May 2014, pp. 1–6.

APPENDIX

A. Paper Acceptance Mail

B. Plagiarism Report



C. Journal Paper

A ROBUST AND DYNAMIC FAULT DETECTION AND CLASSIFICATION OF SEMI-CONDUCTORS USING LOGISTIC REGRESSION

S.Sasanka ^{*1}, S.Anisha ^{*2}, Dr.K.Mohan Prasad ^{*3}

1. Student, Department of Computer Science, Sathyabama Institute of Science and Technology.
2. Student, Department of Computer Science, Sathyabama Institute of Science and Technology.
3. Associate Professor, Department of Computer Science, Sathyabama Institute of Science and Technology.

Abstract—

In the semiconductor industry, the singular goal for automatic graphical assessment is to detect and categorize fabrication defects exploiting advanced image processing practices. The manufacturer main aim to detect the patterns at the quickest feasible recognition of defect models accepts quality management and computerization of production supply chains, companies benefit from a boosted yield and lowered production costs. Because conventional image processing systems are constrained in their capability to detect innovative defect models, and artificial intelligence methodology every so often involve a incredible amount of computational determination, this research presents a innovative deep neural network-based hybrid method.

Keywords— *Machine Learning, Logistic Regression, Neural Network, Deep Learning*

A General System

The designed utilization of equal rapid semi-transmitter circuits can lessen the electrical energy conveyed to the bend to a level low enough to acquire an Arc Flash Category. The utilization of equal fast semi-conductor wires in the field requires the utilization of a designed get together. This paper subtleties the

advancement of such a get together and the troubles defeat to accomplish solid activities. The fruitful finish of this assignment would limit the capital consumption expected to supplant the current gear.

In any case, the dependability of force hardware frameworks is a significant issue nowadays. There are numerous sorts of shortcomings. Agent flaws are short out switch blames and open circuit switch issues. The arrangements depend on the extra parts in the converters. These methods may expand the framework cost and would raise the framework size. This open minded control can decrease the current unbalance, music and secure high unwavering quality.

This coordinated information is utilized for audit, investigation, and assessment of deformities and to create different reports which are valuable for yield improvement. Wafer map age is an interaction of creating an outwardly comparative picture to wafer for review deserts. Through intuitive Wafer map, client can see and break down absconds by perception of imperfections and relating deformity pictures on the guide. Examining should be possible through connection with guide and deformity information record for inspected imperfections can be recovered in instrument's configuration for additional consideration. Because of the many number of

dynamic components, the location and the lenient control of the open switch flaws are more muddled..

Machine Learning

Probabilistic Modeling

Probabilistic modelling is the use of the standards of measurements to information investigation. It was perhaps the most punctual type of AI, it's still generally used right up 'til today. Outstanding amongst other known calculations in this classification is the Naive Bayes calculation. Naive Bayes is a kind of Artificial Intelligence classifier dependent on applying Bayes' hypothesis while expecting that the highlights in the input information are generally autonomous . This type of information examination originates before system and was applied by hand a long time before its first system execution . Bayes' hypothesis and the establishments of measurements date back to the eighteenth century, and these are all you need to begin utilizing Naive Bayes classifiers.

Support Vector Machine SVM

SVMs target taking care of order issues by discovering great decision limits between two arrangements of focuses having a place with two unique classifications. A decision limit can be considered as a line or surface isolating your preparation information into two spaces comparing to two classifications. To arrange new information focuses, you simply need to check which side of the decision limit they fall on.

The information is planned to another high-dimensional portrayal where the decision limit can be communicated as a hyperplane.

A decent decision limit is registered by attempting to expand the distance between the hyperplane and the nearest information focuses from each class, a stage called boosting the edge. This permits the limit to sum up well to new examples outside of the preparation dataset.

Decision Tree & Random Forest

Decision trees are flowchart-like constructions that let you arrange input information focuses or foresee output esteems given inputs. They're not difficult to imagine and decipher. Specifically, the Random Forest calculation presented a hearty, functional interpretation of decision-

tree discovering that includes building an enormous number of particular decision trees and afterward ensembling their outputs. Irregular woodlands are relevant to a wide scope of issues—you could say that they're quite often the second-best calculation for any shallow AI task. A gradient boosting machine, similar as an irregular timberland, is an AI method dependent on ensembling powerless expectation models, by and large decision trees. It utilizes gradient boosting, an approach to improve any AI model by iteratively preparing new models that have practical experience in tending to the flimsy parts of the past models. Applied to decision trees, the utilization of the gradient boosting method brings about models that carefully outflank arbitrary woodlands more often than not, while having comparable properties.

Multiple Logistic Regression Model

Regression methods have become an indispensable part of any information examination worried about depicting the connection between a reaction variable and at least one illustrative variables. Frequently the result variable is discrete, taking on at least two potential qualities. The strategic regression model is the most every now and again utilized regression model for the investigation of these information.

