CHAPTER-1

INTRODUCTION

1.1 GENERAL

Digitalization of manufacturing such as Industry 4.0 is rapidly spreading. This leads to a drastic change of manufacturing paradigm. Traditionally, engineers understand the physical world directly by watching and touching real objects and real machine tools, and embody their ideas in the physical world through the manufacturing process directly with their hands. But digitalization[2] inserts the digital world between the physical world and the engineers. As a result, the interaction between engineers and objects changes from direct to indirect. Such digitalization causes several problems. Among others, this paper focuses on the following two problems:

Problem 1: The current trends of digitalization might not fit the style of the manufacturing industry and may result in extinguishing the strengths of the manufacturing industry.

Problem 2: Some engineers and technicians cannot follow digitalization. In the digitized manufacturing systems, engineering activities of manufacturing systems (including planning, design, construction, operation, maintenance, improvement, and replacement) are drastically reformed. In other words, while traditionally they constructed and improved manufacturing systems only in the physical world, they should execute various engineering activities by utilizing the cyber world in addition to the physical world.

The objective of this research is to develop an application that results in the solutions of the above two problems. More specifically, the program aims to develop the ability of the manufacturing system engineers to operate, execute fault diagnosis and recovery, and design manufacturing systems, through exercising Digital Triplet of learning factories. Industry 4.0 refers to the fourth phase of the Industrial Revolution and focuses on machine interconnectivity, automation, machine learning and real-time data.

When applied to a specific item of equipment, it is often referred to as a smart device or edge technology.

More pump manufacturers are installing smart pump packages that integrate low-cost sensors, variable speed drives and controllers to process the data locally. The added sensors measure pump suction and discharge pressures, pump vibration and temperatures of the pump seal, bearings and motor.

Instead of sending all the information to the cloud for further analysis, the smart device processes the measured data on the local controller and is able to give a warning when a pump is operating outside the control parameters. Since these calculations occur on a device outside the SCADA system, smart pumps can be considered edge technology.

A smart pump controller operational is an example of a digital Triplets being used to simulate the performance of a physical pump.

The mathematical model is based on the energy balance in the pump; specifically, how electrical energy supplied to the pump drive is converted to fluid energy leaving the pump discharge nozzle. The data describing pump operation is found on the manufacturer's pump curve. Some of the installed instruments give the digital Triplets the as-operating data it needs to simulate pump performance; others provide the as-measured values needed to validate the accuracy of the digital Triplets.

Insights from smart pumps can detect many things, such as dry running, operating below a minimum flow, pumping against a closed discharge valve and high temperature and pressure conditions, when the pump is not operating efficiently. They enable you to control the process by adjusting pump speed.

As always, smart devices can still send the operating data and the results to the cloud where machine learning algorithms using AI can provide additional insight about the operation of the piping system.

CHAPTER 2

Literature Review

2.1. From Digital Twin to Digital Triplet

2.1.1. Development of Digital Twin Technology

The Digital Twin concept arose from the natural evolution of early forms of physical and digital object interfacing known as digital models and digital shadows. A digital model is defined as a virtual representation of a physical system without automatic data exchange between the two objects . A digital shadow is an enhancement of the digital model since it introduces the integration of a one way communication channel between the physical system and its virtual counterpart. In such a system, the physical model can send data caused by a change in state to the virtual object, but the reverse is not possible.

A Digital Twin is distinct from the above mentioned, due to its full integration of information communication between the physical and virtual objects. A change in state in the physical object is registered in the virtual system and any applied change in state to the virtual system can affect the physical object.

Several key technologies are driving the increased adoption of Digital Twins in industry. These include but are not limited to, continuous and discrete simulation methods, communication protocols (OPC-UA and MQTT), Internet of things, Cloud computing, Big data and data fusion.

When integrated in CPS most Digital Twin applications include but are not limited to, layout planning, preventive maintenance and production planning and control

2.1.2. Digital Triplet

The Digital Triplet concept was developed by Umeda et al. in order to tackle two key problems in the Japanese manufacturing industry;

i. The adoption of digitisation in Japanese manufacturing industries might not fit the prevailing Japanese engineering philosophy and might extinguish its strengths.

ii. Engineers/technicians found it difficult to adopt digitisation due to the need to apply their knowledge and experience to effect changes not only to the physical system but also to their virtual counterparts.

The key observation was that traditionally, system improvements were applied directly to the physical manufacturing systems but there was a need to execute such activities by utilising both virtual and physical worlds. The solution involved the integration of physical, virtual and intelligent activity worlds resulting in the development of the Digital Triplet concept.

Digital Triplet was developed as a concept and provides a rich area of exploration through research into possible implementation methods and their applications. An example of such an implementation strategy involves the use of machine learning modules in the intelligent activity layer and uses machine to machine communication to form the necessary interconnection between the layers. Machine learning is a key contributor to the achievement of Industrie 4.0 agenda and has a growing body of research with a wide variety of applications in industry

2.2. OPC-UA in Industry 4.0

Open Platform Communication (OPC) is a communication protocol that enables secure and reliable exchange of data between industrial hardware devices. It contains a series of specifications that define the interface of Clients and Servers, as well as between Servers . OPC Data Access was the first OPC Classic specification released in August 1996 with the first revision occurring in 1997. The standard quickly received support from commercial products leading to it being adopted as the industrial standard .

During its early adoption phase, the OPC protocol encountered several challenges which included:

i. Restricted use to only Windows operating system;

ii. Difficulty in handling and integrating different OPC services, i.e.OPC-AE, OPC-DA, OPC-HDA;

iii. Incompatibilities with internet firewalls protocols;

iv. Emergent security issues since initially access and data security was not a concern.

To address these limitations, the OPC Foundation reworked the standard specifications in order to future proof the protocol and unify different address spaces. This resulted in the development of Open Platform Communication-Unified Architecture (OPC-UA). This new protocol allows access to other specifications and ensures secure data exchange between server–server and

client–server communications links. It provides a layer of interoperability, which enables communication regardless of the native operating system

2.3. Integration of Machine learning in Digital Twins and OPC-UA

Generally, artificial intelligence (AI) and machine learning (ML) techniques have been used to perform various tasks in line with Industrie 4.0, including, but not limited to, predictive maintenance, health monitoring, fault diagnosis, adaptive control and operation process optimisation. Integration of such features into the Digital Twin technology has been explored in various manufacturing scenarios, as demonstrated . The integration of both AI and ML with Digital Twin technology is beneficial in enhancing the interaction between the virtual and physical entity so that processes and operations can be analysed, predicted and optimised. This is achievable since Digital Twin technology not only emphasises the importance of simulation in virtual space but also allows the interaction and execution of intelligent activities between physical and virtual spaces during system operation . Artificial intelligence and machine learning requires standardised access to data sources, which it analyses to gain insight into systems. OPC-UA technology comes in handy in addressing this need as it enables a platform-independent interoperability standard for moving data between systems and devices in operation. Intelligent services can utilise data to optimise operations or perform predictive maintenance only if the right levels and standards of connectivity are met .

CHAPTER 3

EXISTING SYSTEM:

- In the existing system each of these terms describe a process that uses new technology to improve and expand on existing processes. For example, in 1982, the first microcomputer, which weighed 21 pounds, had two 90-KB floppy disks to store the program and data, a single eight-bit processor, 64 kilobytes of random-access memory, a keyboard, and 5-inch monitor. The piping program that we developed could calculate the balanced flow rates and pressures in a network of over 500 pipelines and send the results to a printer.
- Today, this program still calculates the balanced flow rates and pressures within a piping system[3]. But now data can be stored on the cloud, and the application can run on an array of computers with the horsepower to swiftly process data from systems of any size. The calculated results can be displayed on multiple monitors, provide insight to system operation, or input data directly into other mission critical applications.
- In the past, plant instrumentation was expensive to purchase, install and maintain. Most plant instrumentation was used to control the process and offer indications on system operation to the plant's staff.

CHAPTER 4

PROPOSED SYSTEM:

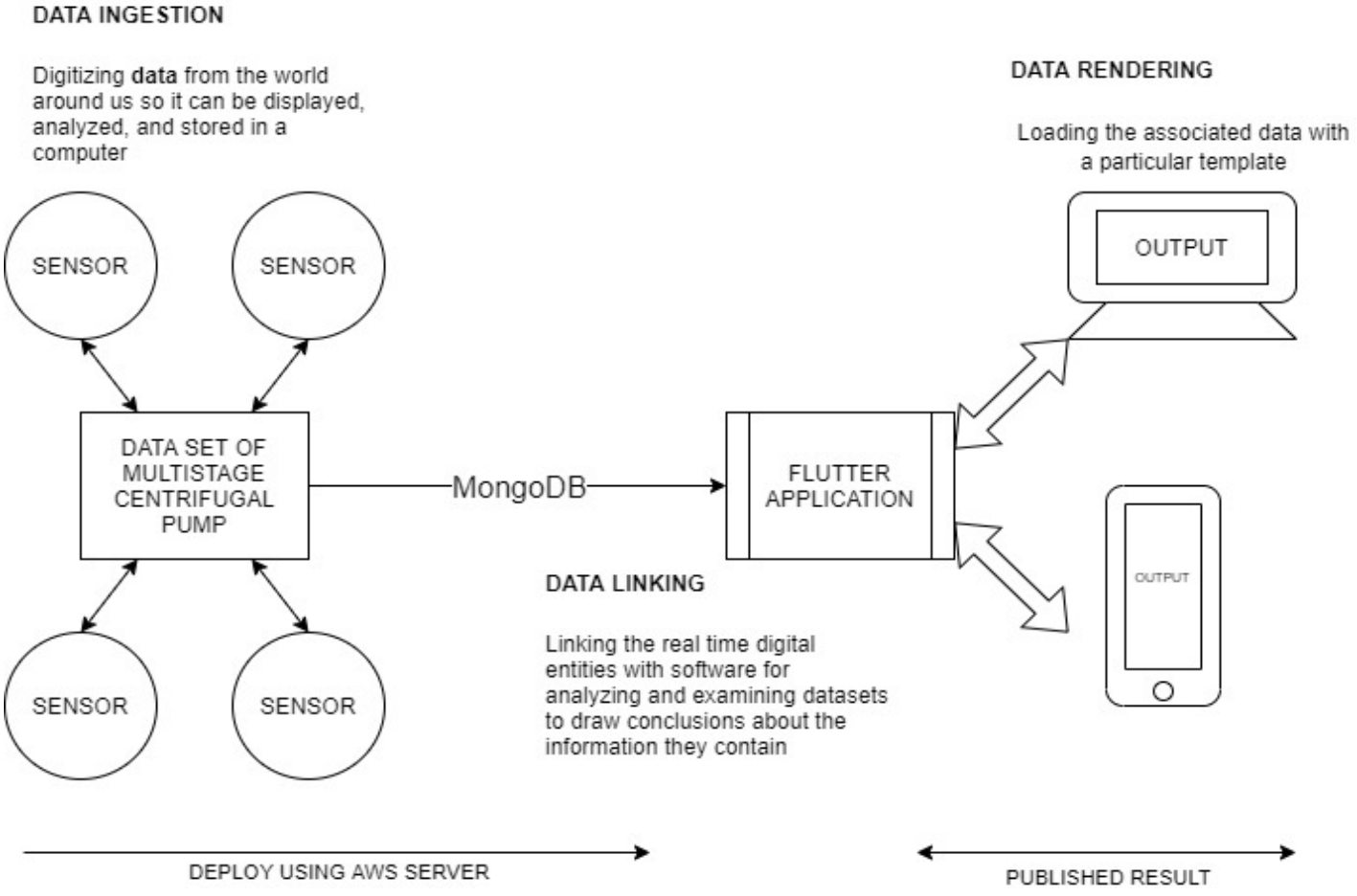
- A digital Triplets is more than a software program, a computer-aided design (CAD) drawing or an equipment database. It is a computer simulation of a physical piping system that can be used throughout the life of the facility. Four key elements are:
    - A mathematical model describing the system.
    - Industry-specific data inserted into the model to describe how the system is built.
    - Measured values from plant instrumentation to simulate the system's operation and validate the calculated results.
    - Methods to analyze those results to gain insights into the operation of the physical piping systems.

- The mathematical model is the foundation of the digital Triplets. For piping systems, it is based on a system's energy balance, which accounts for the change in energy through each item within the system. The next step involves entering the data needed to determine how each item uses the energy. The energy transferred is well understood for each item in the system, including the pumps, pipelines, process equipment and controls.
- Since the digital Triplets is a life cycle document[4], the degree of detail will vary over the project's life. As the system progresses from the conceptual design to a physical system, more details become available. These details are provided in equipment specifications, manufacturer-supplied test data and the installed plant instrumentation. Once the system is built and the as-built data is entered, the Digital Triplets is ready to simulate how the system will operate.

- Once the system is placed into service, it is operated in order to meet market conditions. By specifying the system's as-operating data and its as-built model, the digital Triplets can simulate the operation of the total system. The as-operating data can either be manually entered or automatically imported from the facility's supervisory control and data acquisition (SCADA) system.
- Measured operating data from the physical system can validate the accuracy of the digital Triplets. For example, if the as-measured values of the installed plant instrumentation match the corresponding results of the digital Triplets, the model is valid and accurate.
- The results of the digital Triplets simulation can provide greater insight into system operation. As I've discussed in previous articles, pump cavitation is harmful to both pumps and systems, yet there is no instrument that can directly measure the presence of cavitation.

CHAPTER 5

BLOCK DIAGRAM



CHAPTER 6

SOFTWARE SPECIFICATIONS

- Visual studio code IDE
- Anaconda 3.7
- Flutter 2.0.1
- Android studio 3.6.3

PROGRAMMING LANGUAGES

- HTML/VueJS
- Python
- Dart

6.1 PYTHON

Python is an interpreted, high-level and general-purpose programming language. Python's design philosophy emphasizes code readability with its notable use of significant indentation. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects.

Python is dynamically-typed and garbage-collected. It supports multiple programming paradigms, including structured (particularly, procedural), object-oriented and functional programming. Python is often described as a "batteries included" language due to its comprehensive standard library.

Python's large standard library, commonly cited as one of its greatest strengths, provides tools suited to many tasks. For Internet-facing applications, many standard formats and protocols such as MIME and HTTP are supported. It includes modules for creating graphical user interfaces, connecting to relational databases, generating pseudorandom numbers, arithmetic with arbitrary-precision decimals, manipulating regular expressions, and unit testing.

6.2 Python and Machine Learning

AI projects differ from traditional software projects. The differences lie in the technology stack, the skills required for an AI-based project, and the necessity of deep research. To implement your AI aspirations, you should use a programming language that is stable, flexible, and has tools available. Python offers all of this, which is why we see lots of Python AI projects today.

From development to deployment and maintenance, Python helps developers be productive and confident about the software they're building. Benefits that make Python the best fit for machine learning and AI-based projects include simplicity and consistency, access to great libraries and frameworks for AI and machine learning (ML), flexibility, platform independence, and a wide community. These add to the overall popularity of the language.

