A Report

On

**MACHINE LEARNING THROUGH PYTHON**

***Submitted in partial fulfilment of the requirement for the award of degree of***

**BACHELOR OF TECHNOLOGY**

In

**ELECTRONICS AND COMMUNICATION ENGINEERING**

***Submitted by***

KANDE VARSHA

Registered No: 20FE1A0476

BATCH:2020-24

***Under the supervision of***

**Dr. E Suneel**

Associate Professor



**DEPARTMENT OF ELECTRONICS AND COMMUNICATION ENGINEERING**

**VIGNAN’S LARA INSTITUTE OF TECHNOLOGY & SCIENCE**

**(An ISO 9001:2015 Certified, Approved by AICTE, Affiliated to JNTU, KAKINADA)**

**VADLAMUDI-522213, GUNTUR Dist., ANDHRAPRADESH**

**2020 - 2024**



**CERTIFICATE**

This is to certify that internship “**MACHINE LEARNING WITH PYTHON**” is a bona fide record of work done by **KANDE VARSHA** (20FE1A0476) as part of the curriculum in partial fulfilment of the requirement of the degree for Bachelor of Technology in Electronics and communication Engineering during the academic year (2022-2023)

coordinator Head of Department

DR. E Suneel DR. B Harish

DECLARATION

**KANDE VARSHA**, student of Bachelor of Engineering in Electronics & communication Engineering at Vignan’s Lara Institute of Technology and Science, Vadlamudi hereby declare that this internship work entitled “MACHINE LEARNING WITH PYTHON” is an original and bonafide work carried out by me at VERZEO in particular internship during the academic year 2022- 2023. I also declare that, to the best of my knowledge the work reported here is not from any other thesis or dissertation on the basis of which the degree or award was conferred on an earlier occasion by any student.

SIGNATURE OF CANDIDATE DATE: -

**ACKNOWLEDGEMENT**

I express my sincere gratitude to our respected internship guide **DR. E SUNEEL** for his valuable guidance and suggestions while carrying out the internship on **MACHINE LEARNING WITH PYTHON**. I extend my gratitude to engineering staff and management in smart knower for their support. I’m proud to be the part of VLITS, the institution which stood with us in the success completion of the internship. I take this opportunity to thank Department of Electronics and Communication Engineering, VLITS for supporting me I also thank my parents, group members, friends and almighty for constant support.

NAME OF THE CANDIDATE

KANDE VARSHA

(20FE1A0476)

**ABSTRACT**

Python is defined as a process of handling complexity by hiding unnecessary information from the user. This is one of the core concepts of object-oriented programming (OOP) languages.

In the statistical context, Machine Learning is defined as an application of artificial intelligence where available information is used through algorithms to process or assist the processing of statistical data. While Machine Learning involves concepts of automation, it requires human guidance.

In this report there are three chapters. Where the first chapter describes about the basics of the python and the second chapter describes about the machine learning and implementation of machine learning using python and at last the third chapter describes about the project of machine learning.

**CONTENTS**

**CHAPTER-1 PYTHON**

* 1. Introduction
  2. Data types
     1. Dictionaries
     2. Lists
     3. Sets
     4. Tuples
  3. Type Casting
  4. Python Input
  5. Python iterative execution
     1. While Statement
     2. For Statement
  6. Built in Functions
  7. Type conversion functions
  8. Calling Functions from Build in modules
     1. Random number functions
     2. Math Functions
  9. Function calls
  10. Arrays in python
  11. NumPy Arrays
  12. Data execution
  13. Data Transformation
  14. Catching Exceptions with try and except

