**MACHINE LEARNING**

(CEREALS DATA ANALYSIS)

*A summer internship report Submitted in partial fulfillment of the requirements for the award of degree of*

**BACHELOR OF TECHNOLOGY**

*in*

**ELECTRONICS AND COMMUNICATION ENGINEERING**

by

K.V.S.S SIDDHARTHA (2210416130)

*Under the esteemed guidance of*

SMART BRIDGE





**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**GITAM UNIVERSITY**

Rudraram Mandal, Sangareddy district, Patancheru,

Hyderabad, 502329

Telangana

2019-2020



**CERTIFICATE**

This is Certified that the summer internship entitled **“MACHINE LEARNING (Cereals Data Analysis )”** which is a practical &/ theoretical &/ hardware work carried out by K.V.S.S SIDDHARTHA(2210416130) in partial fulfillment for the award of the degree of **Bachelor of Technology** in Department **Electronics and Communication Engineering**, during the year **2019-2020**. The summer internship has been approved as it satisfies the academic requirements.

Dr. Manjunathachari K

Professor & Head, ECE Department

**ACKNOWLEDGEMENT**

I express my sincere thanks and gratitude to the Head of the Department, DR..Manjunathachari for his supervision and guidance for the internship.

.I also express my sincere thanks and gratitude to of smart bridge for sharing their knowledge with us and for their guidance during the internship.

I wish to express deep sense of gratitude to , Associate Dean-Practice School, , Practice School Manager, , Practice School Faculty, , Practice School Faculty and , Practice School Facultyfor their co-operation, encouragement and timely suggestions.

Project Associates

**K.V.S. SIDDHARTHA (2210416130)**

**INDEX**

**CONTENTS** Page No

**1.INTRODUCTION TO MACHINE LEARNING**

**1.1** **What is MACHINE LEARNING**  **5**

**1.2 HISTORY OF MACHINE LEARNINGG 6**

**1.3 10**

**2.RASPBERRY-PI**

**2.1 Raspberry-pi Introduction 11**

**2.2Essentialto work witRaspberrPi(forgeneraluse) 11**

**2.3 Pin Layout 13**

**2.4 Virtual Network Computing(VNC) 14**

**3.PYTHON**

**3.1 Introduction to Python 18**

**3.2 Features of Python 18**

**3.3 Python variable types 19**

**4.Project(Smart Living) 24**

**5.Conclusion 26**

**1.INTRODUCTION TO MACHINE LEARNING**

The term Machine Learning was coined by Arthur Samuel in 1959, an American pioneer in the field of computer gaming and artificial intelligence and stated that “it gives computers the ability to learn without being explicitly programmed”.  
And in 1997, Tom Mitchell gave a “well-posed” mathematical and relational definition that “A computer program is said to learn from experience E with respect to some task T and some performance measure P, if its performance on T, as measured by P, improves with experience E.

Machine Learning is a latest buzzword floating around. It deserves to, as it is one of the most interesting subfields of Computer Science.

## 1.1 MACHINE LEARNING:

Machine learning implementations are classified into three major categories, depending on the nature of the learning “signal” or “response” available to a learning system which are as follows: -

1. **Supervised learning:** When an algorithm learns from example data and associated target responses that can consist of numeric values or string labels, such as classes or tags, in order to later predict the correct response when posed with new examples comes under the category of Supervised learning.
2. **Unsupervised learning: Whereas** when an algorithm learns from plain examples without any associated response, leaving to the algorithm to determine the data patterns on its own. This type of algorithm tends to restructure the data into something else, such as new features that may represent a class or a new series of un-correlated values. They are quite useful in providing humans with insights into the meaning of data and new useful inputs to supervised machine learning algorithms.

**1.2 HISTORY OF MACHINE LEARNING:**

Machine Learning is a sub-set of artificial intelligence where computer algorithms are used to autonomously learn from data and information. In machine learning computers don’t have to be explicitly programmed but can change and improve their algorithms by themselves.