Python offers concise and readable code. While complex algorithms and versatile workflows stand behind machine learning and AI, Python's simplicity allows developers to write reliable systems. Developers get to put all their effort into solving an ML problem instead of focusing on the technical nuances of the language.

Additionally, Python is appealing to many developers as it's easy to learn. Python code is understandable by humans, which makes it easier to build models for machine learning.

Many programmers say that Python is more intuitive than other programming languages. Others point out the many frameworks, libraries, and extensions that simplify the implementation of different functionalities. It's generally accepted that Python is suitable for collaborative implementation when multiple developers are

involved. Since Python is a general-purpose language, it can do a set of complex machine learning tasks and enable you to build prototypes quickly that allow you to test your product for machine learning purposes.

Implementing AI and ML algorithms can be tricky and requires a lot of time. It's vital to have a well-structured and well-tested environment to enable developers to come up with the best coding solutions.

To reduce development time, programmers turn to a number of Python frameworks and libraries. A software library is pre-written code that developers use to solve common programming tasks. Python, with its rich technology stack, has an extensive set of libraries for artificial intelligence and machine learning. Here are some of them:

- Keras, TensorFlow, and Scikit-learn for machine learning
- NumPy for high-performance scientific computing and data analysis
- SciPy for advanced computing
- Pandas for general-purpose data analysis
- Seaborn for data visualization
- Scikit-learn features various classification, regression, and clustering algorithms, including support vector machines, random forests, gradient boosting, k-means, and DBSCAN, and is designed to work with the Python numerical and scientific libraries NumPy and SciPy.

With these solutions, you can develop your product faster. Your development team won't have to reinvent the wheel and can use an existing library to implement necessary features.

6.3 ANACONDA

Anaconda is a distribution of the Python and R programming languages for scientific computing (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify package management and deployment.

Package versions in Anaconda are managed by the package management system conda. This package manager was spun out as a separate open-source package as it ended up being useful on its own and for other things than Python. There is also a small, bootstrap version of Anaconda called Miniconda, which includes only conda, Python, the packages they depend on, and a small number of other packages.

Anaconda Navigator is a desktop graphical user interface (GUI) included in Anaconda distribution that allows users to launch applications and manage conda packages, environments and channels without using command-line commands. Navigators can search for packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux.

The following applications are available by default in Navigator:

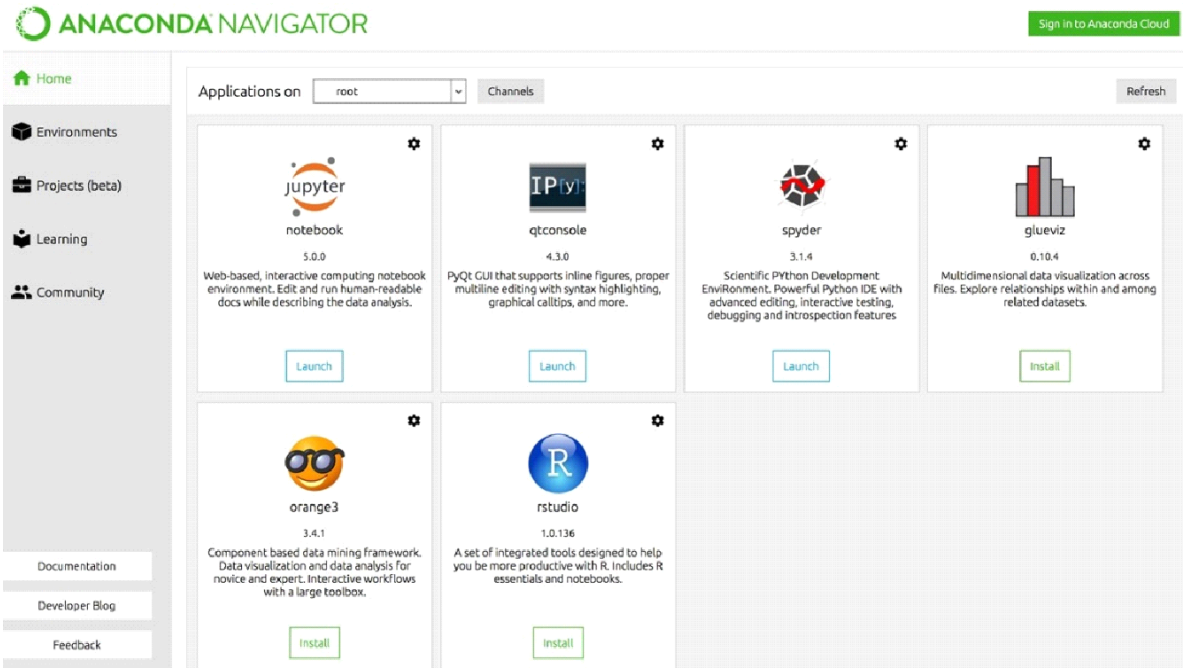


Fig.6.1 Anaconda Navigator

- Jupyter Notebook
- QtConsole
- Spyde
- Orange
- RStudio
- Visual Studio Code

6.4 ANDROID STUDIO

Android Studio is the official integrated development environment (IDE) for Google's Android operating system, built on JetBrains' IntelliJ IDEA software and designed specifically for Android development. It is available for download on

Windows, macOS and Linux based operating systems or as a subscription-based service in 2020. It is a replacement for the Eclipse Android Development Tools (E-ADT) as the primary IDE for native Android application development.

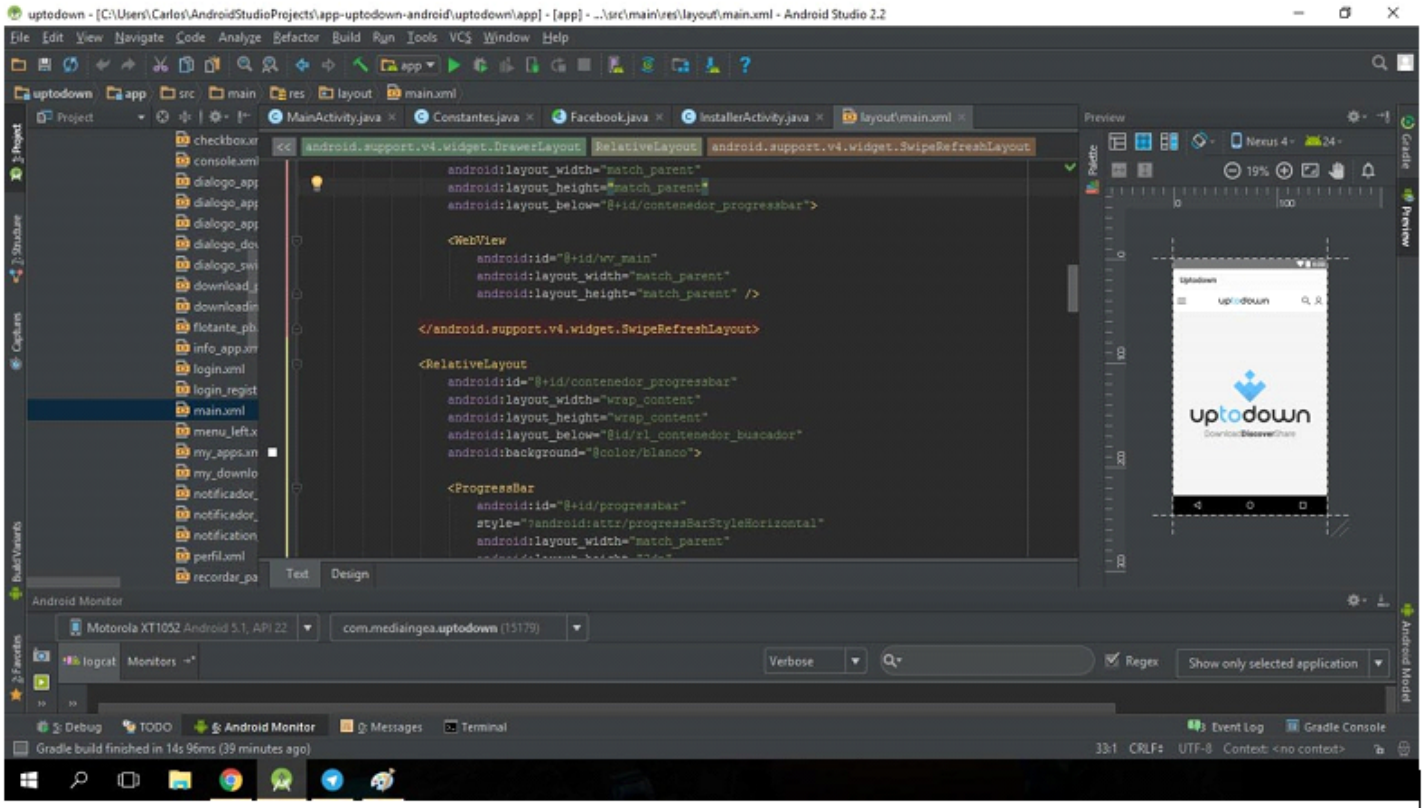


Fig.6.2 Android Studio Works Environment

The specific feature of the Android Studio is an absence of the possibility to switch autosave features off.

The following features are provided in the current stable version:

- Gradle-based build support
- Android-specific refactoring and quick fixes
- Lint tools to catch performance, usability, version compatibility and other problems
- Pro-Guard integration and app-signing capabilities

- Template-based wizards to create common Android designs and components
- A rich layout editor that allows users to drag-and-drop UI components, option to preview layouts on multiple screen configurations
- Support for building Android Wear apps
- Built-in support for Google Cloud Platform, enabling integration with Firebase Cloud Messaging (Earlier 'Google Cloud Messaging') and Google App Engine
- Android Virtual Device (Emulator) to run and debug apps in the Android studio.

Android Studio supports all the same programming languages of IntelliJ (and CLion) e.g. Java, C++, and more with extensions, such as Go; and Android Studio 3.0 or later supports Kotlin and "all Java 7 language features and a subset of Java 8 language features that vary by platform version. "External projects backport some Java 9 features. While IntelliJ states that Android Studio supports all released Java versions, and Java 12, it's not clear to what level Android Studio supports Java versions up to Java 12 (the documentation mentions partial Java 8 support). At least some new language features up to Java 12 are usable in Android.Once an app has been compiled with Android Studio, it can be published on the Google Play Store. The application has to be in line with the Google Play Store developer content policy.

6.4.1 Features of Android Studio

Write better code, work faster, and be more productive with an intelligent code editor that provides code completion for Kotlin, Java, and C/C++ languages.

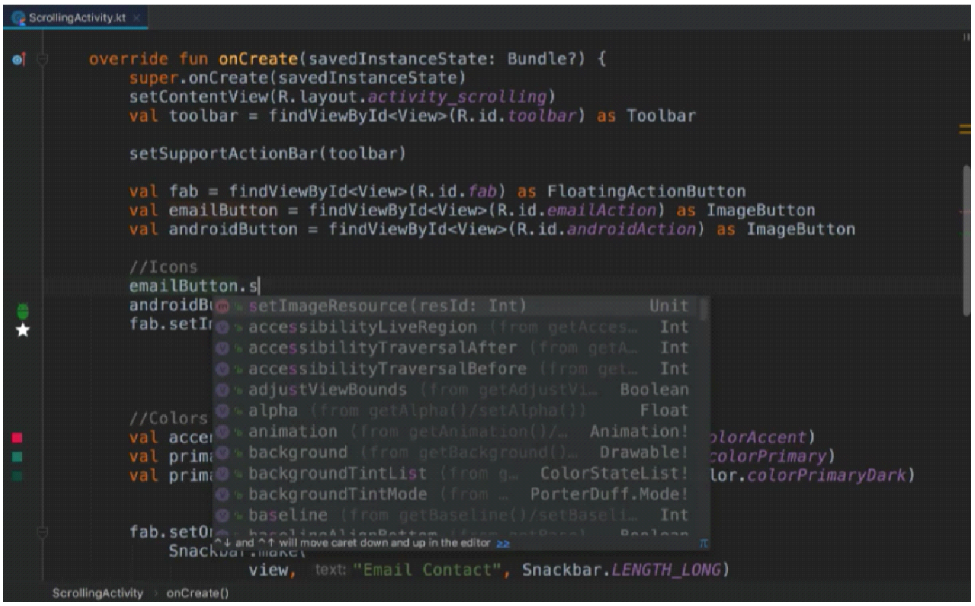


Fig.6.3. Intelligent Code Editor

6.4.2 Visual layout editor

Create complex layouts with **ConstraintLayout** by adding constraints from each view to other views and guidelines. Then preview your layout on any screen size by selecting one of various device configurations or by simply resizing the preview window.

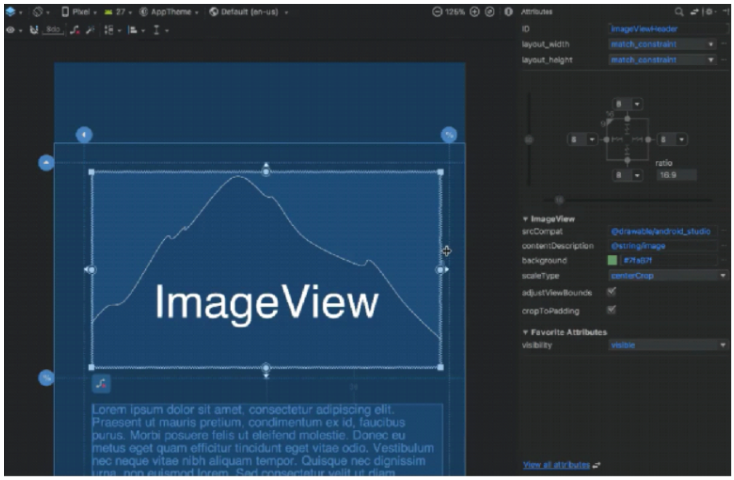


Fig.6.4. Visual Layout Editor

6.4.3 Android Emulator

Install and run your apps faster than with a physical device and simulate different configurations and features, including ARCore, Google's platform for building augmented reality experiences.

Fig.6.5. Android Editor

6.4.4 APK Analyzer

Find opportunities to reduce your Android app size by inspecting the contents of your app APK file, even if it wasn't built with Android Studio. Inspect the manifest file, resources, and DEX files. Compare two APKs to see how your app size changed between app versions.