$$g(\mathbf{x}) = \ln \left(\frac{\pi(\mathbf{x})}{1 - \pi(\mathbf{x})} \right) = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + \cdots + \beta_p x_p$$

where, for the multiple logistic regression model

$$\pi(\mathbf{x}) = \frac{e^{g(\mathbf{x})}}{1 + e^{g(\mathbf{x})}}$$

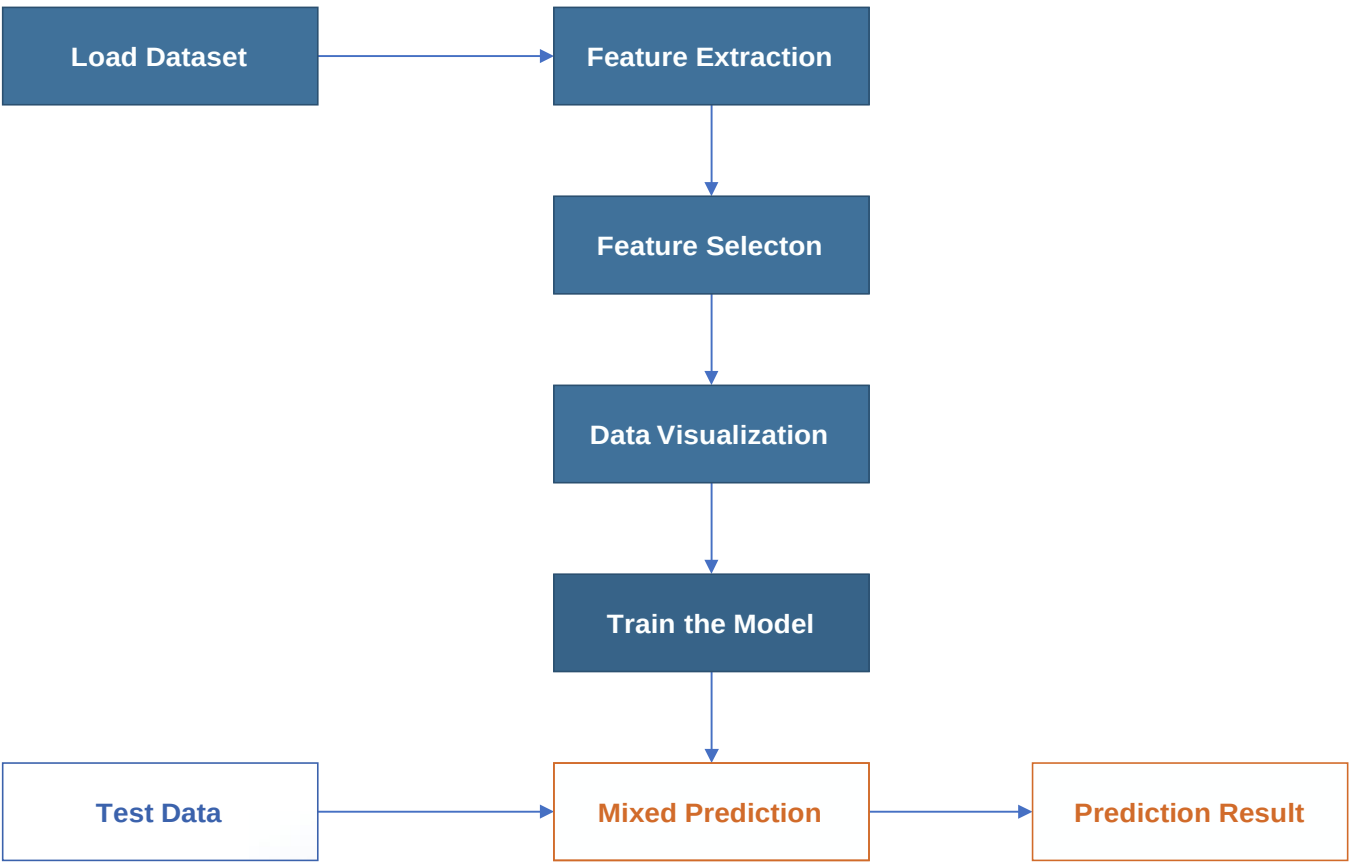
In the event that a portion of the independent variables are discrete, ostensible scale variables like race, sex, treatment gathering, etc, it is unseemly to remember them for the model as though they were span scale variables. The numbers used to address the different levels of these ostensible scale variables are simply identifiers, and have no numeric importance. In the present circumstance, the technique for decision is to utilize an assortment of plan variables (or sham variables). Assume, for instance, that

one of the independent variables is race, which has been coded as "white," "dark," and "other." For this situation, two plan variables are fundamental.

At whatever point a categorical independent variable is incorporated (or avoided) from a model, the entirety of its plan variables ought to be incorporated (or rejected); to do in any case suggests that we have recoded the variable. Assuming k is the quantity of levels of a categorical variable, the commitment to the levels of opportunity for the probability proportion test for the avoidance of this variable is $k - 1$.

Details of Proposed Operations

The function mapping planning from a bunch of pixels to an item personality is extremely convoluted. Learning or assessing this planning appears to be unconquerable whenever handled straightforwardly. Profound learning settle this trouble by breaking the ideal convoluted planning into a progression of settled basic mappings, each portrayed by an alternate layer of the model. The input is introduced at the obvious layer, so named in light of the fact that it contains the variables that we can notice.