**CHAPTER-2 MACHINE LEARNING**

2.1 Introduction

2.2 Supervised and Unsupervised learning

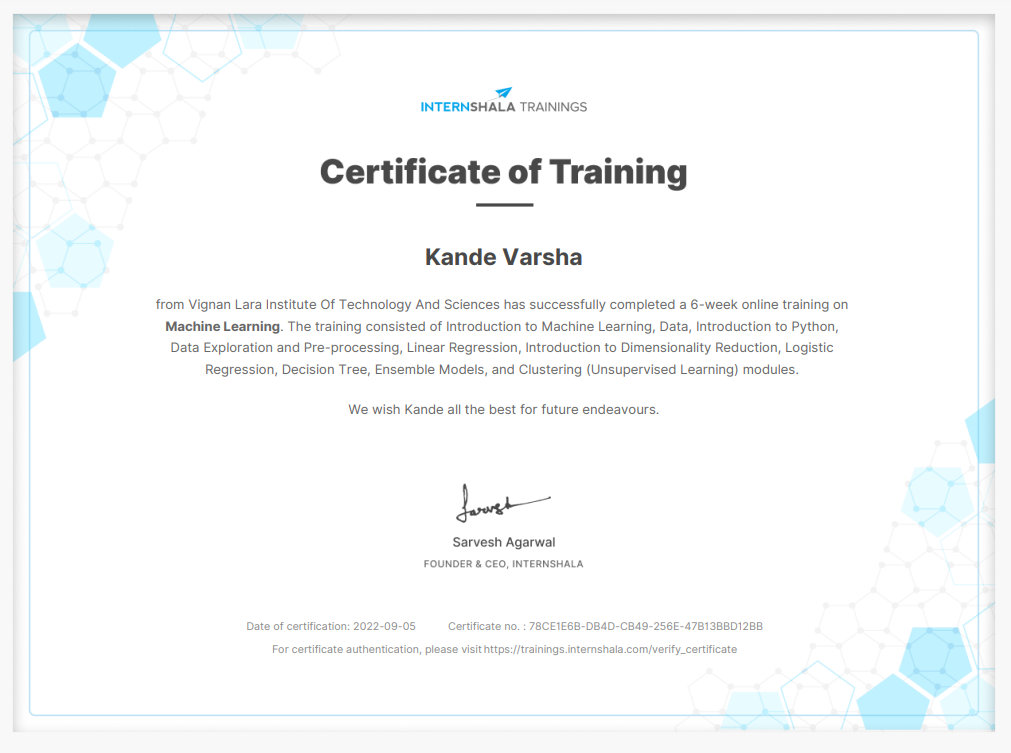
2.3 Machine Learning Model

2.4 SVN (Support vector machine)

2.4 Implementation of SVN in python

**CHAPTER-3 CONCLUSION**

**CERTIFICATE OF INTERNSHIP:**

****

**CHAPTER 1**

**PYTHON**

**1.1 Introduction**

Python is a popular programming language. It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* system scripting.

What can Python do?

* Python can be used on a server to create web applications.
* Python can be used alongside software to create workflows.
* Python can connect to database systems. It can also read and modify files.
* Python can be used to handle big data and perform complex mathematics.
* Python can be used for rapid prototyping, or for production-ready software development.

Python Syntax compared to other programming languages

* Python was designed for readability, and has some similarities to the English language with influence from mathematics.
* Python uses new lines to complete a command, as opposed to other programming languages which often use semicolons or parentheses.
* Python relies on indentation, using whitespace, to define scope; such as the scope of loops, functions and classes. Other programming languages often use curly-brackets for this purpose.

**print("Hello, World!")**

**1.2 Data types**

There are many different types of objects in Python. The most common object types: strings, integers and

floats. Anytime you are using words (text) in Python, you're using character strings (str for short). The most

common numbers, on the other hand, are integers (e.g. -2, 0, 99) and floats, which represent real numbers (e.g.

1.99, -23.340).

1 # integer

1.5 # float

'my string' #String

True #Bool

Integers

It can be positive or negative but without decimal values.

Floats

Floats are numbers with decimal points in it. The representation can vary from system to system.

Boolean

Boolean has only two possible values: True or False

**1.2.1 Dictionaries**

Dictionaries are the structures which holds the data in key - value pair format. For example, key can be student

id and value can be student object corresponding to that student id. It allows faster access to the object as

compared to the elements stored as part of list.

Dictionaries are more general form of the lists. They are more readable and one does not to keep the index in

mind while accessing it. For example, a list of months is available, in order to access the October month, one

need to keep its index in mind. Also, if days of those months also needs to be referred, then another list needs

to be maintained.

Dictionary:

my\_dict = {'Jan': 31, 'Feb':28, 'March': 31, 'Apr':30} # keys are strings

my\_dict

**1.2.2 Lists**

Lists are ordered collection of elements. The elements can be of any type like strings, numbers, Booleans, lists,

tuples etc. Each element of list has an address associated with it i.e. index. This index can be used to get

access to the list elements.

Creation

Empty list can be created with two ways. Usually [] brackets are used to define a list , whereas the list() function

is used for the creation of list from other collection.

my\_number\_list = [1, 2, 3, 4, 5, 6, 7]

my\_number\_list

month\_list = ['Jan', 'Feb', 'Mar', 'Apr', "May", 'Jun', 'Jul', "Aug", 'Sep', 'Oct', 'Nov',

month\_list[9] # accessing Oct

**1.2.3 Sets**

A set is a unique collection of objects in Python. It can be denoted with a curly bracket {}. Duplicate will be

removed.