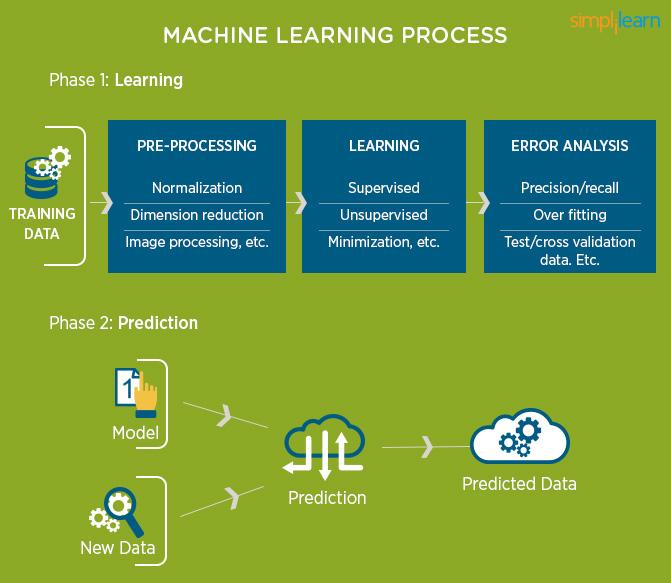
Today, machine learning algorithms enable computers to communicate with humans, autonomously drive cars, write and publish sport match reports, and find terrorist suspects. I firmly believe machine learning will severely impact most industries and the jobs within them, which is why every manager should have at least some grasp of what machine learning

# 1.2.1 Importance of Machine learning:

Consider some of the instances where machine learning is applied: the self-driving Google car, cyber fraud detection, online recommendation engines—like friend suggestions on Facebook, Netflix showcasing the movies and shows you might like, and “more items to consider” and “get yourself a little something” on Amazon—are all examples of applied machine learning.

All these examples echo the vital role machine learning has begun to take in today’s data-rich world. Machines can aid in filtering useful pieces of information that help in major advancements, and we are already seeing how this technology is being implemented in a wide variety of industries.

The process flow depicted here represents how machine learning works



With the constant evolution of the field, there has been a subsequent rise in the uses, demands, and importance of machine learning. Big data has become quite a buzzword in the last few years; that’s in part due to increased sophistication of machine learning, which helps analyze those big chunks of big data. Machine learning has also changed the way data extraction, and interpretation is done by

involving automatic sets of generic methods that have replaced traditional statistical techniques.

**Uses of Machine Learning**

Earlier in this article, we mentioned some applications of machine learning. To understand the concept of machine learning better, let’s consider some more examples: web search results, real-time ads on web pages and mobile devices, email spam filtering, network intrusion detection, and pattern and image recognition. All these are by-products of applying machine learning to analyze huge volumes of data.

Traditionally, data analysis was always being characterized by trial and error, an approach that becomes impossible when data sets are large and heterogeneous. Machine learning comes as the solution to all this chaos by proposing clever alternatives to analyzing huge volumes of data. By developing fast and efficient algorithms and data-driven models for real-time processing of data, machine learning is able to produce accurate results and analysis.

**Data Mining, Machine Learning, and Deep Learning**

Put simply, machine learning and data mining use the same algorithms and techniques as data mining, except the kinds of predictions vary. While data mining discovers previously unknown patterns and knowledge, machine learning reproduces known patterns and knowledge—and further automatically applies that information to data, decision-making, and actions.

Deep learning, on the other hand, uses advanced computing power and special types of neural networks and applies them to large amounts of data to learn, understand, and identify complicated patterns. Automatic language translation and medical diagnoses are examples of deep learning.

**Popular Machine Learning Methods**

How exactly do machines learn? Two popular methods of machine learning are supervised learning and unsupervised learning. It is estimated that about 70 percent of machine learning is supervised learning, while unsupervised learning ranges from 10 – 20 percent. Other methods that are less-often used are semi-supervised and reinforcement learning.

. **Supervised Learning**

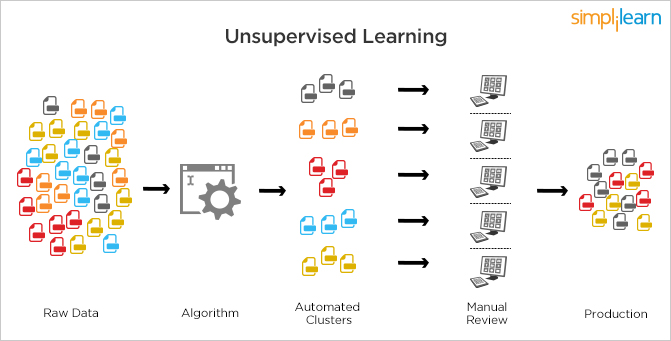
This kind of learning is possible when inputs and the outputs are clearly identified, and algorithms are trained using labeled examples. To understand this better, let’s consider the following example: an equipment could have data points labeled F (failed) or R (runs)

The learning algorithm using supervised learning would receive a set of inputs along with the corresponding correct output to find errors. Based on these inputs, it would further modify the model accordingly. This is a form of pattern recognition, as supervised learning happens through methods like classification, regression, prediction, and gradient boosting. Supervised learning uses patterns to predict the values of the label on additional unlabeled data.