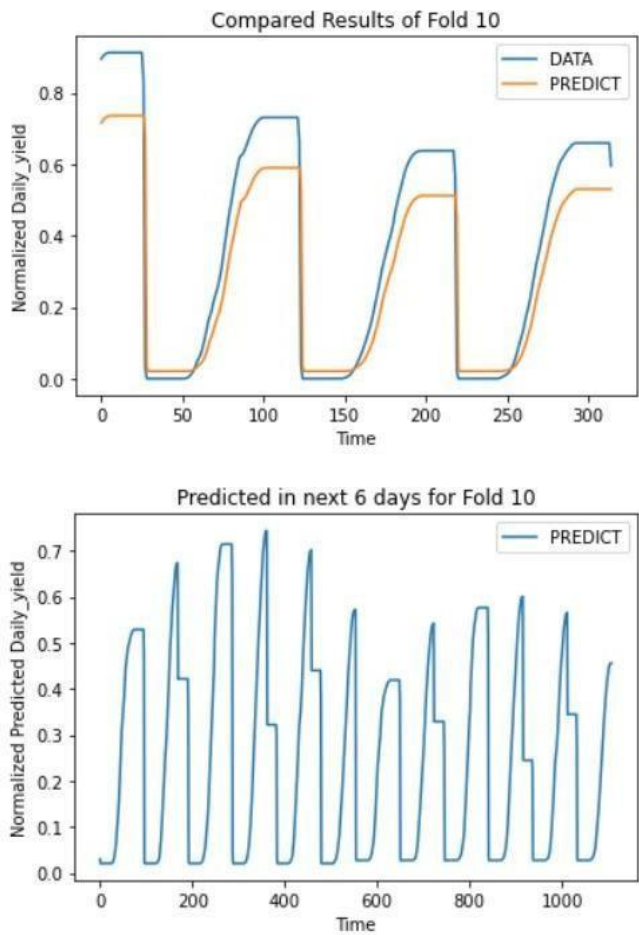
Layer Processing

At that point a progression of covered up layers separates progressively conceptual highlights from the picture. These layers are designated "covered up" on the grounds that their qualities are not given in the information; rather the model should figure out which ideas are valuable for clarifying the connections in the noticed information. The pictures here are representations

of the sort of highlight addressed by each secret unit. Given the pixels, the primary layer can without much of a stretch distinguish edges, by contrasting the splendour of adjoining pixels. Given the previously covered up layer's depiction of the edges, the second secret layer can undoubtedly look for corners and expanded forms, which are unmistakable as assortments of edges. Given the second secret layer's portrayal of the picture as far as

corners and forms, the third secret layer can recognize whole pieces of explicit items, by discovering explicit assortments of shapes and corners. At long last, this depiction of the picture regarding the article parts it contains can be utilized to perceive the items present in the picture.

Experiment Results



conclusion

The acknowledged stacked half breed convolutional neural network based framework joins the upsides of old style picture handling approaches with artificial neural networks and consequently permits a more effective acknowledgment of the best constructions in pixel size. For this reason, the in our application recognized chip and road mistake classes comprise the base of the framework, as they are given to draw the concentration over the degree of detail from the wafer to its chips and roads. Our test outcomes show that the acknowledged framework

outperforms current methodologies of learning-based mechanized visual examination, whereby a qualification relying upon the degree of detail empowers the discovery and grouping of imperfection designs in the prior phases of the assembling cycle.

references

[1] G. Eason, B. Noble, and I. N. Sneddon, "On certain integrals of Lipschitz-Hankel type involving products of Bessel functions," Phil. Trans. Roy. Soc. London, vol. A247, pp. 529–551, April 1955. (references)

[2] J. Clerk Maxwell, A Treatise on Electricity and Magnetism, 3rd ed., vol. 2. Oxford: Clarendon, 1892, pp.68–73.

[3] I. S. Jacobs and C. P. Bean, "Fine particles, thin films and exchange anisotropy," in Magnetism, vol. III, G. T. Rado and H. Suhl, Eds. New York: Academic, 1963, pp. 271–350.

[4] R. Nicole, "Title of paper with only first word capitalized," J. Name Stand. Abbrev., in press.

[5] Y. Yorozu, M. Hirano, K. Oka, and Y. Tagawa, "Electron spectroscopy studies on magneto-optical media and plastic substrate interface," IEEE Transl. J. Magn. Japan, vol. 2, pp. 740–741, August 1987 [Digests 9th Annual Conf. Magnetics Japan, p. 301, 1982].

[6] M. Young, The Technical Writer’s Handbook. Mill Valley, CA: University Science, 1989.

D. Source Code

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import plotly.express as px
import datetime as datetime
import plotly.graph_objects as go
from plotly.subplots import make_subplots
%matplotlib inline

datacenter01 = pd.read_csv('DataCenter_Power_Dataset_1.csv')
weather01 = pd.read_csv('DataCenter_Weather_Dataset_1.csv')
datacenter02 = pd.read_csv('DataCenter_Power_Dataset_2.csv')
weather02 = pd.read_csv('DataCenter_Weather_Dataset_2.csv')
datacenter01.head()

DATE_TIMEDATACENTER_IDSOURCE_KEYDC_POWERAC_
POWERDAILY_YIELDTOTAL_YIELD015-05-2020
00:0041350011BY6WEcLGh8j5v70.00.00.06259559.0115-05-
2020 00:0041350011IF53ai7Xc0U56Y0.00.00.06183645.0215-
05-2020
00:0041350013PZuoBAID5Wc2HD0.00.00.06987759.0315-05-
2020 00:0041350017JYdWkrLSPkdwr40.00.00.07602960.0415-
05-2020 00:004135001McdE0feGgRqW7Ca0.00.00.07158964.0
print("Generation DataCenter 1's info")
datacenter01.info()
Generation DataCenter 1's info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 68778 entries, 0 to 68777
Data columns (total 7 columns):
print("Generation DataCenter 2's info")
datacenter02.info()
Generation DataCenter 2's info
<class 'pandas.core.frame.DataFrame'>
```