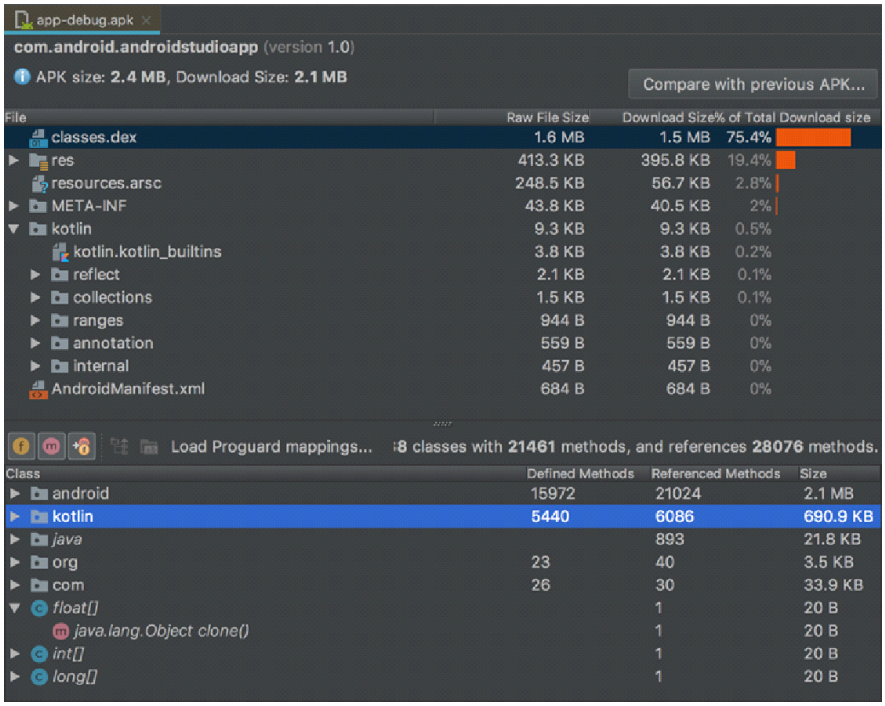


Fig.6.6. APK Analyzer

6.4.5 Flexible build system

Powered by Gradle, Android Studio's build system allows you to customize your build to generate multiple build variants for different devices from a single project.

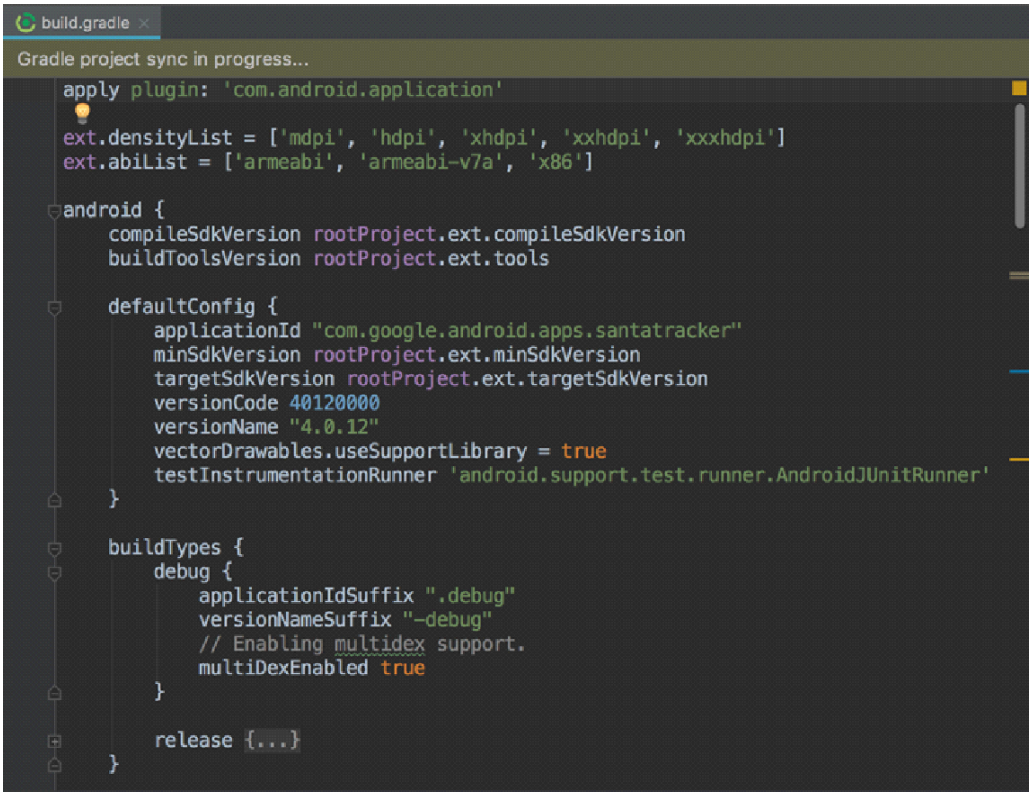


Fig.6.7. Flexible Build System

6.5 Realtime profilers

The built-in profiling tools provide real-time statistics for your app's CPU, memory, and network activity. Identify performance bottlenecks by recording method traces, inspecting the heap and allocations, and see incoming and outgoing network payloads.

Fig.6.8. Realtime Profilers

6.6 MACHINE LEARNING

Machine learning is an application of artificial intelligence (AI) that provides systems the ability to automatically learn and improve from experience without being explicitly programmed. Machine learning focuses on the development of computer programs that can access data and use it to learn for themselves.

The process of learning begins with observations or data, such as examples, direct experience, or instruction, in order to look for patterns in data and make better decisions in the future based on the examples that we provide. The primary aim is to allow the computers to learn automatically without human intervention or assistance and adjust actions accordingly.

But, using the classic algorithms of machine learning, text is considered as a sequence of keywords; instead, an approach based on semantic analysis mimics the human ability to understand the meaning of a text.

6.6.1 Some Machine Learning Methods

Machine learning algorithms are often categorized as **SUPERVISED** or **UNSUPERVISED**.

- Supervised machine learning algorithms can apply what has been learned in the past to new data using labeled examples to predict future events. Starting from the analysis of a known training dataset, the learning algorithm produces an inferred function to make predictions about the output values. The system is able to provide targets for any new input after sufficient training. The learning algorithm can also compare its output with the correct, intended output and find errors in order to modify the model accordingly.

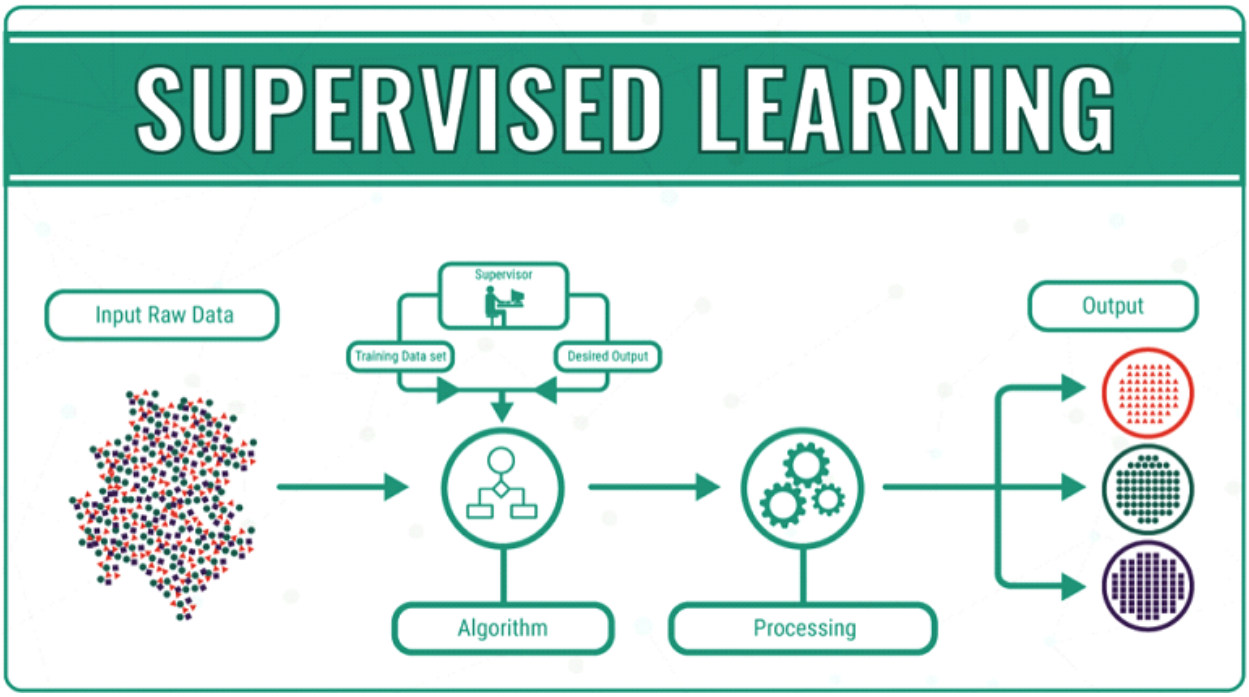


Fig.6.9. Supervised Learning

- In contrast, unsupervised machine learning algorithms are used when the information used to train is neither classified nor labeled. Unsupervised learning studies how systems can infer a function to describe a hidden structure from unlabeled data. The system doesn't figure out the right output,

but it explores the data and can draw inferences from datasets to describe hidden structures from unlabeled data.

- Semi-supervised machine learning algorithms fall somewhere in between supervised and unsupervised learning, since they use both labeled and unlabeled data for training – typically a small amount of labeled data and a large amount of unlabeled data. The systems that use this method are able to considerably improve learning accuracy. Usually, semi-supervised learning is chosen when the acquired labeled data requires skilled and relevant resources in order to train it / learn from it. Otherwise, acquiring unlabeled data generally doesn't require additional resources.
- Reinforcement machine learning algorithms is a learning method that interacts with its environment by producing actions and discovers errors or rewards. Trial and error search and delayed reward are the most relevant characteristics of reinforcement learning. This method allows machines and software agents to automatically determine the ideal behavior within a specific context in order to maximize its performance. Simple reward feedback is required for the agent to learn which action is best; this is known as the reinforcement signal.

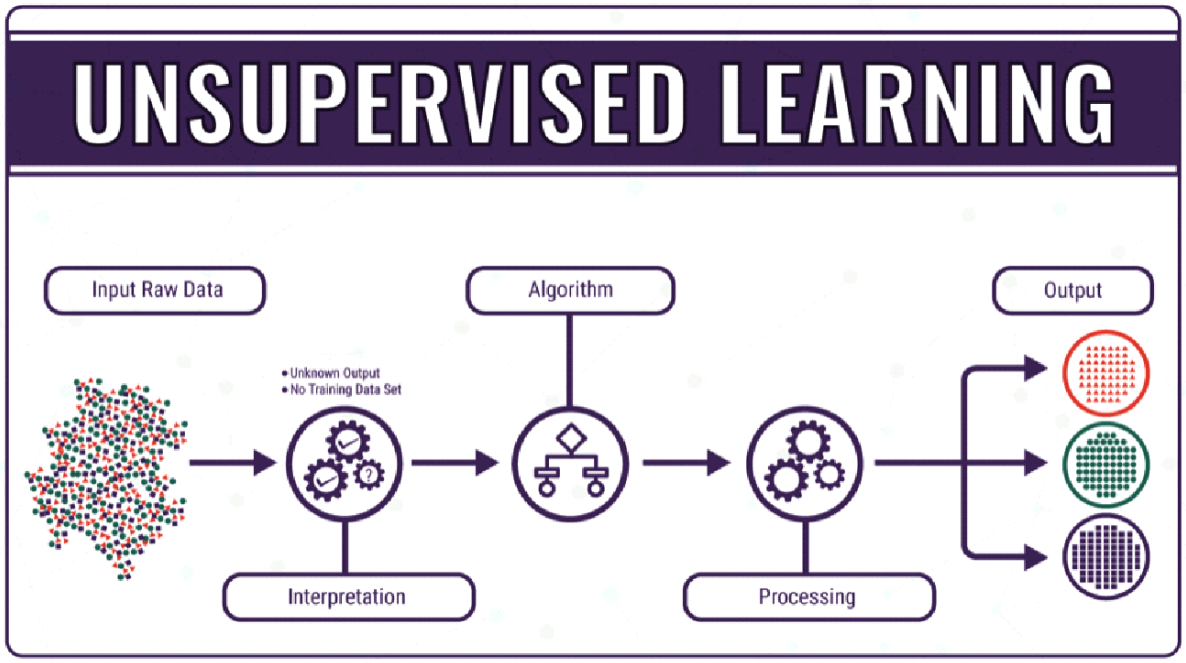


Fig.6.10. Unsupervised Learning

Machine learning enables analysis of massive quantities of data. While it generally delivers faster, more accurate results in order to identify profitable opportunities or dangerous risks, it may also require additional time and resources to train it properly. Combining machine learning with AI and cognitive technologies can make it even more effective in processing large volumes of information.

TYPES OF MACHINE LEARNING ALGORITHMS:

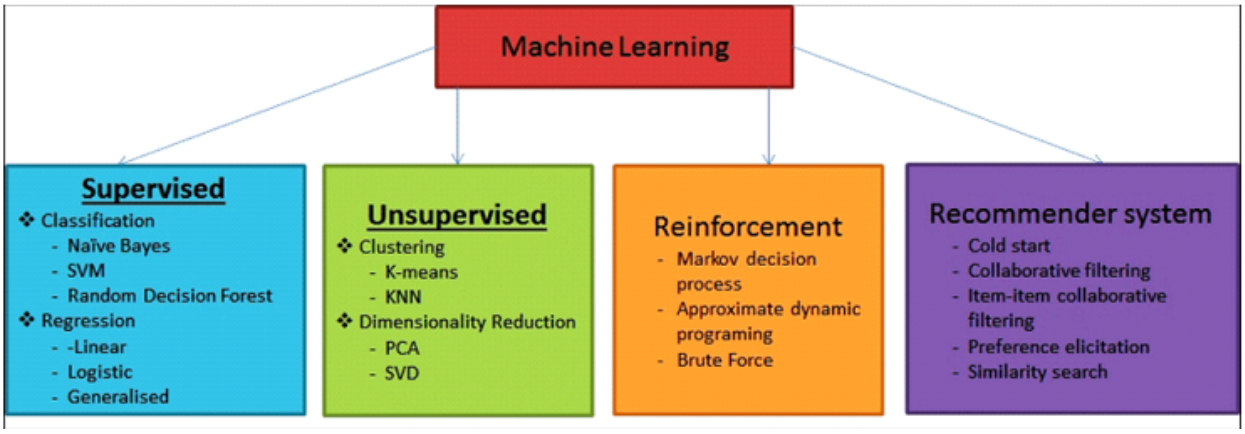


Fig.6.11. Machine Learning

So, we will be using **Logistic Regression** method which is a type **of Supervised Machine Learning Algorithm**

SVM Classification

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In the SVM algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiates the two classes very well

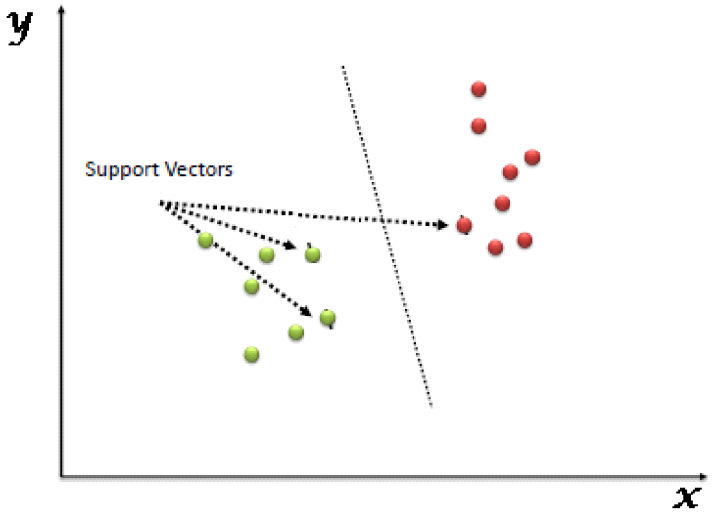


Fig.6.12. Graph 1

Support Vectors are simply the co-ordinates of individual observation. The SVM classifier is a frontier which best segregates the two classes (hyper-plane/ line).