Supervised learning is more commonly used in applications where historical data predict future events, such as fraudulent credit card transactions.

**Unsupervised Learning**

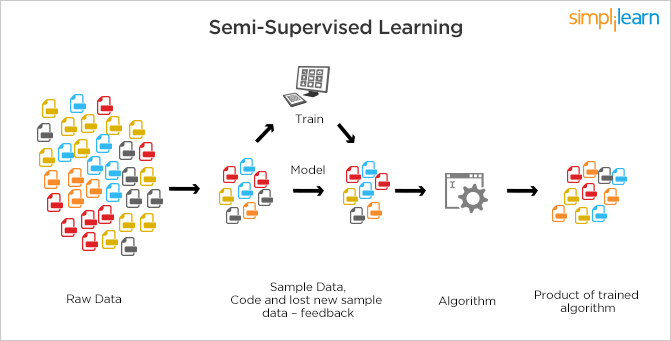
Unlike supervised learning, unsupervised learning is used with data sets without historical data. An unsupervised learning algorithm explores surpassed data to find the structure. This kind of learning works best for transactional data; for instance, it helps in identifying customer segments and clusters with certain attributes—this is often used in content personalization.

.

Popular techniques where unsupervised learning is used also include self-organizing maps, nearest neighbor mappig, singular value decomposition, and k-means clustering. Basically, online recommendations, identification of data outliers, and segment text topics are all examples of unsupervised learning.

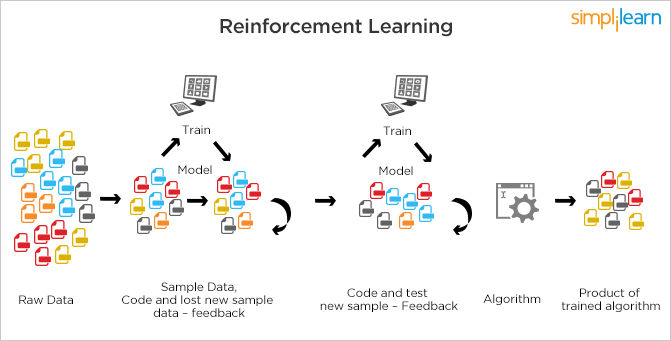
**Semi-supervised learning**

As the name suggests, semi-supervised learning is a bit of both supervised and unsupervised learning and uses both labeled and unlabeled data for training. In a typical scenario, the algorithm would use a small amount of labeled data with a large amount of unlabeled data.



**Reinforcement Learning**

This is a bit similar to the traditional type of data analysis; the algorithm discovers through trial and error and decides which action results in greater rewards. Three major components can be identified in reinforcement learning functionality: the agent, the environment, and the actions. The agent is the learner or decision-maker, the environment includes everything that the agent interacts with, and the actions are what the agent can do.



Reinforcement learning Reinforcement learning occurs when the agent chooses actions that maximize the expected reward over a given time. This is best achieved when the agent has a good policy to follow.

**2.Exploratory Data Analysis**

**2.1 Exploratory Data Analysis Introduction**

Exploratory Data Analysis (EDA) is the first step in our data analysis process. Here, we make a sense of the data you have and then figure out what questions we want to ask and how to frame them, as well as how best to manipulate our available data sources to get the answers you need.

We do this by taking a broad look at patterns, trends, outliers, unexpected results and so on in our existing data, using visual and quantitative methods to get a sense of the story this tells. We are looking for clues that suggest our logical next steps, questions or areas of research.

Developed by John Tukey in the 1970s, exploratory analysis is often described as a philosophy, and there are no hard-and-fast rules for how you approach it. That said, it also gave rise to a whole family of statistical-computing environments both used to help define, “What is EDA?” and to tackle specific tasks such as:

Spotting mistakes and missing data;

Mapping out the underlying structure of the data;

Identifying the most important variables;

Listing anomalies and outliers;

Testing a hypotheses / checking assumption related to a specific model;

Establishing a parsimonious model (one that can be used to explain the data with minimal predictor variables);

Estimating parameters and figuring out the associated confidence intervals or margins of error.

The EDA approach is precisely that--an approach--not a set of techniques, but an attitude/philosophy about how a data analysis should be carried out.

EDA is not identical to statistical graphics although the two terms are used almost interchangeably. Statistical graphics is a collection of techniques--all graphically based and all focusing on one data characterization aspect. EDA encompasses a larger venue;

EDA is an approach to data analysis that postpones the usual assumptions about what kind of model the data follow with the more direct approach of allowing the data itself to reveal its underlying structure and model. EDA is not a mere collection of techniques; EDA is a philosophy as to how we dissect a data set; what we look for; how we look; and how we interpret. It is true that EDA heavily uses the collection of techniques that we call "statistical graphics", but it is not identical to statistical graphics per se.