```

print('DataCenter 2 has '+
str(datacenter02['SOURCE_KEY'].nunique()) + ' inverters')
DataCenter 1 has 22 inverters
DataCenter 2 has 22 inverters
Check data for inverters
print('DataCenter 1')
datacenter01.groupby('SOURCE_KEY').count()
Datacenter 1
print('DataCenter 2')
datacenter02.groupby('SOURCE_KEY').count()
Datacenter 2
weather01.head()
DATE_TIME DATACENTER_ID SOURCE_KEY AMBIENT_TEMP
ERATURE MODULE_TEMPERATURE IRRADIATION
015-05-2020
00:00 4135001 HmiyD2 TTLFNqkNe 25.18 4316 22.85 75070.0 115-05-2020
00:15 4135001 HmiyD2 TTLFNqkNe 25.08 4589 22.76 16680.0 215-05-2020
00:30 4135001 HmiyD2 TTLFNqkNe 24.93 5753 22.59 23060.0 315-05-2020
00:45 4135001 HmiyD2 TTLFNqkNe 24.84 6130 22.36 08520.0 415-05-2020
01:00 4135001 HmiyD2 TTLFNqkNe 24.62 1525 22.16 54230.0
Weather02.head()
DATE_TIME DATACENTER_ID SOURCE_KEY AMBIENT_TEMP
ERATURE MODULE_TEMPERATURE IRRADIATION
015-05-2020
00:00 4136001 iq8k7ZNt4Mwm3w 27.00 4764 25.06 07890.0 115-05-2020
00:15 4136001 iq8k7ZNt4Mwm3w 26.88 0811 24.42 18690.0 215-05-2020
00:30 4136001 iq8k7ZNt4Mwm3w 26.68 2055 24.42 72900.0 315-05-2020
00:45 4136001 iq8k7ZNt4Mwm3w 26.50 0589 24.42 06780.0 415-05-2020
01:00 4136001 iq8k7ZNt4Mwm3w 26.59 6148 25.08 82100.0

```

```

print("Sensor 1's info")
weather01.info()
Sensor 1's info
<class 'pandas.core.frame.DataFrame'>
print("Sensor 2's info")
weather02.info()
Sensor 2's info
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 3259 entries, 0 to 3258
Data columns (total 6 columns):

print('Sensor 1 has '+str(weather01['SOURCE_KEY'].nunique())+'
source key')
print('Sensor 2 has '+str(weather02['SOURCE_KEY'].nunique())+'
source key')
Sensor 1 has 1 source key
Sensor 2 has 1 source key
print('Sensor 1')
weather01.isnull().sum()
print('Sensor 2')
weather02.count()
print('the total should be '+ str(34*24*4)+' rows')
the total should be 3264 rows
datacenter01.columns = datacenter01.columns.str.lower()
datacenter02.columns = datacenter02.columns.str.lower()
datacenter01.drop('datacenter_id', axis=1, inplace=True)
datacenter02.drop('datacenter_id', axis=1, inplace=True)
datacenter01['date_time']
pd.to_datetime(datacenter01['date_time'],format='%d-%m-%Y
%H:%M')

```

```

datacenter02['date_time'] =
pd.to_datetime(datacenter02['date_time'],format='%d-%m-%Y
%H:%M')

datacenter01['date'] = datacenter01['date_time'].dt.date
datacenter01['time'] = datacenter01['date_time'].dt.time
datacenter02['date'] = datacenter01['date_time'].dt.date
datacenter02['time'] = datacenter01['date_time'].dt.time
datacenter01_inverter_id = datacenter01['source_key'].unique()
datacenter02_inverter_id = datacenter02['source_key'].unique()

datacenter01['source_key'] =
datacenter01['source_key'].apply(lambda x :
int(np.where(datacenter01_inverter_id == x)[0]))

datacenter02['source_key'] =
datacenter02['source_key'].apply(lambda x :
int(np.where(datacenter02_inverter_id == x)[0]))

datacenter01.head()

date_time source_key dc_power ac_power daily_yield total_yield dat
etime 02020-05-1500.00.00.06259559.02020-05-
1500:00:0012020-05-1510.00.00.06183645.02020-05-
1500:00:0022020-05-1520.00.00.06987759.02020-05-
1500:00:0032020-05-1530.00.00.07602960.02020-05-
1500:00:0042020-05-1540.00.00.07158964.02020-05-
1500:00:00

datacenter02.head()

date_time source_key dc_power ac_power daily_yield total_yield dat
etime 02020-05-1500.00.09425.0000002.429011e+062020-05-
1500:00:0012020-05-1510.00.00.0000001.215279e+092020-05-
1500:00:0022020-05-
1520.00.03075.3333332.247720e+092020-05-
1500:00:0032020-05-1530.00.0269.9333331.704250e+062020-
05-1500:00:0042020-05-
1540.00.03177.0000001.994153e+072020-05-1500:00:00

datacenter01[(datacenter01['source_key']==0) &
(datacenter01['date_time'].between('2020-05-15','2020-05-21'))]

data = datacenter01[(datacenter01['source_key']==0) &
(datacenter01['date_time'].between('2020-05-15','2020-05-21'))]

```

```
data['time'] = data['time'].astype(str)
```

C:\anaconda\Data\lib\site-packages\ipykernel_launcher.py:2: SettingWithCopyWarning:

A value is trying to be set on a copy of a slice from a DataFrame.

Try using .loc[row_indexer,col_indexer] = value instead

```
data=data,
```

```
x='time',
```

```
y='dc_power',
```

```
row='date',
```

```
kind='line',
```

```
height=2,
```

```
aspect=6)
```

```
data = datacenter01[(datacenter01['source_key']==0) &
(datacenter01['date_time'].between('2020-05-15','2020-05-21'))]
```

```
data['time'] = data['time'].astype(str)
```

```
g = sns.relplot(
```

```
data=data,
```

```
x='time',
```

```
y='daily_yield',
```

```
row='date',
```

```
kind='scatter',
```

```
height=2,
```

```
aspect=6
```

```
)
```

```
<seaborn.axisgrid.FacetGrid at 0x21ad5f98ac8>
```

```
<seaborn.axisgrid.FacetGrid at 0x21ad8665c88>
```

```
ata = datacenter01[(datacenter01['source_key']==0) &
(datacenter01['date_time'].between('2020-05-15','2020-05-21'))]
```