Identify the right hyper-plane (Scenario-1): Here, we have three hyper-planes (A, B and C). Now, identify the right hyper-plane to classify star and circle.

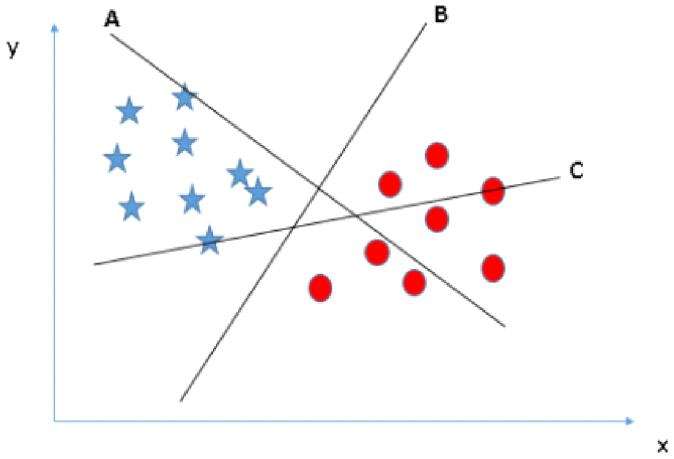


Fig.6.13. Graph 2

You need to remember a thumb rule to identify the right hyper-plane: “Select the hyper-plane which segregates the two classes better”. In this scenario, hyper-plane “B” has excellently performed this job.

Identify the right hyper-plane (Scenario-2): Here, we have three hyper-planes (A, B and C) and all are segregating the classes well. Now, how can we identify the right hyper-plane?

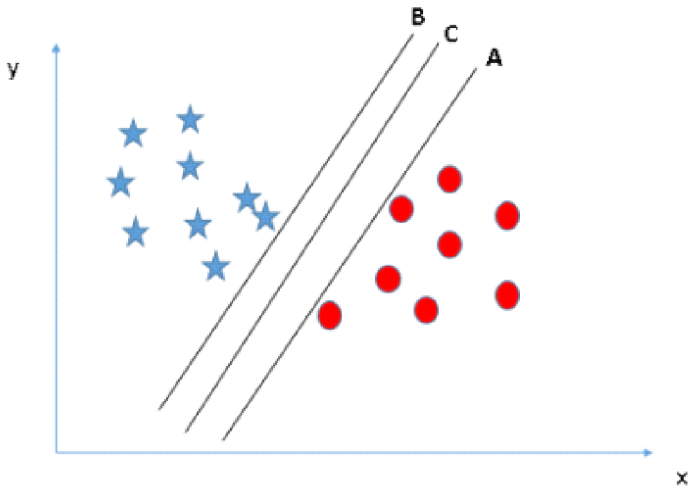


Fig.6.14. Graph 3

Here, maximizing the distances between the nearest data point (either class) and hyper-plane will help us to decide the right hyper-plane. This distance is called Margin.

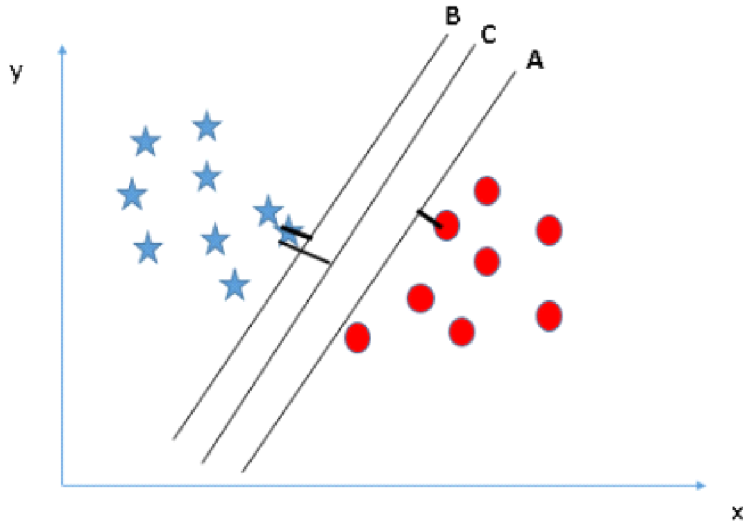


Fig.6.15. Graph 4

Above, you can see that the margin for hyper-plane C is high as compared to both A and B. Hence, we name the right hyper-plane as C. Another lightning reason for selecting the hyper-plane with higher margin is robustness. If we select a hyper-plane having low margin then there is a high chance of miss-classification.

Identify the right hyper-plane (Scenario-3):

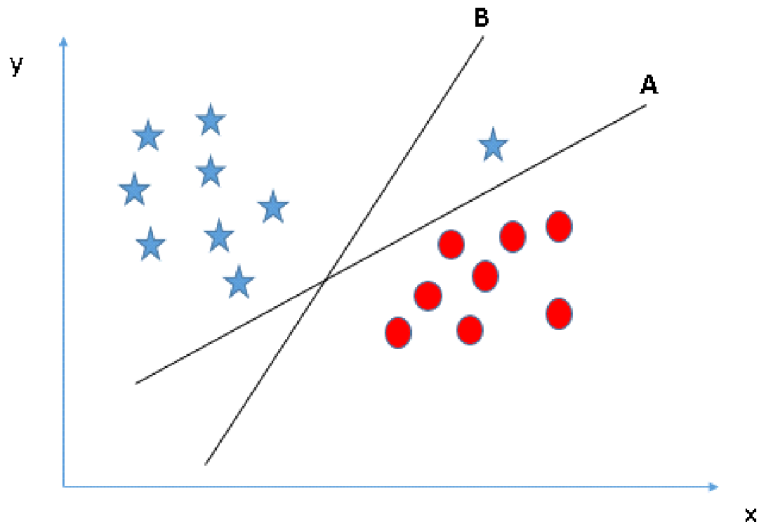


Fig.6.16.Graph 5

Some of you may have selected the hyper-plane B as it has higher margin compared to A. But here is the catch, SVM selects the hyper-plane which classifies the classes accurately prior to maximizing margin. Here, hyper-plane B has a classification error and A has classified all correctly. Therefore, the right hyper-plane is A.

Can we classify two classes Below, I am unable to segregate the two classes using a straight line, as one of the stars lies in the territory of another(circle) class as an outlier.

As I have already mentioned, one star at another end is like an outlier for star class. The SVM algorithm has a feature to ignore outliers and find the hyper-plane that has the maximum margin. Hence, we can say, SVM classification is robust to outliers.

Find the hyper-plane to segregate to classes (Scenario-5): In the scenario below, we can't have a linear hyper-plane between the two classes, so how does SVM classify these two classes? Till now, we have only looked at the linear hyper-plane.

SVM can solve this problem. Easily! It solves this problem by introducing additional features. Here, we will add a new feature z=x^2+y^2. Now, let's plot the data points on axis x and z:

In above plot, points to consider are: All values for z would be positive always because z is the squared sum of both x and y

In the original plot, red circles appear close to the origin of x and y axes, leading to lower value of z and star relatively away from the origin result to higher value of z.

In the SVM classifier, it is easy to have a linear hyper-plane between these two classes. But, another burning question which arises is, should we need to add this feature manually to have a hyper-plane. No, the SVM algorithm has a technique called the kernel trick. The SVM kernel is a function that takes low dimensional input space and transforms it to a higher dimensional space i.e. it converts not separable problem to separable problem. It is mostly useful in non-linear separation problems. Simply put, it does some extremely complex data transformations, then finds out the process to separate the data based on the labels or outputs you've defined.

When we look at the hyper-plane in original input space it looks like a circle:

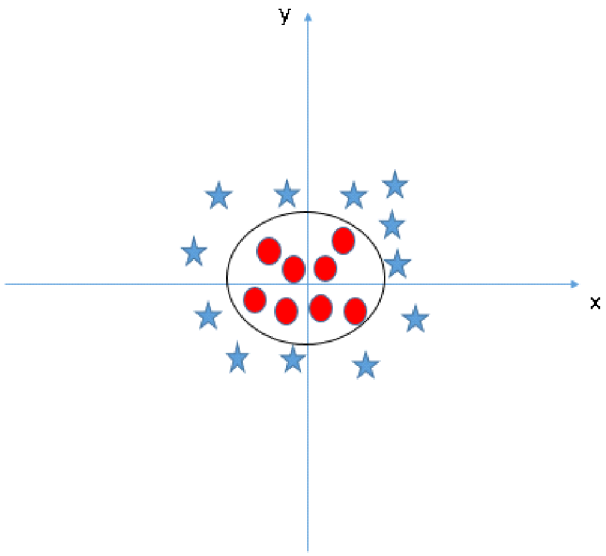


Fig.6.17. Graph 6

Now, let's look at the methods to apply the SVM classifier algorithm in a data science challenge.

How to implement SVM in Python?

In Python, scikit-learn is a widely used library for implementing machine learning algorithms. SVM is also available in the scikit-learn library and we follow the same structure for using it(Import library, object creation, fitting model and prediction).

6.6.2 LOGISTIC REGRESSION

Logistic regression is a supervised learning classification algorithm used to predict the probability of a target variable. The nature of the target or dependent variable is dichotomous, which means there would be only two possible classes.

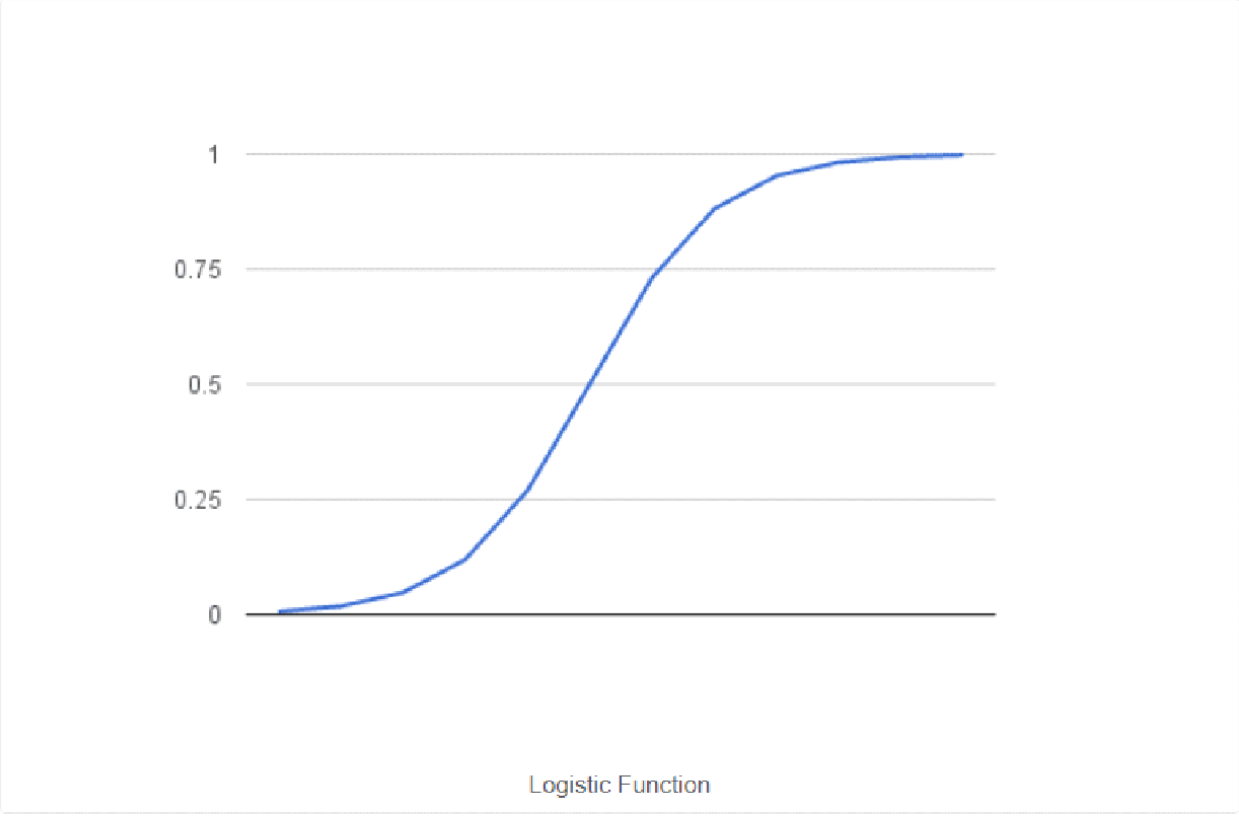
Logistic Function

Logistic regression is named for the function used at the core of the method, the logistic function.

The logistic function, also called the sigmoid function was developed by statisticians to describe properties of population growth in ecology, rising quickly and maxing out at the carrying capacity of the environment. It's an S-shaped curve that can take any real-valued number and map it into a value between 0 and 1, but never exactly at those limits.

1 / (1 + e^-value)

Where e is the base of the natural logarithms (Euler's number or the EXP () function in your spreadsheet) and value is the actual numerical value that you want to transform. Below is a plot of the numbers between -5 and 5 transformed into the range 0 and 1 using the logistic function.



. **Fig.6.18.**Graph 7

Now that we know what the logistic function is, let's see how it is used in logistic regression.

Representation Used for Logistic Regression

Logistic regression uses an equation as the representation, very much like linear regression.

Input values (x) are combined linearly using weights or coefficient values (referred to as the Greek capital letter Beta) to predict an output value (y). A key difference from linear regression is that the output value being modeled is a binary value (0 or 1) rather than a numeric value.

Below is an example logistic regression equation:

y = e^ (b0 + b1*x) / (1 + e^ (b0 + b1*x))

Where y is the predicted output, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each column in your input data has an associated b coefficient (a constant real value) that must be learned from your training data.

The actual representation of the model that you would store in memory or in a file are the coefficients in the equation (the beta value or b's).

Logistic Regression Predicted Probabilities (Technical Interlude)

Logistic regression models the probability of the default class (e.g. the first class).