The seminal work in EDA is Exploratory Data Analysis, Tukey, (1977). Over the years it has benefitted from other noteworthy publications such as Data Analysis and Regression, Mosteller and Tukey (1977), Interactive Data Analysis, Hoaglin (1977), The ABC's of EDA, Velleman and Hoaglin (1981) and has gained a large following as "the" way to analyze a data set.

**Techniques**

Most EDA techniques are graphical in nature with a few quantitative techniques. The reason for the heavy reliance on graphics is that by its very nature the main role of EDA is to open-mindedly explore, and graphics gives the analysts unparalleled power to do so, enticing the data to reveal its structural secrets, and being always ready to gain some new, often unsuspected, insight into the data. In combination with the natural pattern-recognition capabilities that we all possess, graphics provides, of course, unparalleled power to carry this out.

The particular graphical techniques employed in EDA are often quite simple, consisting of various techniques of:Plotting the raw data (such as histograms, a box plot)Plotting simple statistics such as mean plots, standard deviation plots, box plots, and main effects plots of the raw data.

Positioning such plots so as to maximize our natural pattern-recognition abilities, such as using multiple plots per page.

evenly. I hope you can see that this is the same as “summing then dividing by n”.For any symmetrically shaped distribution (i.e., one with a symmetric histogram or pdf or pmf) the mean is the point around which the symmetry holds.

For non-symmetric distributions, the mean is the “balance point”: if the histogram

is cut out of some homogeneous stiff material such as cardboard, it will balance on

a fulcrum placed at the mean.For many descriptive quantities, there are both a sample and a population version. For a fixed finite population or for a theoretic infinite population described

by a pmf or pdf, there is a single population mean which is a fixed, often unknown,value called the mean parameter (see section 3.5). On the other hand, the “sample mean” will vary from sample to sample as different samples are taken, and so isa random variable. The probability distribution of the sample mean is referred toas its sampling distribution. This term expresses the idea that any experiment

could (at least theoretically, given enough resources) be repeated many times andvarious statistics such as the sample mean can be calculated each time. Oftenwe can use probability theory to work out the exact distribution of the samplestatistic, at least under certain assumptions.

The median is another measure of central tendency. The sample median is the middle value after all of the values are put in an ordered list. If there are an even number of values, take the average of the two middle values. (If there are ties at the middle, some special adjustments are made by the statistical software we will use. In unusual situations for discrete random variables, there may not be a unique median.) For symmetric distributions, the mean and the median coincide. For unimodal skewed (asymmetric) distributions, the mean is farther in the direction of the “pulled out tail” of the distribution than the median is. Therefore, for many cases of skewed distributions, the median is preferred as a measure of central tendency. For example, according to the US Census Bureau 2004 Economic Survey, the median income of US families, which represents the income above and below which half of families fall, was $43,318. This seems a better measure of central tendency than the mean of $60,828, which indicates how much each family would have if we all shared equally. And the difference between these two numbers is quite substantial. Nevertheless, both numbers are “correct”, as long as you understand their meanings. The median has a very special property called robustness.

A sample statistic is “robust” if moving some data tends not to change the value of the statistic. The median is highly robust, because you can move nearly all of the upper half and/or lower half of the data values any distance away from the median without changing the median. More practically, a few very high values or very low values usually have no effect on the median. A rarely used measure of central tendency is the mode, which is the most likely or frequently occurring value.

More commonly we simply use the term “mode” when describing whether a distribution has a single peak (unimodal) or two or more peaks (bimodal or multi-modal). In symmetric, unimodal distributions, the mode equals both the mean and the median. In unimodal, skewed distributions the mode is on the other side of the median from the mean. In multi-modal distributions there is either no unique highest mode, or the highest mode may well be unrepresentative of the central tendency. The most common measure of central tendency is the mean. For skewed distribution or when there is concern about outliers, the me

**3. PYTHON**

Basic program used in Machine learning is: **‘PYTHON’.**

**3.1 Introduction of PYTHON:**

* Python is a high-level, interpreted, interactive and object-oriented scripting language.
* Python is Interpreted: Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
* Python is Interactive: You can actually sit at a Python prompt and interact with the interpreter directly to write your programs.
* Python is Object-Oriented: Python supports Object Oriented style or technique of programming that encapsulates code within objects.