```
data['date_time'] = data['date_time'].astype(str)
```

```
g = sns.relplot(
```

```
data=data,
```

```
x='date_time',
```

```

y='total_yield',
kind='scatter',
height=6,
aspect=2
)

g.set(xlim=('2020-05-15 00:00:00', '2020-05-21 00:00:00'),
xticks=['2020-05-15 00:00:00','2020-05-17 00:00:00','2020-05-19
00:00:00','2020-05-21 00:00:00'])

<seaborn.axisgrid.FacetGrid at 0x21ad921d808>
fulltime = pd.date_range(start='2020-05-15 00:00',end='2020-06-
17 23:45', freq='15T')

fulltime = pd.DataFrame({'date_time':fulltime})

fulltime

datacenter01_inv_0 =
datacenter01[datacenter01['source_key']==0].reset_index(drop=
True)

datacenter01_inv_0 = pd.merge(fulltime, datacenter01_inv_0,
how='outer')

datacenter01_inv_0

datacenter01_inv_0.index = datacenter01_inv_0['date_time']
datacenter01_inv_0.drop('date_time', axis=1, inplace=True)
sns.heatmap(datacenter01_inv_0.isnull())

<matplotlib.axes._subplots.AxesSubplot at 0x21adab8cfc8>
datacenter01_inv_0['date'] = datacenter01_inv_0.index.date
datacenter01_inv_0['time'] = datacenter01_inv_0.index.time
datacenter01_inv_0['source_key'] = 0
datacenter01_inv_0.isnull().sum()

early_morning = datacenter01_inv_0.between_time('00:00:00','05:45:00')
afternoon = datacenter01_inv_0.between_time('06:00:00','18:30:00')
night = datacenter01_inv_0.between_time('18:45:00','23:45:00')
early_morning['dc_power'].fillna(value=0, inplace=True)
early_morning['ac_power'].fillna(value=0, inplace=True)
early_morning['daily_yield'].fillna(value =0, inplace=True)

```

```

night['dc_power'].fillna(value=0, inplace=True)
night['ac_power'].fillna(value=0, inplace=True)
night['daily_yield'].fillna(method='ffill', inplace=True)
datacenter01_inv_0 = pd.concat([early_morning, afternoon, night])
datacenter01_inv_0 = datacenter01_inv_0.sort_index()
data = datacenter01_inv_0
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)

g = sns.relplot(
    data=data,
    x='time',
    y='dc_power',
    col='date',
    kind='scatter',
    height=2,
    aspect=3,
    col_wrap=3
)

g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])
data = datacenter01_inv_0
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)

g = sns.relplot(
    data=data,
    x='time',
    y='daily_yield',
    col='date',
    kind='scatter',
    height=2,
    aspect=3,
    col_wrap=3
)

g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])
datacenter01_inv_0.isnull().sum()
source_key      0
dc_power        35
ac_power        35
daily_yield     35
total_yield    110

```

```

date            0
time            0
dtype: int64
datacenter01_inv_0[['ac_power','dc_power','daily_yield']] = datacenter01_inv_0
[['ac_power','dc_power','daily_yield']].interpolate(method='time')
data = datacenter01_inv_0
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)

```

```

g = sns.relplot(
    data=data,
    x='time',
    y='dc_power',
    col='date',
    kind='scatter',
    height=2,
    aspect=3,
    col_wrap=3
)

```

```

g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])

```

```

<seaborn.axisgrid.FacetGrid at 0x21ae4efac08>

```

```

data = datacenter01_inv_0
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)

```

```

g = sns.relplot(
    data=data,
    x='time',
    y='daily_yield',
    col='date',
    kind='scatter',
    height=2,
    aspect=3,
    col_wrap=3
)

```

```

g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])

```

```

<seaborn.axisgrid.FacetGrid at 0x21aea2dbec8>

```

```

datacenter01
datacenter01_inv_0['total_yield'] = datacenter01_inv_0['total_yield'].interpolate(
method='time')
data = datacenter01_inv_0

```

```

data.index = data.index.astype(str)

f, ax = plt.subplots(figsize=(12,8))
ax = sns.lineplot(x=data.index,
                  y='total_yield',
                  data = data
                  )

ax.set(xlim=('2020-05-15 00:00:00','2020-06-17 00:00:00'),xticks=['2020-05-15
00:00:00','2020-06-17 00:00:00'])
[<matplotlib.axis.XTick at 0x21af0675c48>,
 <matplotlib.axis.XTick at 0x21ae6039508>],
datacenter01_inv = [0]*22

def data_filling(inverter_id):
    datacenter01_inv[inverter_id] = datacenter01[datacenter01['source_key']==i
nverter_id].reset_index(drop=True)
    datacenter01_inv[inverter_id] = pd.merge(fulltime, datacenter01_inv[inverter
_id], how='outer')
    datacenter01_inv[inverter_id]['date'] = datacenter01_inv[inverter_id]['date_ti
me'].dt.date
    datacenter01_inv[inverter_id]['time'] = datacenter01_inv[inverter_id]['date_ti
me'].dt.time
    datacenter01_inv[inverter_id]['source_key'] = inverter_id
    datacenter01_inv[inverter_id].index = datacenter01_inv[inverter_id]['date_ti
me']
    datacenter01_inv[inverter_id].drop('date_time', axis=1, inplace=True)
    early_morning = datacenter01_inv[inverter_id].between_time('00:00:00','05:
45:00')
    afternoon    = datacenter01_inv[inverter_id].between_time('06:00:00','18:30:
00')
    night        = datacenter01_inv[inverter_id].between_time('18:45:00','23:45:0
0')
    early_morning['dc_power'].fillna(value=0, inplace=True)
    early_morning['ac_power'].fillna(value=0, inplace=True)
    early_morning['daily_yield'].fillna(value =0, inplace=True)
    night['dc_power'].fillna(value=0, inplace=True)
    night['ac_power'].fillna(value=0, inplace=True)
    night['daily_yield'].fillna(method='ffill', inplace=True)
    datacenter01_inv[inverter_id] = pd.concat([early_morning,afternoon, night])
    datacenter01_inv[inverter_id] = datacenter01_inv[inverter_id].sort_index()
    datacenter01_inv[inverter_id]['dc_power'] = datacenter01_inv[inverter_id]['dc
_power'].interpolate(method='time')
    datacenter01_inv[inverter_id]['ac_power'] = datacenter01_inv[inverter_id]['ac
_power'].interpolate(method='time')

```