For example, if we are modeling people's sex as male or female from their height, then the first class could be male and the logistic regression model could be written as the probability of male given a person's height, or more formally:

P(sex=male|height)

Written another way, we are modeling the probability that an input (X) belongs to the default class (Y=1), we can write this formally as:

P(X) = P(Y=1|X)

We're predicting probabilities? I thought logistic regression was a classification algorithm?

Note that the probability prediction must be transformed into a binary value (0 or 1) in order to actually make a probability prediction. More on this later when we talk about making predictions.

Logistic regression is a linear method, but the predictions are transformed using the logistic function. The impact of this is that we can no longer understand the predictions as a linear combination of the inputs as we can with linear regression, for example, continuing on from above, the model can be stated as:

p(X) = e^ (b0 + b1*X) / (1 + e^ (b0 + b1*X))

I don't want to dive into the math too much, but we can turn around the above equation as follows (remember we can remove the e from one side by adding a natural logarithm (ln) to the other):

ln(p(X) / 1 – p(X)) = b0 + b1 * X

This is useful because we can see that the calculation of the output on the right is linear again (just like linear regression), and the input on the left is a log of the probability of the default class.

This ratio on the left is called the odds of the default class (it's historical that we use odds, for example, odds are used in horse racing rather than probabilities). Odds are calculated as a ratio of the probability of the event divided by the probability of not the event, e.g. 0.8/ (1-0.8) which has the odds of 4. So, we could instead write:

ln(odds) = b0 + b1 * X

Because the odds are log transformed, we call this left-hand side the log-odds or the probity. It is possible to use other types of functions for the transform (which is out of scope_, but as such it is common to refer to the transform that relates the linear regression equation to the probabilities as the link function, e.g. the probit link function.

We can move the exponent back to the right and write it as:

odds = e^ (b0 + b1 * X)

All of this helps us understand that indeed the model is still a linear combination of the inputs, but that this linear combination relates to the log-odds of the default class.

Learning the Logistic Regression Model

The coefficients (Beta values b) of the logistic regression algorithm must be estimated from your training data. This is done using maximum-likelihood estimation.

Maximum-likelihood estimation is a common learning algorithm used by a variety of machine learning algorithms, although it does make assumptions about the distribution of your data (more on this when we talk about preparing your data).

The best coefficients would result in a model that would predict a value very close to 1 (e.g. male) for the default class and a value very close to 0 (e.g. female) for the other class. The intuition for maximum-likelihood for logistic regression is that a search procedure seeks values for the coefficients (Beta values) that minimize the error in the probabilities predicted by the model to those in the data (e.g. probability of 1 if the data is the primary class).

We are not going to go into the math of maximum likelihood. It is enough to say that a minimization algorithm is used to optimize the best values for the coefficients for your training data. This is often implemented in practice using efficient numerical optimization algorithms (like the Quasi-newton method).

When you are learning logistic, you can implement it yourself from scratch using the much simpler gradient descent algorithm.

Making Predictions with Logistic Regression

Making predictions with a logistic regression model is as simple as plugging in numbers into the logistic regression equation and calculating a result.

Let's make this concrete with a specific example.

Let's say we have a model that can predict whether a person is male or female based on their height (completely fictitious). Given a height of 150cm is the person male or female.

We have learned the coefficients of b0 = -100 and b1 = 0.6. Using the equation above we can calculate the probability of male given a height of 150cm or more formally P(male|height=150). We will use EXP () for e, because that is what you can use if you type this example into your spreadsheet:

y = e^ (b0 + b1*X) / (1 + e^ (b0 + b1*X))

y = exp (-100 + 0.6*150) / (1 + EXP (-100 + 0.6*X))

y = 0.0000453978687

Or a probability of near zero that the person is a male.

In practice we can use the probabilities directly. Because this is classification and we want a crisp answer, we can snap the probabilities to a binary class value, for example:

0 if p(male) < 0.5

1 if p(male) >= 0.5

Now that we know how to make predictions using logistic regression, let's look at how we can prepare our data to get the most from the technique.

Prepare Data for Logistic Regression

The assumptions made by logistic regression about the distribution and relationships in your data are much the same as the assumptions made in linear regression.

Much study has gone into defining these assumptions and precise probabilistic and statistical language is used. My advice is to use these as guidelines or rules of thumb and experiment with different data preparation schemes.

Ultimately in predictive modeling machine learning projects you are laser focused on making accurate predictions rather than interpreting the results. As such, you can break some assumptions as long as the model is robust and performs well.

- **Binary Output Variable:** This might be obvious as we have already mentioned it, but logistic regression is intended for binary (two-class) classification problems. It will predict the probability of an instance belonging to the default class, which can be snapped into a 0 or 1 classification.
- **Remove Noise:** Logistic regression assumes no error in the output variable (y), consider removing outliers and possibly misclassified instances from your training data.
- **Gaussian Distribution:** Logistic regression is a linear algorithm (with a non-linear transform on output). It does assume a linear relationship between the input variables with the output. Data transforms of your input variables that better expose this linear relationship can result in a more accurate model. For example, you can use log, root, Box-Cox and other univariate transforms to better expose this relationship.

- **Remove Correlated Inputs:** Like linear regression, the model can overfit if you have multiple highly-correlated inputs. Consider calculating the pairwise correlations between all inputs and removing highly correlated inputs.
- **Fail to Converge:** It is possible for the expected likelihood estimation process that learns the coefficients to fail to converge. This can happen if there are many highly correlated inputs in your data or the data is very sparse (e.g. lots of zeros in your input data).

6.7 FLUTTER (UI Software Development Kit)

Flutter is an open-source UI software development kit created by Google. It is used to develop applications for Android, iOS, Linux, Mac, Windows, Google Fuchsia, and the web from a single codebase.

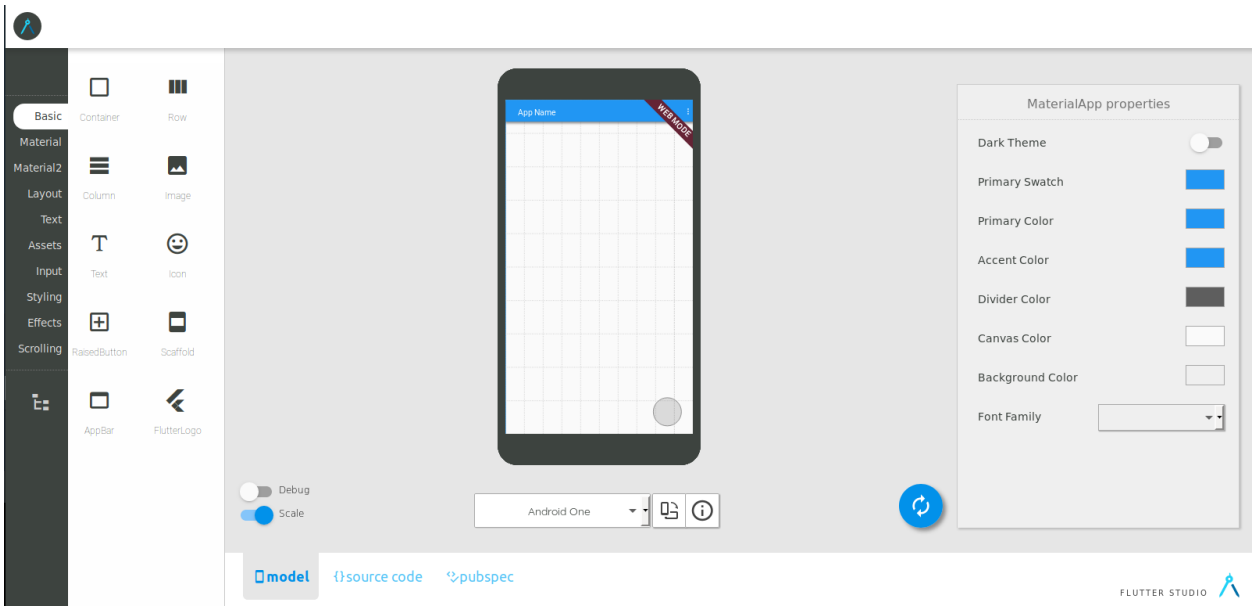


Fig.6.19.Flutter

6.7.1 Architecture of Flutter

Dart Platform

Flutter apps are written in the Dart language and make use of many of the language's more advanced features.

On Windows, macOS, and Linux Flutter runs in the Dart virtual machine, which features a just-in-time execution engine. While writing and debugging an app, Flutter uses Just in Time compilation, allowing for "hot reload", with which modifications to source files can be injected into a running application. Flutter extends this with support for stateful hot reload, where in most cases changes to source code are reflected immediately in the running app without requiring a restart or any loss of state.

Release versions of Flutter apps are compiled with a head-of-time (AOT) compilation on both Android and iOS, making Flutter's high performance on mobile devices possible.

Flutter Engine

Flutter's engine, written primarily in C++, provides low-level rendering support using Google's Sika graphics library. Additionally, it interfaces with platform-specific SDKs such as those provided by Android and iOS. The Flutter Engine is a portable runtime for hosting Flutter applications. It implements Flutter's core libraries, including animation and graphics, file and network I/O, accessibility support, plugin architecture, and a Dart runtime and compile toolchain. Most developers interact with Flutter via the Flutter Framework, which provides a reactive framework and a set of platform, layout, and foundation widgets.

Foundation Library

The Foundation library, written in Dart, provides basic classes and functions that are used to construct applications using Flutter, such as APIs to communicate with the engine.

Design Specific Widgets

The Flutter framework contains two sets of widgets that conform to specific design languages: Material Design widgets implement Google's design language of the same name, and Cupertino widgets implement Apple's iOS Human interface guidelines.

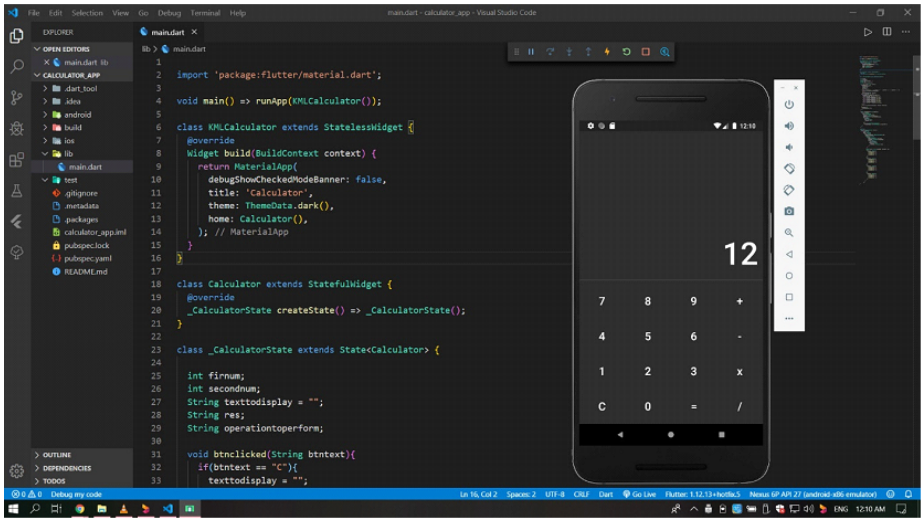


Fig.6.20 The output of a flutter code

6.8 Visual Studio Code IDE

Visual Studio Code is a freeware source-code editor made by Microsoft for Windows, Linux and macOS. Features include support for debugging, syntax highlighting, intelligent code completion, snippets, code refactoring, and embedded Git. Users can change the theme, keyboard shortcuts, preferences, and install extensions that add additional functionality.

Features

Visual Studio Code is a source-code editor that can be used with a variety of programming languages, including Java, JavaScript, Go, Node.js, Python and C++. It is based on the Electron framework, which is used to develop Node.js Web applications that run on the Blink layout engine. Visual Studio Code employs the same editor component (codenamed "Monaco") used in Azure DevOps (formerly called Visual Studio Online and Visual Studio Team Services).

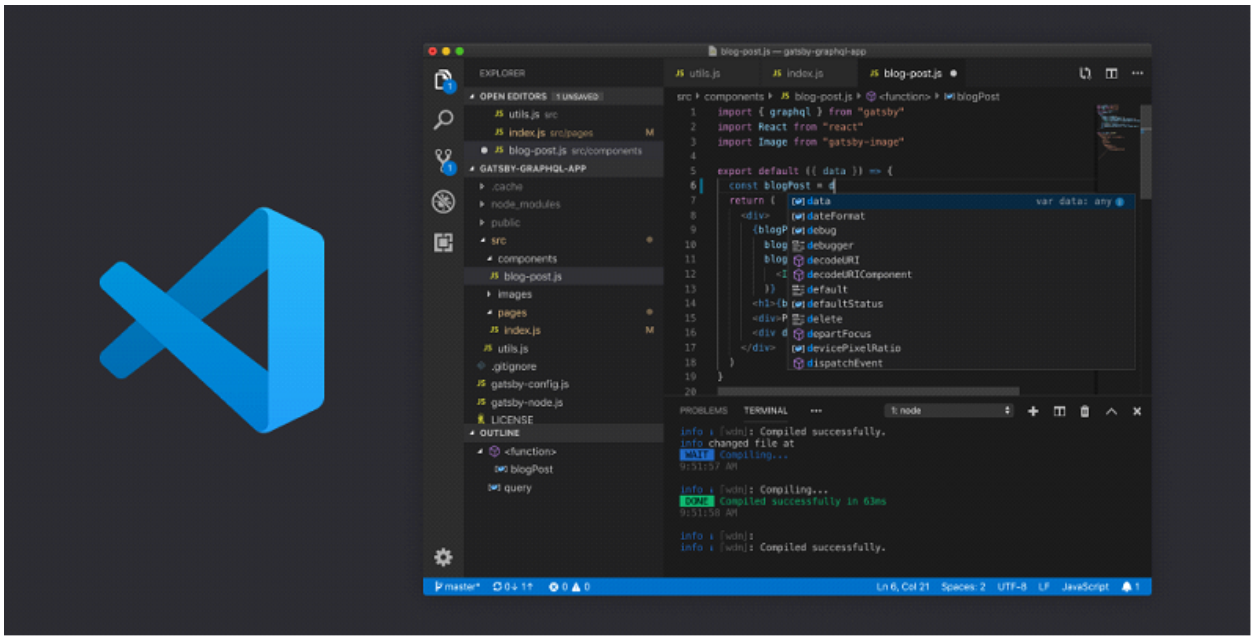


Fig.6.21. Visual studio Code editor

Instead of a project system, it allows users to open one or more directories, which can then be saved in workspaces for future reuse. This allows it to operate as a language-agnostic code editor for any language. It supports a number of programming languages and a set of features that differs per language. Unwanted files and folders can be excluded from the project tree via the settings. Many Visual Studio Code features are not exposed through menus or the user interface but can be accessed via the command palette.

Visual Studio Code can be extended via extensions, available through a central repository. This includes additions to the editor and language support. A notable feature is the ability to create extensions that add support for new languages, themes, and debuggers, perform static code analysis, and add code linters using the Language Server Protocol.