**History of PYTHON:**

* Python was developed by Guido van Rossum in the late eighties and early nineties at the National Research Institute for Mathematics and Computer Science in the Netherlands.
* Python is derived from many other languages, including ABC, Modula-3, C, C++, Algol-68, Small Talk, and Unix shell and other scripting languages.
* Python is copyrighted. Like Perl, Python source code is now available under the GNU General Public License (GPL).
* Python is now maintained by a core development team at the institute, although Guido van Rossum still holds a vital role in directing its progress.

**3.2 Features of PYTHON:**

* Easy-to-learn: Python has few keywords, simple structure, and a clearly defined syntax. This allows the student to pick up the language quickly.
* Easy-to-read: Python code is more clearly defined and visible to the eyes.
* Easy-to-maintain: Python's source code is fairly easy-to-maintaining.
* A broad standard library: Python's bulk of the library is very portable and cross-platform compatible on UNIX, Windows, and Macintosh.
* Portable: Python can run on a wide variety of hardware platforms and has the same interface on all platforms.
* Extendable: You can add low-level modules to the Python interpreter. These modules enable programmers to add to or customize their tools to be more efficient.
* Databases: Python provides interfaces to all major commercial databases.
* GUI Programming: Python supports GUI applications that can be created and ported to many system calls, libraries and windows systems, such as Windows MFC, Macintosh, and the X Window system of Unix.

**How to setup Python:**

* Python is available on a wide variety of platforms including Linux and Mac OS X. Let's understand how to set up our Python environment.
* The most up-to-date and current source code, binaries, documentation, news, etc., is available on the official website of Python.

<https://www.python.org/>

**3.3 Python Variable Types**:

* Variables are nothing but reserved memory locations to store values. This means that when you create a variable you reserve some space in memory.
* Python has various standard data types that are used to define the operations possible on them and the storage method for each of them.
* Python has five standard data types –
* Numbers
* String
* List
* Tuple
* Dictionary

**Python Numbers:**

* Number data types store numeric values. Number objects are created when you assign a value to them.
* Python supports four different numerical types − int (signed integers) long (long integers, they can also be represented in octal and hexadecimal) float (floating point real values) complex (complex numbers).

**Python Strings:**

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

**Python Lists:**

* Lists are the most versatile of Python's compound data types.
* A list contains items separated by commas and enclosed within square brackets ([]).
* To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.
* The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1.
* The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator.

**Python Tuples**:

* A tuple is another sequence data type that is similar to the list.
* A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.
* The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated.
* Tuples can be thought of as read-only lists.

For example − Tuples are fixed size in nature whereas lists are dynamic. In other words, a tuple is immutable whereas a list is mutable. You can't add elements to a tuple. Tuples have no append or extend method. You can't remove elements from a tuple. Tuples have no remove or pop method.

**Python Dictionary:**

* Python's dictionaries are kind of hash table type. They work like associative arrays or hashes found in Perl and consist of key-value pairs. A dictionary key can be almost any Python type, but are usually numbers or strings. Values, on the other hand, can be any arbitrary Python object.
* Dictionaries are enclosed by curly braces ({ }) and values can be assigned and accessed using square braces ([]).
* You can use numbers to "index" into a list, meaning you can use numbers to find out what's in lists. You should know this about lists by now, but make sure you understand that you can only use numbers to get items out of a list.
* What a dict does is let you use anything, not just numbers. Yes, a dict associates one thing to another, no matter what it is.

**Python Functions:**

Defining a Function –

You can define functions to provide the required functionality. Here are simple rules to define a function in Python.

Function blocks begin with the keyword def followed by the function name and parentheses ( ( ) ).

Any input parameters or arguments should be placed within these parentheses. You can also define parameters inside these parentheses.

The code block within every function starts with a colon (:) and is indented. The statement returns [expression] exits a function, optionally passing back an expression to the caller. A return statement with no arguments is the same as return None.

**Calling a Function**

* Defining a function only gives it a name, specifies the parameters that are to be included in the function and structures the blocks of code. Once the basic structure of a function is finalized, you can execute it by calling it from another function or directly from the Python prompt.

**Python Classes:**

* Python has been an object-oriented language since it existed. Because of this, creating and using classes and objects are downright easy.
* Class: A user-defined prototype for an object that defines a set of attributes that characterize any object of the class. The attributes are data members (class variables and instance variables) and methods, accessed via dot notation.
* Class variable: A variable that is shared by all instances of a class. Class variables are defined within a class but outside any of the class's methods. Class variables are not used as frequently as instance variables are.
* Data member: A class variable or instance variable that holds data associated with a class and its objects.
* Instance variable: A variable that is defined inside a method and belongs only to the current instance of a class.

**Project:Cereals data analysis**