```

datacenter01_inv[inverter_id]['daily_yield'] = datacenter01_inv[inverter_id]['daily_yield'].interpolate(method='time')
datacenter01_inv[inverter_id]['total_yield'] = datacenter01_inv[inverter_id]['total_yield'].interpolate(method='time')

for i in range (22):
    data_filling(i)
datacenter01 = pd.concat(datacenter01_inv)
datacenter02_inv = [0]*22

def data_filling(inverter_id):
    datacenter02_inv[inverter_id] = datacenter02[datacenter02['source_key']==inverter_id].reset_index(drop=True)
    datacenter02_inv[inverter_id] = pd.merge(fulltime, datacenter02_inv[inverter_id], how='outer')
    datacenter02_inv[inverter_id]['date'] = datacenter02_inv[inverter_id]['date_time'].dt.date
    datacenter02_inv[inverter_id]['time'] = datacenter02_inv[inverter_id]['date_time'].dt.time
    datacenter02_inv[inverter_id]['source_key'] = inverter_id
    datacenter02_inv[inverter_id].index = datacenter02_inv[inverter_id]['date_time']
    datacenter02_inv[inverter_id].drop('date_time', axis=1, inplace=True)
    early_morning = datacenter02_inv[inverter_id].between_time('00:00:00','05:45:00')
    afternoon = datacenter02_inv[inverter_id].between_time('06:00:00','18:30:00')
    night = datacenter02_inv[inverter_id].between_time('18:45:00','23:45:00')
    early_morning['dc_power'].fillna(value=0, inplace=True)
    early_morning['ac_power'].fillna(value=0, inplace=True)
    early_morning['daily_yield'].fillna(value=0, inplace=True)
    night['dc_power'].fillna(value=0, inplace=True)
    night['ac_power'].fillna(value=0, inplace=True)
    night['daily_yield'].fillna(method='ffill', inplace=True)
    datacenter02_inv[inverter_id] = pd.concat([early_morning, afternoon, night])
    datacenter02_inv[inverter_id] = datacenter02_inv[inverter_id].sort_index()
    datacenter02_inv[inverter_id]['dc_power'] = datacenter02_inv[inverter_id]['dc_power'].interpolate(method='time')
    datacenter02_inv[inverter_id]['ac_power'] = datacenter02_inv[inverter_id]['ac_power'].interpolate(method='time')
    datacenter02_inv[inverter_id]['daily_yield'] = datacenter02_inv[inverter_id]['daily_yield'].interpolate(method='time')

```

```

datacenter02_inv[inverter_id]['total_yield'] = datacenter02_inv[inverter_id]['total_yield'].interpolate(method='time')

for i in range (22):
    data_filling(i)
datacenter02 = pd.concat(datacenter02_inv)
weather01.columns = weather01.columns.str.lower()
weather02.columns = weather02.columns.str.lower()
weather01.drop('datacenter_id', axis=1, inplace=True)
weather02.drop('datacenter_id', axis=1, inplace=True)
weather01['date_time'] = pd.to_datetime(weather01['date_time'], format = '%d-%m-%Y %H:%M')
weather02['date_time'] = pd.to_datetime(weather02['date_time'], format = '%d-%m-%Y %H:%M')
weather01['date'] = weather01['date_time'].dt.date
weather01['time'] = weather01['date_time'].dt.time
weather02['date'] = weather02['date_time'].dt.date
weather02['time'] = weather02['date_time'].dt.time
weather01.head()
date_timesource_keyambient_temperaturemodule_
data = weather01
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)
g = sns.relplot(data=data,
                x='time',
                y='ambient_temperature',
                col='date',
                kind='scatter',
                height=3,
                aspect=3,
                col_wrap=3
                )
g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00','06:00:00','12:00:00','18:00:00','23:45:00'])
<seaborn.axisgrid.FacetGrid at 0x21ad59a0e08>
data = weather01
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)
g = sns.relplot(data=data,
                x='time',
                y='module_temperature',
                col='date',
                kind='scatter',
                height=3,
                aspect=3,

```

```

col_wrap=3)
g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00', '06:00:00', '12:00:00', '18:00:00', '23:45:00'])
data = weather01
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)
g = sns.relplot(data=data,
                x='time',
                y='irradiation',
                col='date',
                kind='scatter',
                height=3,
                aspect=3,
                col_wrap=3)
g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00', '06:00:00', '12:00:00', '18:00:00', '23:45:00'])
weather01 = pd.merge(fulltime, weather01, how='outer')
weather02 = pd.merge(fulltime, weather02, how='outer')
weather01.index = weather01['date_time']
weather02.index = weather02['date_time']
sns.heatmap(weather01.isnull())
weather01['date'] = weather01.index.date
weather01['time'] = weather01.index.time
weather02['date'] = weather02.index.date
weather02['time'] = weather02.index.time
weather01[['ambient_temperature', 'module_temperature', 'irradiation']] = weather01[['ambient_temperature', 'module_temperature', 'irradiation']].interpolate(method='time')
weather02[['ambient_temperature', 'module_temperature', 'irradiation']] = weather02[['ambient_temperature', 'module_temperature', 'irradiation']].interpolate(method='time')
data = weather01
data['time'] = data['time'].astype(str)
sns.set(font_scale =1.5)
g = sns.relplot(data=data,
                x='time',
                y='module_temperature',
                col='date',
                kind='scatter',
                height=3,
                aspect=3,
                col_wrap=3)
g.set(xlim=('00:00:00', '23:45:00'), xticks=['00:00:00', '06:00:00', '12:00:00', '18:00:00', '23:45:00'])
<seaborn.axisgrid.FacetGrid at 0x21a89f88d08>

```