Visual Studio Code includes multiple extensions for FTP, allowing the software to be used as a free alternative for web development. Code can be synced between the editor and the server, without downloading any extra software.

Visual Studio Code allows users to set the code page in which the active document is saved, the newline character, and the programming language of the active document. This allows it to be used on any platform, in any locale, and for any given programming language.

Language support

Out-of-the-box, Visual Studio Code includes basic support for most common programming languages. This basic support includes syntax highlighting, bracket matching, code folding, and configurable snippets. Visual Studio Code also ships with IntelliSense for JavaScript, TypeScript, JSON, CSS, and HTML, as well as debugging support for Node.js. Support for additional languages can be provided by freely available extensions on the VS Code Marketplace.

Data collection

Visual Studio Code collects usage data and sends it to Microsoft, although this can be disabled.[26] In addition, because of the open-source nature of the application, the telemetry code is accessible to the public, who can see exactly what is collected. According to Microsoft, the data is shared with Microsoft-controlled affiliates and subsidiaries, although law enforcement may request it as part of a legal process.

Version control

Source control is a built-in feature of Visual Studio Code. It has a dedicated tab inside of the menu bar where you can access version control settings and view changes made to the current project. To use the feature, you must link Visual Studio Code to any supported version control system (Git, Subversion, Perforce, etc.). This allows you to create repositories as well as make push and pull requests directly from the Visual Studio Code program.

Dart Code

Dart is a client-optimized programming language for apps on multiple platforms. It is developed by Google and is used to build mobile, desktop, server, and web applications.

The Dart usage can be in four ways as mentioned below:

Compiled as JavaScript

To run in mainstream web browsers, Dart relies on a source-to-source compiler to JavaScript. According to the project site, Dart was "designed to be easy to write development tools for, well-suited to modern app development, and capable of high-performance implementations. "When running Dart code in a web browser the code is precompiled into JavaScript using the dart2js compiler. Compiled as JavaScript, Dart code is compatible with all major browsers with no need for browsers to adopt Dart. Through optimizing the compiled JavaScript output to avoid expensive checks and operations, code written in Dart can, in some cases, run faster than equivalent code hand-written using JavaScript idioms.

Stand-alone

The Dart software development kit (SDK) ships with a stand-alone Dart VM, allowing Dart code to run in a command-line interface environment. As the language tools included in the Dart SDK are written mostly in Dart, the stand-alone

Dart VM is a critical part of the SDK. These tools include the dart2js compiler and a package manager called pub. Dart ships with a complete standard library allowing users to write fully working system apps, such as custom web servers.

Ahead-of-time compiled

Dart code can be AOT-compiled into machine code (native instruction sets). Apps built with Flutter, a mobile app SDK built with Dart, are deployed to app stores as AOT-compiled Dart code.

Native

Dart 2.6 with dart2native compiler to compile to self-contained, native executables code. Before Dart 2.6, this feature only exposed this capability on iOS and Android mobile devices via Flutter.

```
  Widget build(BuildContext context) {
    return new AlertDialog(
      title: const Text('Not Implemented'),
      content: const Text('This feature has not yet been implemented.'),
      actions: <Widget>[
        new FlatButton(
          onPressed: debugDumpApp,
          child: new Row(
            children: <Widget>[
              const Icon(
                Icons.dvr,
                size: 18.0,
              ),
              new Container(
                width: 8.0,
              ),
              const Text('DUMP APP TO CONSOLE'),
            ],
          ),
        ),
        new FlatButton(
          onPressed: () {
            Navigator.pop(context, false);
          },
          child: const Text('OH WELL'),
        ),
      ],
    );
  }
}
```

Fig.6.22 Dart Code

CHAPTER 7

WORKING

The Digital Triplet application consists of 5 components each having specific importance.

- About
- Dataset
- Machine Status
- ML Prediction
- Data Analysis

7.1 About:

The about tab consists of specifications and details related to the pump proposed in video format. This ‘About’ video helps the user to understand the application and details of Multistage Centrifugal Pump in a digital triplet app which can be used both in Android and IOS.

Fig.7.1. About

7.2 Dataset:

The data are from all available sensors, all of them are raw value. Total sensors are 52 units. This dataset contains 3 main group of data

1. Timestamp data
2. Sensor data(52 series): All values are raw values
3. Machine status: This is target label that I want to predict when the failure will happen

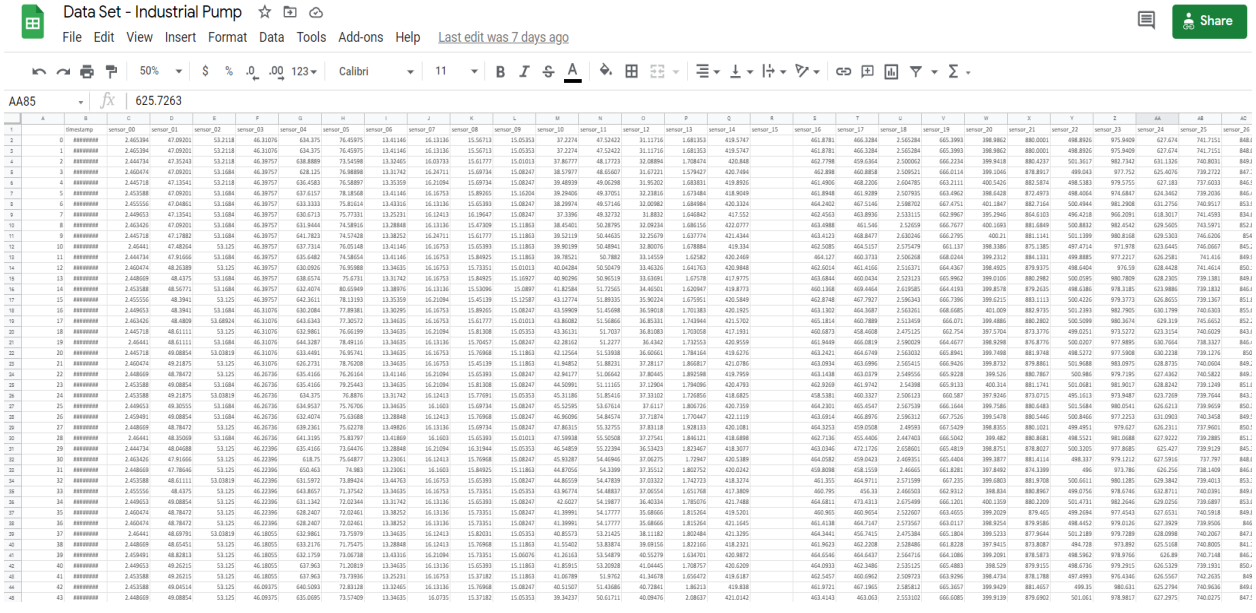


Fig.7.2.Data Set

7.3 Machine Status:

```
<!DOCTYPE HTML>

<html lang="en">

<head>

  <meta charset="UTF-8">

  <title>Digital Triplet - Live Sensor Output</title>


  <meta name="viewport" content="width=device-width, user-scalable=no">

  <link rel="stylesheet" href="fv.css" type="text/css" />
```

```
</head>

<body>

  <div id='wrap'>

    <div class='options'>

    </div>

    <h1 title='how forms should be done.'>ENTER THE LIVE SENSORS INPUT</h1>

                                                                                <img
src="https://www.flowmorepumps.com/flow-admin/assets/ckeditor/kcfinder/upload/images/MULTISTA
GE-PUMP.jpg" width="800" height="500" class="center">

    <style>

      img {

        display: block;

        margin-left: auto;

        margin-right: auto;

      }

      </style>

    <section class='form'>

      <form action="" method="post" novalidate>

        <fieldset>

          <div class="field">

            <label>

              <span>Date</span>

              <input class='date' type="date" name="date" required='required'>

            </label>
```

```
            </div>
            <div class="field">
              <label>
                <span>Time</span>
                <input class='time' type="time" name="time" required='required'>
                <div class='tooltip help'>
                <span>?</span>
                <div class='content'>
                  <b></b>
                  <p>Format should be: XX:XX</p>
                </div>
              </div>
              </label>
            </div>
            <div class="field">
              <label>
                <span>Sensor 1</span>
                  <input type="number" class='number' name="number" data-validate-minmax="2,3"
required='required'>
              </label>
              <div class='tooltip help'>
                <span>?</span>
                <div class='content'>
                  <b></b>
```

```
                    <p>Number must be between 2 and 3</p>
                </div>
            </div>
        </div>
        <div class="field">
            <label>
                <span>sensor 2</span>
                                        <input type="number" class='number' name="number"
data-validate-minmax="45,50" required='required'>
            </label>
            <div class='tooltip help'>
                <span>?</span>
                <div class='content'>
                    <b></b>
                    <p>Number must be between 45 and 50</p>
                </div>
            </div>
        </div>
        <div class="field">
            <label>
                <span>sensor 3</span>
                                        <input type="number" class='number' name="number"
data-validate-minmax="50,53" required='required'>
            </label>
```

```
                <div class='tooltip help'>
                  <span>?</span>
                  <div class='content'>
                    <b></b>
                    <p>Number must be between 50 and 53</p>
                  </div>
                </div>
              </div>
<fieldset>
              <p>
                PROJECT BY:
                K M SRIKRISHNA,
                R SANTHOSH,
                M KARTHIK SRINIVAS,
                V SRI HARI SUDARSHAN.
              </p>
              <input name="somethingHidden" required="required" type="text" style='display:none' />
            </fieldset>
            <button type='submit'>Submit</button>
            <button type='reset'>Reset</button>
          </form>
        </section>
      </div>
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.9.1/jquery.min.js"></script>
```

```
<script src="multifield.js"></script>

<script src="validator.js"></script>

<script>

   var validator = new FormValidator({"events" : ['blur', 'paste', 'change']}, document.forms[0]);



   // on form "submit" event

   document.forms[0].onsubmit = function(e){

      var submit = true;

            alert("Your sensor inputs are stored in Database successfully!");

   }


   // on form "reset" event

   document.forms[0].onreset = function(e){

         validator.reset();

   }


   // stuff related ONLY for this demo page:

   $('.toggleValidationTooltips').change(function(){

      validator.settings.alerts = !this.checked;


      if( this.checked )

         $('form .alert').remove();

   }).prop('checked',false);
```

```
  </script>
</body>
</html>




 regex : {

            numeric      : /^[0-9]+$/i,
            alphanumeric : /^[a-zA-Z0-9]+$/i,
            time         : /^([0-1][0-9]|2[0-3]):[0-5][0-9]$/,

            }
```

Output:

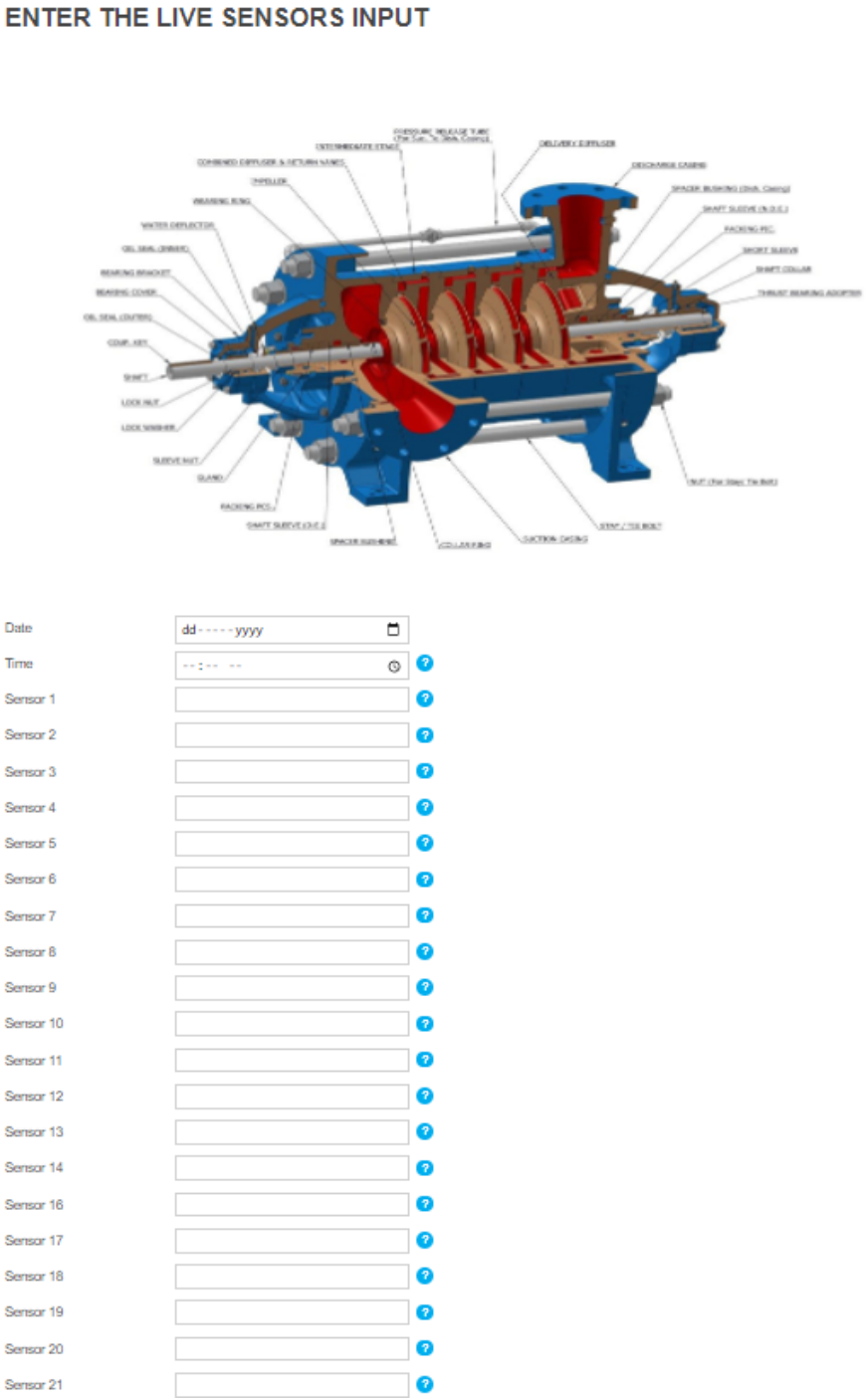


Fig.7.3 Code being executed and deployed using deployment server

7.4 ML Prediction:

```
"""

Created on Mon Feb 22 20:41:42 2021



@author: srihari

"""



import numpy as np

import matplotlib.pyplot as plt

import pandas as pd



ds=pd.read_csv('sensor.csv')

ds=ds.drop(['timestamp','Unnamed: 0','sensor_15'],axis=1)



ds.replace([np.inf, -np.inf], np.nan, inplace=True)

ds=ds.fillna(ds.mean())

ds.shape

x=ds.iloc[:,0:51].values



ds['machine_status'].unique()



#label_encoder

from sklearn import preprocessing
```

```
label_encoder = preprocessing.LabelEncoder()
ds['machine_status']= label_encoder.fit_transform(ds['machine_status'])
y=ds.iloc[:,-1].values


#splitting data
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=0)


ds.plot(subplots =True, sharex = True, figsize = (20,50))



#logistic regression
from sklearn.linear_model import LogisticRegression
c= LogisticRegression(random_state=0)
c.fit(xtrain,ytrain)


#prediction
ypred=c.predict(xtest)
```