Creation

Set is created with {} brackets.

my\_set = {'a' , 'b', 'c'}

my\_set

**1.2.4 Tuples**

Tuple is data structure that holds collection of different objects in it. For example, a tuple can hold numerical as

well as string values in it along with the other collections like list, tuples etc.

Creation

Empty tuple can be created with angular brackets.

My\_ tuple = ('a', 'b', 'c', 'd', 'e')

my\_tuple

**1.3 Type casting**

Converting from one object type to a different object type

int(1.5) #covert float to integer, loss of info

int('1') #convert string to integer

Converting integers to floats

type(2)

int

type(float(2))

float

Converting from strings to integers or floats

int('1') #Converting from string to int

1

type(int('1'))

int

Converting numbers to strings

str(1) #Converting string to int

‘1’

type(str(1))

str

Converting boolens to integers

int(True)

1

float(True)

1.0

Converting numbers to Booleans

bool(1)

True

bool(1.5)

True

bool(0)

False

**1.4 Python Inputs**

The input function is a simple way to get data from user. for example:

The string specified in the input function serves as message to the user about what needs to be keyed in. Any value entered by user is treated as string only. If number is entered by user, still it will be considered as string. For further processing it needs to be type casted to the appropriate data type.

input('What is your age?')

What is your age?28

Out:'28'

The user input can be saved as part of variable.

variable\_name = input('message to the user')

Convert type of input variable

age = input('What is your age?')

modified\_age = int(age)

print("You have entered age as ", modified\_age)

print("Type : " , type(modified\_age))

#What is your age?23

You have entered age as 23

Type: <class 'int'>

Using eval

The eval function converts the text entered by user into a number without explicitly typecasting.

my\_age = val (input ('What is your age? \n'))

print("You have entered age as ", my\_age)

print("Type : " , type(my\_age))

#What is your age?

12

You have entered age as 12

Type: <class 'int'>

**1.5 Python - Iterative Executions**

Sometimes its required to execute the same code block multiple times. For example, if list of student’s marks is given and we need to compute the total or average marks of that class. We need to iterate over all students marks one by one and need to use each of them for the operation to be performed i.e., summation or averaging.

In in each iteration, the operation is same but the data is changing. In such situations, loop statements helps us to execute same line of code multiple times in order to perform the same operation again and again.

**1.5.1 While Statement**

While statement helps to execute the code multiple times based on certain logical condition. Until the condition evaluates to true, the statements within the while block are executed. Once the condition evaluates to false, then control does not enter into the while block statements and continues the execution of the code that follows after the while block.

#Iterate over the first five numbers

my\_number\_list = range (5) #generate first five integers starting from zero

i = 0 # initialize the iteration variable

while i < len(my\_number\_list): #evaluate condition

print('(i) ', i, " (number) ", my\_number\_list[i]) # while block statement 1

i = i + 1 # while block statement 2

print ("Length of list ", len(my\_number\_list))

**Infinite Loops**

#Example of infinite loop

i = 0

while i < 5:

print (" i ", i)

print ("Out of while loop")

**Break Statement**

The break statement is use to break out of a loop before the loop is finished.

# Example

i = 0 #iteration variable initialized

while i < 5: # condition is evaluated

print(i) # block statements are executed until the condition is true i.e. 5 times

i = i + 1

**Continue Statement**

Sometimes we want to finish the execution of the current iteration and jump to the next iteration without

executing the further statements in the code block, then continue statement can be used. It helps to start a new

iteration.

stmt = ""

while stmt! = 'Quit':

stmt = input ("Enter the stmt (Use 'Quit' to stop)")

if stmt.startswith("#"):

continue

if len(stmt) > 0:

print(stmt)

**1.5.2 For Statement**

Computers are experts at doing things in repeated manner, especially without any errors. Python for loops in

one of the ways by which the code blocks can be executed in the repetitive manners, same as while loops.

With for loops, in advance you how many times the code execution will be repeated which is not the case with

while loops.

#Example: print first five integers along with value of i

for i in range (5):

print("(i) ", i, "(number)”, i+1)

**1.6 Built in Functions**

Built-in functions are developed by creators of Python and are we can use them without defining them. we can

just call them using the function name and with argument (if they accept any).

len() is built-in function that returns the number of elements present in data structure.

max() is built-in function that returns the maximum element from the given data structure

min() is built-in function that returns the minimum element from the given data structure

sum() is built-in function that returns the sum of elements in the given data structure

type() - returns the type of variable or object

sorted() arranges the list of elements in specified order.

reversed() returns the list of elements in reversed order

**1.7 Type conversion functions**

Python provides us built-in functions that helps to convert one variable to the other variable type.