```

datacenter01_with_weather01_inv= [0]*22
datacenter02_with_weather02_inv= [0]*22
for i in range(22):
    datacenter01_with_weather01_inv[i] = pd.concat([datacenter01_inv[i],weather01.drop(['date','time'], axis=1)], axis=1)
for i in range(22):
    datacenter02_with_weather02_inv[i] = pd.concat([datacenter02_inv[i],weather02.drop(['date','time'], axis=1)], axis=1)
datacenter01_with_weather01 = pd.concat(datacenter01_with_weather01_inv)
datacenter02_with_weather02 = pd.concat(datacenter02_with_weather02_inv)
datacenter01_with_weather01.head()
source_keydc_powerac_powerdaily_yieldtotal_yieldddatetimedate_timesource_keyambient_temperaturemodule_temperatureirradiationdate_time2020-05-15
datacenter01_with_weather01['plant_ID'] = '1'
datacenter02_with_weather02['plant_ID'] = '2'
full_data = pd.concat([datacenter01_with_weather01, datacenter02_with_weather02], ignore_index=True)
full_data
data1=datacenter01.groupby(['source_key']).sum().reset_index()
data1['source_key'] = data1['source_key'].apply(lambda x: datacenter01_inverter_id[x])
data2=datacenter02.groupby(['source_key']).sum().reset_index()
data2['source_key'] = data2['source_key'].apply(lambda x: datacenter02_inverter_id[x])
specs = [[{'type':'domain'}, {'type':'domain'}]]
fig = make_subplots(rows=1, cols=2, specs=specs)
pull_factor = [0]*22
pull_factor[7] = 0.05
fig.add_trace(go.Pie(labels='P1 ' + data1['source_key'],
                        values=data1['dc_power'],
                        name='DataCenter 1',
                        title='DataCenter 1',
                        titlefont=dict(
                            size=25
                        ),
                        hovertemplate="%{label} <br />generates %{value:,.0f} kW",
                        marker_colors = px.colors.qualitative.Dark24,
                        legendgroup = 'DataCenter 1',
                    ), 1, 1)

fig.add_trace(go.Pie(labels='P2 ' + data2['source_key'],
                        values=data2['dc_power'],
                        name='DataCenter 2',
                        title='DataCenter 2',
                        titlefont=dict(

```

```

        ),
        hovertemplate="%{label} <br />generates %{value:,.0f} kW",
        marker_colors = px.colors.qualitative.Light24,
        legendgroup = 'DataCenter 2',
        pull = pull_factor,
    ), 1, 2)

fig.update_traces(hole=.4)

fig.update_layout(
    title_text="DC Power Generation of each Inverter"
)
data=datacenter01.groupby(['source_key','date']).sum().reset_index()
data['source_key'] = data['source_key'].apply(lambda x: datacenter01_inverter
_id[x])

fig=px.bar(
    data_frame = data,
    x = data['date'],
    y = data['dc_power'],
    color = 'source_key',
    color_discrete_sequence = px.colors.qualitative.Dark24,
    hover_data = {'date':True,
                  'source_key':True,
                  'dc_power':':,.0f',
                  },
    opacity = 0.8,
    labels={'date':'date',
            'dc_power':'DC Power Generated (kW)',
            'source_key':'Inverter ID'
            },
    title='DC Power Generated in DataCenter 1 based on date',
    height = 650
)
data=datacenter02.groupby(['source_key','date']).sum().reset_index()
data['source_key'] = data['source_key'].apply(lambda x: datacenter02_inverter
_id[x])

fig=px.bar(
    data_frame = data,
    x = data['date'],
    y = data['dc_power'],
    color = 'source_key',
    color_discrete_sequence = px.colors.qualitative.Light24,

```

```

hover_data = {'date':True,
              'source_key':True,
              'dc_power'::,.0f',
              },

labels={'date':'date',
        'dc_power':'DC Power Generated (kW)',
        'source_key':'Inverter ID'},
title='DC Power Generated in DataCenter 2 based on date',

)
data1=datacenter01.groupby(['source_key']).sum().reset_index()
data1['source_key'] = data1['source_key'].apply(lambda x: datacenter01_inverter_id[x])
data2=datacenter02.groupby(['source_key']).sum().reset_index()
data2['source_key'] = data2['source_key'].apply(lambda x: datacenter02_inverter_id[x])

specs = [[{'type':'domain'}, {'type':'domain'}]]
fig = make_subplots(rows=1, cols=2, specs=specs

fig.add_trace(go.Pie(labels='P1 ' + data1['source_key'],
                      values=data1['ac_power'],
                      name='DataCenter 1',
                      title='DataCenter 1',
                      titlefont=dict(

                      ),
                      hovertemplate="%{label} <br />generates %{value:,.0f} kW",
                      marker_colors = px.colors.qualitative.Dark24,
                      legendgroup = 'DataCenter 1',
                      ), 1, 1)

fig.add_trace(go.Pie(labels='P2 ' + data2['source_key'],
                      values=data2['ac_power'],
                      name='DataCenter 2',
                      title='DataCenter 2',
                      titlefont=dict(