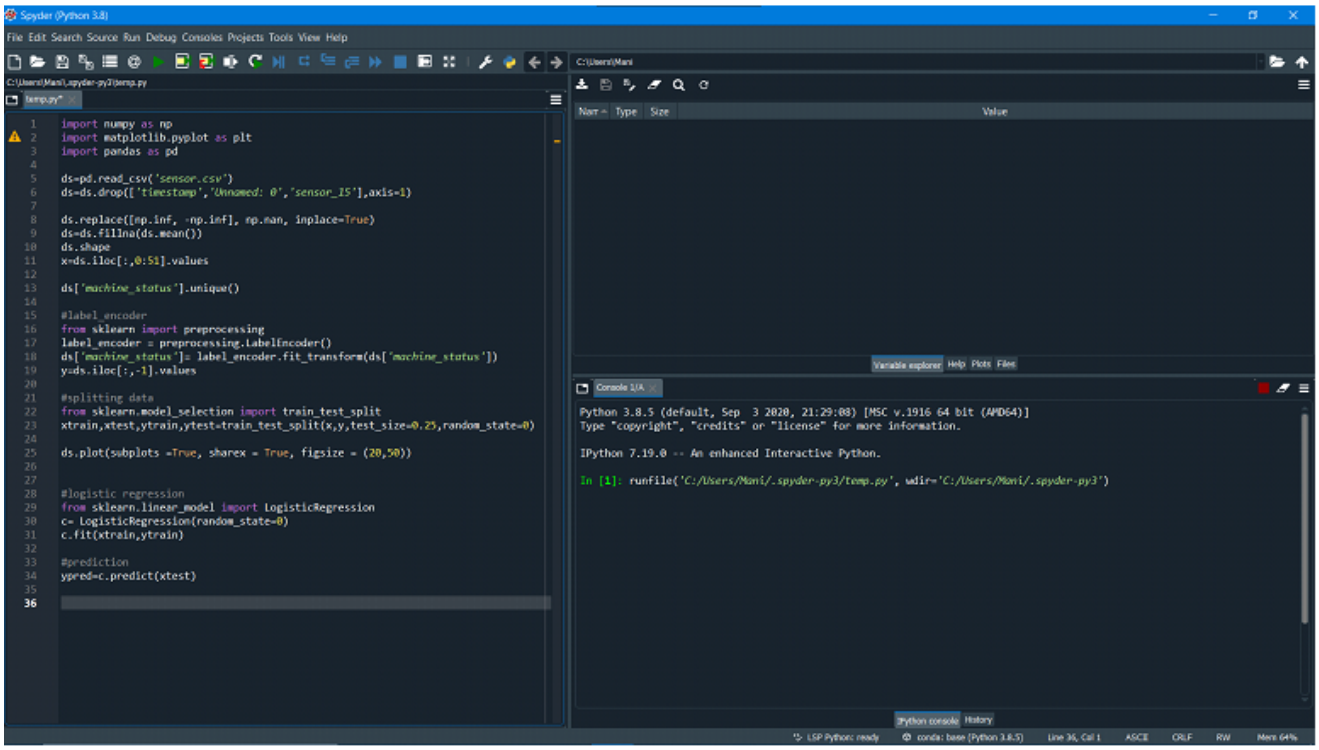


Fig.7.4 Code being executed on Spyder(Anaconda3)

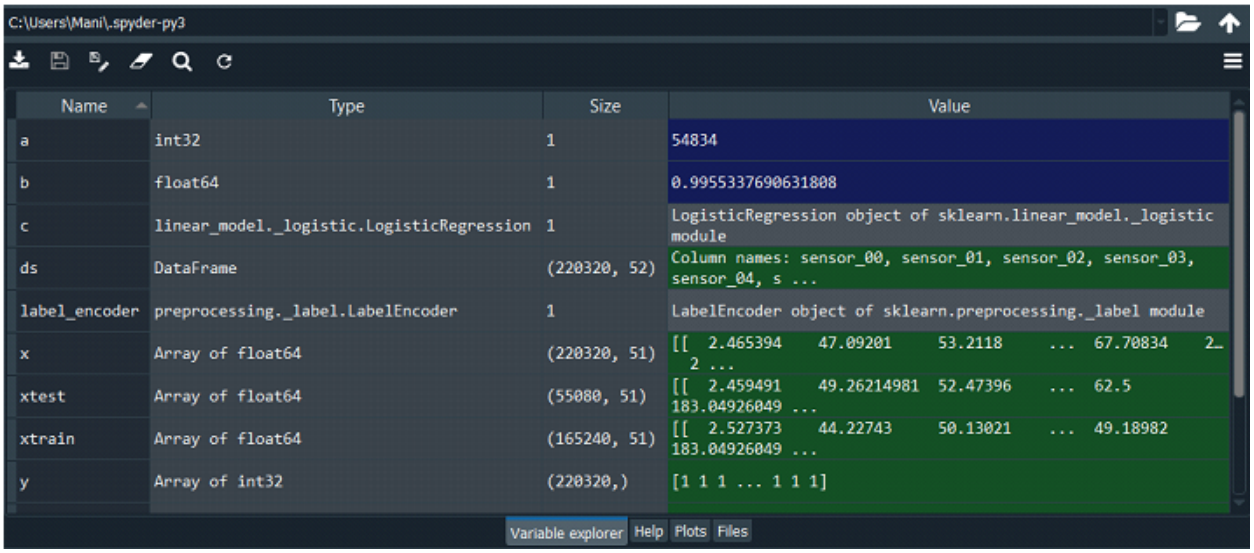
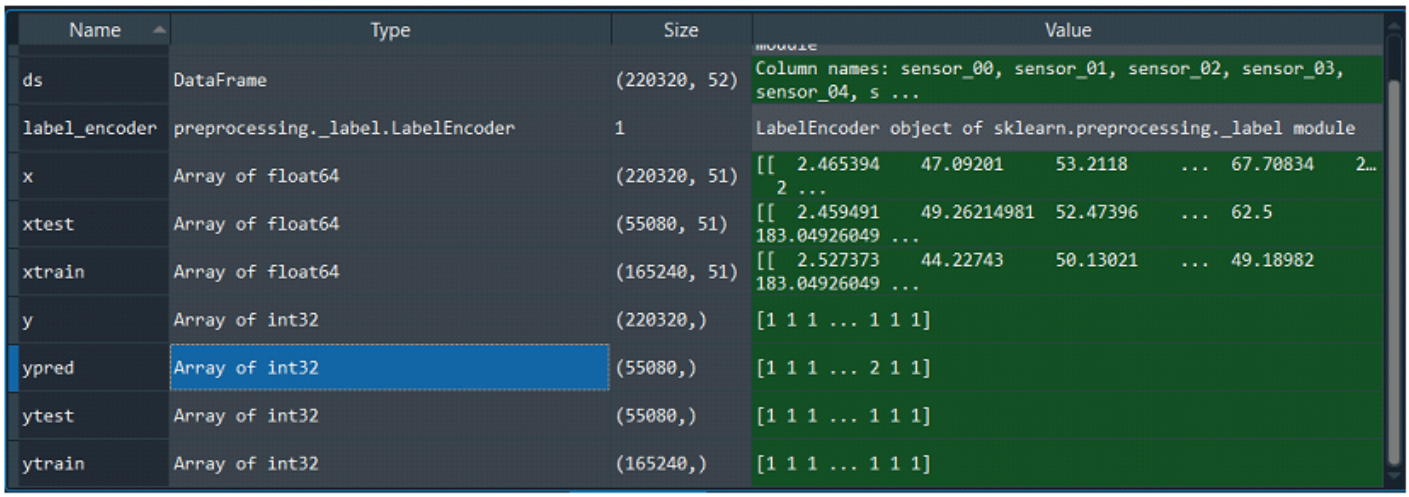


Fig.7.5 Output of the predicted values



Name	Type	Size	Value
ds	DataFrame	(220320, 52)	Column names: sensor_00, sensor_01, sensor_02, sensor_03, sensor_04, s ...
label_encoder	preprocessing._label.LabelEncoder	1	LabelEncoder object of sklearn.preprocessing._label module
x	Array of float64	(220320, 51)	[[2.465394 47.09201 53.2118 ... 67.70834 2… 2 ...
xtest	Array of float64	(55080, 51)	[[2.459491 49.26214981 52.47396 ... 62.5 183.04926049 ...
xtrain	Array of float64	(165240, 51)	[[2.527373 44.22743 50.13021 ... 49.18982 183.04926049 ...
y	Array of int32	(220320,)	[1 1 1 ... 1 1 1]
ypred	Array of int32	(55080,)	[1 1 1 ... 2 1 1]
ytest	Array of int32	(55080,)	[1 1 1 ... 1 1 1]
ytrain	Array of int32	(165240,)	[1 1 1 ... 1 1 1]

Fig.7.6.Output Screen

The outlined value**(ypred)** is the predicted value using Logistic Regression (A Machine Learning Algorithm)

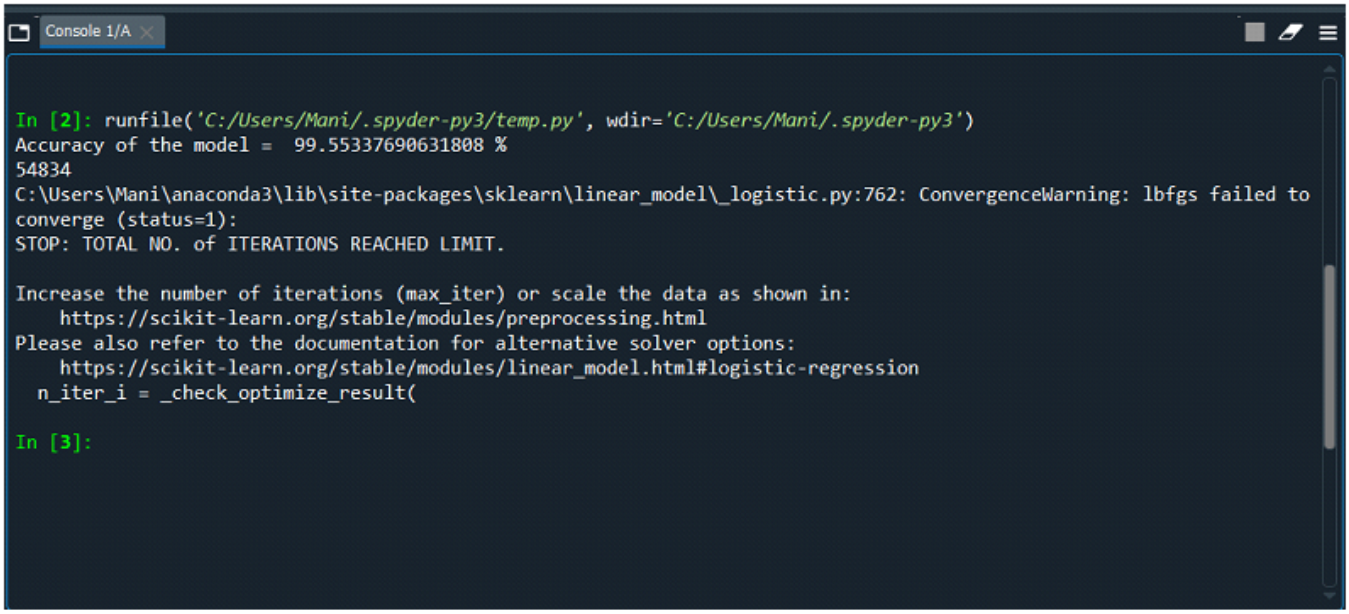


Fig.7.7. Output Screen

The above output shows the difference in the predicted and resultant values. There is an accuracy of **99.53379690631808%** which is the difference in value between **55080** and predicted **54834**.

7.5 Data Analysis:

Now We use Machine Learning to analyze the normal, broken and recovery state of the pump sensor. The graph shows the visual representation on a time scale about how long the pump sensor has been in normal state and has gone into broken state and finally revealing the time taken to jump to recovery.

```
fig, ax = plt.subplots(figsize=(18,5))

plt.xticks(rotation=90)

ax.plot(data.loc['2018-04-17 12:00:00':'2018-04-20 12:00:00', 'machine_status'],marker='o', linestyle='-')

plt.grid(True)

ax.set_ylabel('Reading Unit')

ax.set_title('Second Broken: machine_status Reading')

# Set x-axis major ticks to weekly interval, on Mondays
```

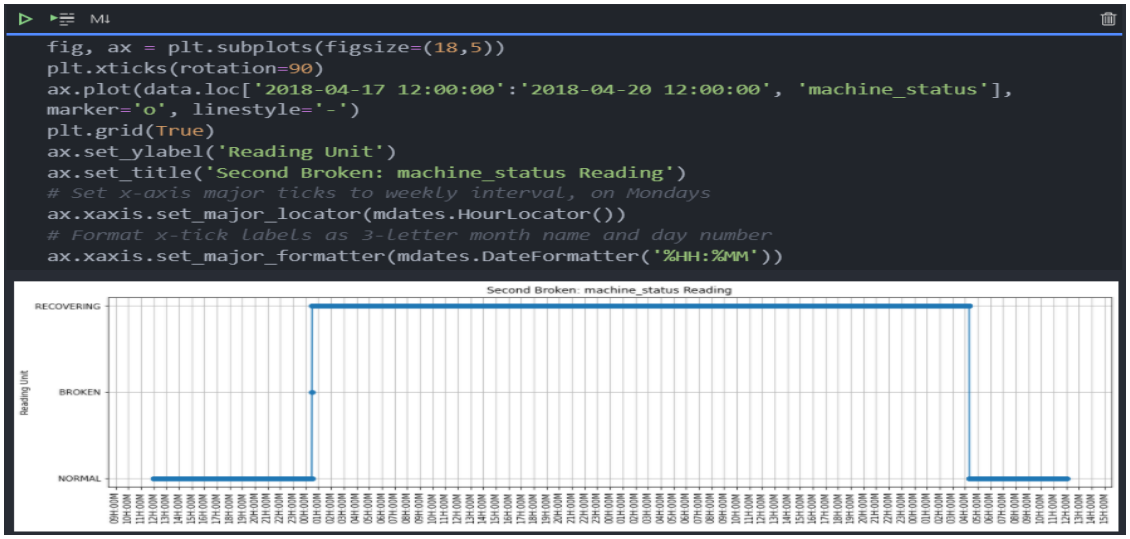


Fig.7.8. Analysis graph

```
ax.xaxis.set_major_locator(mdates.HourLocator())

# Format x-tick labels as 3-letter month name and day number

ax.xaxis.set_major_formatter(mdates.DateFormatter('%HH:%MM'))
```

```
fig, ax = plt.subplots(figsize=(18,5))

plt.xticks(rotation=90)

ax.plot(data.loc['2018-06-28 12:00:00':'2018-07-06 12:00:00', 'machine_status'],marker='o', linestyle='-')

plt.grid(True)

ax.set_ylabel('Reading Unit')

ax.set_title('Fifth Broken: machine_status Reading')

# Set x-axis major ticks to weekly interval, on Mondays

ax.xaxis.set_major_locator(mdates.DayLocator())

# Format x-tick labels as 3-letter month name and day number

ax.xaxis.set_major_formatter(mdates.DateFormatter('%d'))
```