                      ),
                      hovertemplate="%{label} <br />generates %{value:,.0f} kW",
                      marker_colors = px.colors.qualitative.Light24,
                      legendgroup = 'DataCenter 2',
                      pull =pull_factor,
                      ), 1, 2)

```

```

fig.update_traces(hole=.4)

fig.update_layout(
    title_text="AC Power Generation of each Inverter"
)
data=datacenter01.groupby(['source_key','date']).sum().reset_index()
data['source_key'] = data['source_key'].apply(lambda x: datacenter01_inverter_id[x])

fig=px.bar(
    data_frame = data,
    x = data['date'],
    y = data['ac_power'],
    color = 'source_key',
    color_discrete_sequence = px.colors.qualitative.Dark24,
    hover_data = {'date':True,
                  'source_key':True,
                  'ac_power':':,.0f',
                  },
    opacity = 0.8,
    labels={'date':'date',
            'ac_power':'AC Power Generated (kW)',
            'source_key':'Inverter ID'},
    title='AC Power Generated in DataCenter 1 based on date',
    height = 650
)
data=datacenter02.groupby(['source_key','date']).sum().reset_index()
data['source_key'] = data['source_key'].apply(lambda x: datacenter02_inverter_id[x])

fig=px.bar(
    data_frame = data,
    x = data['date'],
    y = data['ac_power'],
    color = 'source_key',
    color_discrete_sequence = px.colors.qualitative.Light24,
    hover_data = {'date':True,
                  'source_key':True,
                  'ac_power':':,.0f',
                  },
    opacity = 0.8,
    labels={'date':'date',
            'ac_power':'AC Power Generated (kW)',
            'source_key':'Inverter ID'},

```

```

        title='AC Power Generated in DataCenter 2 based on date',
        height = 650
    )
    data1=datacenter01
    data2=datacenter02

    fig = go.Figure()

    fig.add_trace(go.Scattergl(x=data1['time'],
                              y=data1['dc_power'],
                              mode='markers',
                              marker=dict(
                                  size=4,
                                  color= data1['dc_power'],
                                  cauto=True,
                                  colorscale ='Oryel',
                                  opacity=0.3
                              ),
                              name='DataCenter 1 DC power'))

    fig.add_trace(go.Scatter(x=data1['time'],
                              y=data1.groupby('time').mean()['dc_power'],
                              mode='lines',
                              line=dict(
                                  color='DarkGray',
                                  width=3
                              ),
                              name='DataCenter 1 Mean'))

    fig.add_trace(go.Scattergl(x=data2['time'],
                              y=data2['dc_power'],
                              mode='markers',
                              marker=dict(
                                  size=4,
                                  color= data2['dc_power'],
                                  cauto=True,
                                  colorscale ='Blugrn',
                                  opacity=0.3
                              ),
                              name='DataCenter 2 DC power'))

    fig.add_trace(go.Scatter(x=data2['time'],
                              y=data2.groupby('time').mean()['dc_power'],
                              mode='lines',
                              line=dict(

```

```

        color='DarkOliveGreen',
        width=3
    ),
    name='DataCenter 2 Mean'))

fig.update_layout(title= 'DC Power Generation by time',
                    height = 600)
data1=datacenter01
data2=datacenter02

fig = go.Figure()

fig.add_trace(go.Scattergl(x=data1['time'],
                           y=data1['ac_power'],
                           mode='markers',
                           marker=dict(
                               size=4,
                               color= data1['ac_power'],
                               cauto=True,
                               colorscale = 'Oryel',
                               opacity=0.3
                           ),
                           name='DataCenter 1 AC power'))

fig.add_trace(go.Scatter(x=data1['time'],
                          y=data1.groupby('time').mean()['ac_power'],
                          mode='lines',
                          line=dict(
                              color='DarkGray',
                              width=3
                          ),
                          name='DataCenter 1 Mean'))

fig.add_trace(go.Scattergl(x=data2['time'],
                            y=data2['ac_power'],
                            mode='markers',
                            marker=dict(
                                size=4,
                                color= data2['ac_power'],
                                cauto=True,
                                colorscale = 'Blugrn',
                                opacity=0.3
                            ),
                            name='DataCenter 2 AC power'))

```

```

fig.add_trace(go.Scatter(x=data2['time'],
                        y=data2.groupby('time').mean()['ac_power'],
                        mode='lines',
                        line=dict(
                            color='DarkOliveGreen',
                            width=3
                        ),
                        name='DataCenter 2 Mean'))

fig.update_layout(title= 'AC Power Generation by time',
                  height = 600)
plt.figure(figsize=(10,8))
sns.heatmap(datacenter01_with_weather01.corr(), annot=True)
<matplotlib.axes._subplots.AxesSubplot at 0x21b0a06ea88>
.data = full_data.groupby(['plant_ID','date','time']).mean().reset_index()
data = data.drop(['date','time'], axis=1)
sns.relplot(data=data,
            x="ambient_temperature",
            y="module_temperature",
            hue='irradiation',
            size='irradiation',
            sizes=(50,200),
            palette='gist_heat',
            height=12,
            col='plant_ID'
            )
<seaborn.axisgrid.FacetGrid at 0x21b0c5e5148>
data = full_data.groupby(['plant_ID','date','time']).mean().reset_index()
data = data.drop(['date','time'], axis=1)
sns.relplot(data=data,
            x="irradiation",
            y="dc_power",
            hue='ac_power',
            size='irradiation',
            sizes=(50,200),
            palette='gist_heat',
            height=12,
            col='plant_ID'
            )

```