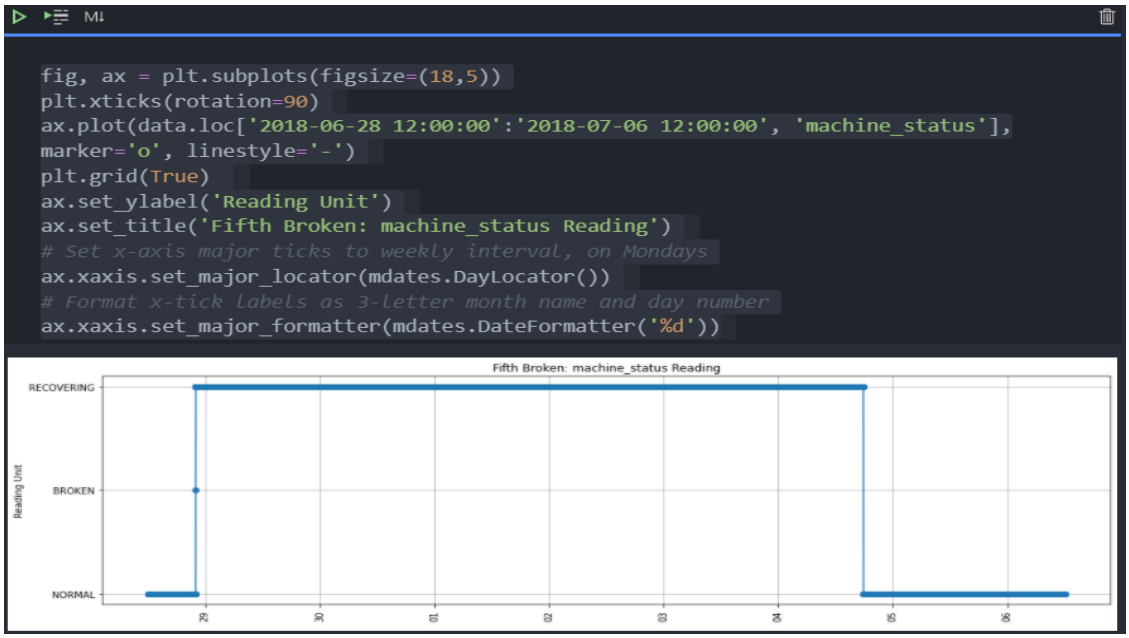


Fig.7.9. Analysis Graph 2

CHAPTER 8

Application Interface (Mobile App)

The below given is the written dart code on Visual Code Studio IDE using Dart

```
import 'dart:math' as math show pi;

import 'package:flutter/material.dart';

import 'package:collapsible_sidebar/collapsible_sidebar.dart';

import 'package:url_launcher/url_launcher.dart';

void main() => runApp(MyApp());

class MyApp extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      debugShowCheckedModeBanner: false,
      title: 'Digital Triplet',
      home: Scaffold(
        body: SidebarPage(),
      ),
    );
  }
}

class SidebarPage extends StatefulWidget {
  @override
  _SidebarPageState createState() => _SidebarPageState();
}

class _SidebarPageState extends State<SidebarPage> {
  List<CollapsibleItem> _items;
  String _headline;
  NetworkImage _avatarImg =
```

```
NetworkImage('https://www.clipartmax.com/png/middle/47-475704_waterpump-water-pump-clip-art.png');

@override
void initState() {
  super.initState();
  _items = _generateItems;
  _headline = _items.firstWhere((item) => item.isSelected).text;
}
//launchers
_about() async {
  const url = 'https://drive.google.com/file/d/1i3M1heXXzZOnMOdJHE2i-xrI3K9Z3Un7/view?usp=sharing';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
_data() async {
  const url =
'https://docs.google.com/spreadsheets/d/1H1VFxPh5ed4VexdxPgd9dYk4O1CZdcNvjGLqrjKX5tY/edit?usp=sharing';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
_status() async {
  const url = 'https://adoring-wing-dfce4b.netlify.app/';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
_ml() async {
  const url = 'https://digital-triplet-analysis.tiiny.site/';
  if (await canLaunch(url)) {
```

```
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
_analysis() async {
  const url = 'https://digital-triplet-analysis.tiiny.site/';
  if (await canLaunch(url)) {
    await launch(url);
  } else {
    throw 'Could not launch $url';
  }
}
List<CollapsibleItem> get _generateItems {
  return [
    CollapsibleItem(
      text: 'About',
      icon: Icons.import_contacts,
      onPressed: _about,
      isSelected: true,
    ),
    CollapsibleItem(
      text: 'Data Set',
      icon: Icons.snippet_folder,
      onPressed: _data,
    ),
    CollapsibleItem(
      text: 'Machine Status',
      icon: Icons.memory,
      onPressed: _status,
    ),
    CollapsibleItem(
      text: 'ML Prediction',
      icon: Icons.flaky,
      onPressed: _ml,
    ),
    CollapsibleItem(
      text: 'Data Analysis',
```

```
      icon: Icons.analytics,
      onPressed: _analysis,
    ),
  ];
}

@override
Widget build(BuildContext context) {
  var size = MediaQuery.of(context).size;
  return SafeArea(
    child: CollapsibleSidebar(
      items: _items,
      avatarImg: _avatarImg,
      title: 'DIGITAL TRIPLET',
      body: _body(size, context),
    ),
  );
}

Widget _body(Size size, BuildContext context) {
  return Container(
    height: double.infinity,
    width: double.infinity,
    color: Colors.blueGrey[50],
    child: Center(
      child: Transform.rotate(
        angle: math.pi / 2,
        child: Transform.translate(
          offset: Offset(-size.height * 0.3, -size.width * 0.23),
          child: Text(
            _headline,
            style: Theme.of(context).textTheme.headline1,
            overflow: TextOverflow.visible,
            softWrap: false,
```

8.1 Output of the Flutter Code (Application Interface)

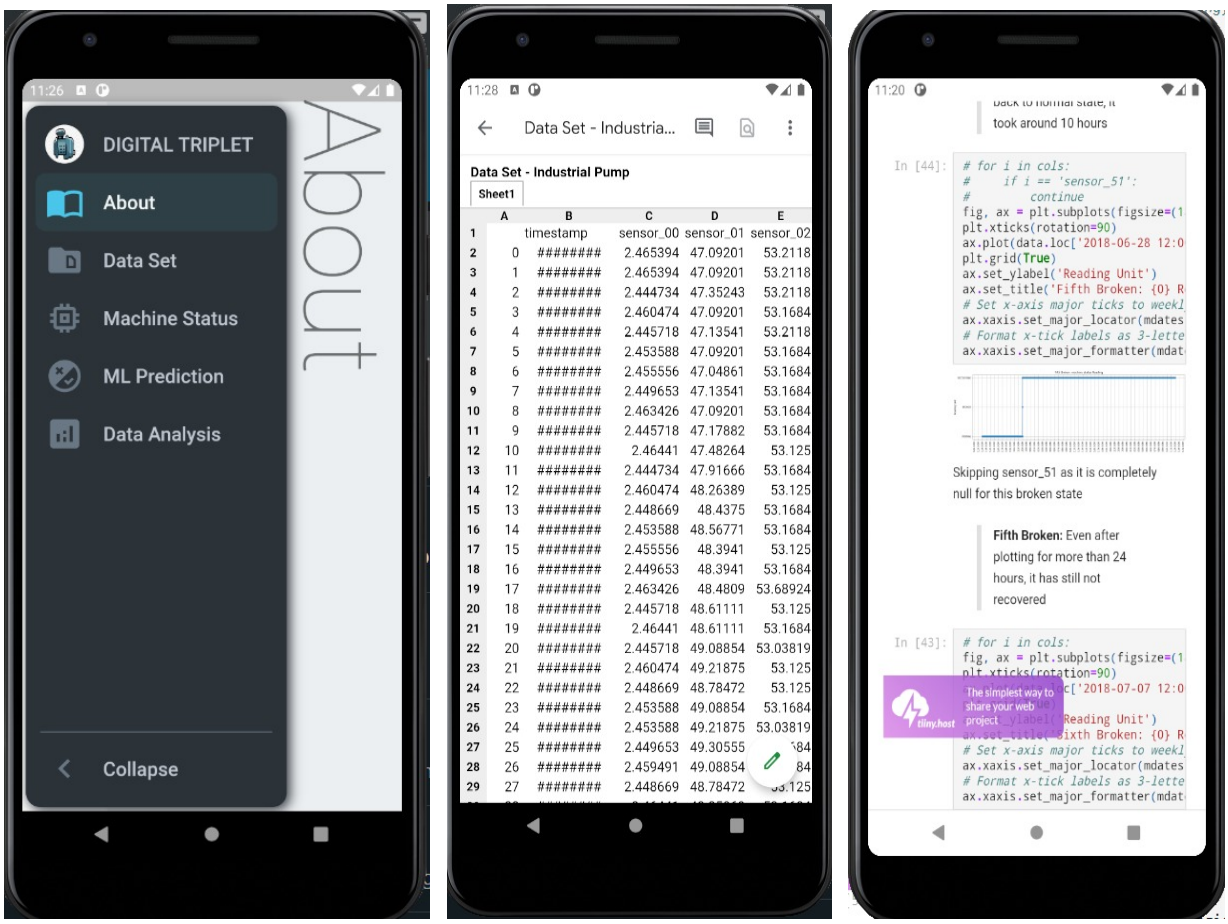


Fig.8.1. Output of Flutter Code

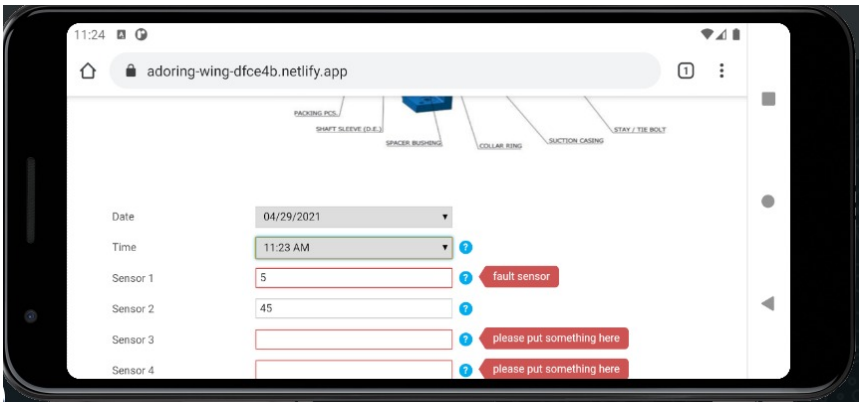


Fig.8.2. Output of Flutter Code

CHAPTER 9

RESULT

The interface we create for the application of the digital triplet will be user-friendly and easily accessible. The main aim we try to achieve is to find the right balance between machine and technology so that lots of time and energy will be saved in the course of accessing machines with their individual properties. Thus, one place to find all the answers about a machine is at our fingerprints.

CHAPTER 10

CONCLUSION

This paper described in conclusion aims to educate the ability of the manufacturing system engineers to operate, execute fault diagnosis and recovery, and design manufacturing systems[5], through exercising Digital Triplet of learning factories. First, this paper pointed out that Industry 4.0 takes rather a top-down approach and it seems to be difficult to deal with such continuous improvement where engineers construct various engineering cycles. For solving this problem, this paper proposed Digital Triplet, which aims to support engineers to solve problems and create various values throughout a product life cycle by integrating the physical world, the cyber world, and the intelligent activity world. Then, this paper introduced the education program, especially the exercise course using the digital Triplets of a learning factory. Future works includes:

- We will execute a trial of the exercise course. By analyzing the results[6] of the trial, we will improve the course as well as the whole curriculum.

- We recognize that one of the most important abilities of the manufacturing system engineers to educate is to construct the Digital Triplet of a manufacturing system. We will expand the program so as to deal with this issue.

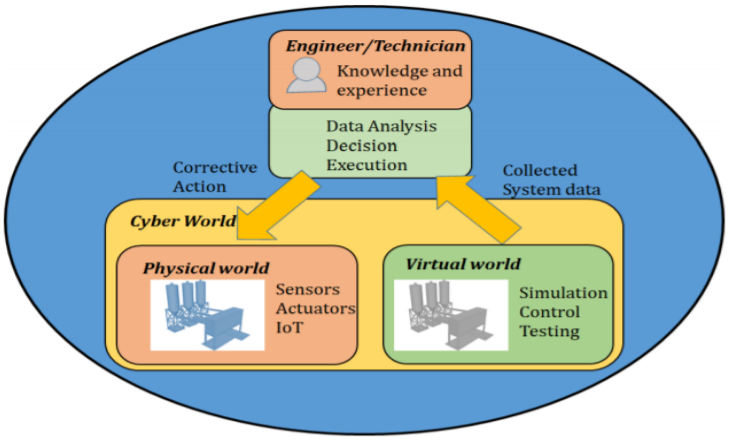


Fig.10.1.Conclusion Diagram

CHAPTER 11

REFERENCES

(1) Lee, J.; Lapira, E.; Bagheri, B.; Kao, H.-A. Recent advances and trends in predictive manufacturing systems in the big data environment. Manuf. Lett. 2013, 1, 38–41.

(2) SIEMENS Digitalization in Industry: Twins with Potential. Available online: https://new.siemens.com/global/en/company/stories/industry/the-digital-twin.html (accessed on 26 November 2019).

(3) Patriarca, R.; Bergström, J.; Di Gravio, G.; Costantino, F. Resilience engineering: Current status of the research and future challenges. Saf. Sci. 2018, 102, 79–100.

(4) Uhlemann, T.; Lehmann, C.; Steinhilper, R. The Digital Twin: Realizing the Cyber-Physical Production System for Industry 4.0. In Proceedings of the 24th CIRP Conference on Life Cycle Engineering, Kamakura, Japan, 8–10 March 2017; pp. 335–340.

(5) Qi, Q.; Tao, F. Digital Twin and Big Data Towards Smart Manufacturing and Industry 4.0. IEEE Access 2018, 6, 3585–3593.

(6) Um, J.; Weyer, S.; Quint, F. Plug-and-Simulate with Modular Assembly Line enabled by Digital Twins and the use of AutomationML. IFAC PapersOnLine 2017, 50, 15904–15909.