For example,

int()

- converts float, Boolean or string into integer

float()

- converts int, Boolean or string into float

str()

- converts float, Boolean or integer into string

bool()

- converts float, string or integer into Boolean

**1.8 Calling functions from built-in modules**

**1.8.1 Random number functions**

Functions are part of modules as well. Random module is available as part of core Python language. One just need to import random module in order to use the functions defined in it. Once imported, the functions can be called in as usual i.e., with or without passing arguments to it.

import random

help(random)

print(random. random() )

**1.8.2 Math functions**

Math package is available by default. One can import it in the program and can call the functions defined in it.

import math

print(math.floor(1.4))

print(math.ceil(1.4))

print(math.isnan(2))

print(math.sqrt(5))

**1.9 Functions calls**

Functions are the collection of named statements. It can be considered as grouped collection of statements which can be reused at multiple places by just calling the function.

Function definition

The statements computing average can be enclosed into a block that is function. Function may or may not

accept input but will carry out the designated operations. In return, it may or may not return the outcome.

Let’s define a function for computing the average. It will accept list of numbers as input and will produce the

average as outcome.

def print\_welcome\_message():

print('\*' \* 12)

print("Welcome to the ABC software")

print()

print()

print("(c) All rights reserved.")

**1.10 Array in python**

**Array Indexing**

Similar to list indexing, arrays can be accessed using the indices.

In multi-dimensional arrays, comma separated type of indices can be used to access elements.

Values can be modified using indices,

**Array Slicing**

It can be used access the sub arrays i.e., array slicing is supported similar to list. To access a slice of array x,

use:

x [start: stop: step]

**Array copying**

Array slices are views, hence if they are modified then original array is changed.

In order to prevent this behaviour copy needs to be done.

**Reshaping of Arrays**

Reshape can be used to change the structure of array. Size of original and reshaped arrays should be same.

Reshaping can be used to convert an array into row matrix or column matrix.

**Array Concatenation**

Joining of two arrays is possible through np. concatenate, np.vstack and np.hstack.

**Array Splitting**

Splitting is done by split, vsplit and hsplit.

Fancy Indexing

Here arrays of indices is passed to access the elements.

NumPy (Numeric Python) provides an efficient way of storage and accessing the elements in numeric formats. They provide an easy to use interface by which the elements can be stored in a dense data structure and can be efficiently accessed whenever required.

Python Arrays

Python has arrays that stores the data in efficient manner which is having a fixed data type. The built-in array module helps in doing so.

import array

my\_array = array.array('i', [1, 2, 3, 4])

print("type : ", type(my\_array))

my\_array

The above code snippet creates a array holding all the elements which are of same type i.e. integer. 'i',

designates integer.

**1.11 NumPy Arrays**

But ndarray provides many operations on top of the arrays and hence needs to preferred over the built-in array

module.

NumPy array creation from List

Use np.array to create the ndarrays from Lists

import numpy as np

my\_list = [1, 2, 3, 4, 5]

my\_np\_array = np.array(my\_list)

print("type : ", type(my\_np\_array))

my\_np\_array

my\_float\_list = [1.1, 2.2, 3.3]

my\_float\_array = np.array(my\_float\_list)

my\_float\_array

All elements should be of same type. If types do not match, Numpy will upcast if possible.

**1.12 Data Exploration**

Lets use a real life example to explore the useful data analysis features of pandas. For this purpose, we are

going to use the Bike Sharing dataset that is made available on the UCI website. Using the various measures

those are captured as part of this dataset, lets explore how pandas powerful capabilities can be utilized for the

data wrangling and exploration.

The Basic Requirements

1.Reading Data From CSV

2.Formatting, cleaning and filtering Data Frames

3.Group-by and Concat / Merge

4.Writing Data to CSV

import pandas as pd

import numpy as np

Step1 - Reading Data From CSV

As the data is stored in csv format, use read\_csv function of pandas to read the data in pandas structure named

DataFrame.

bikes = pd. read\_csv("bike\_shairing\_hourly.csv")

Data Viewing

Let’s explore the first and last few rows of dataframe.

Step 2 - Formatting, cleaning and filtering Data

Frames

Let’s check how many data values are present in each column.

bikes.info ()

As all the features contains same number of data values, there are no missing values in dataset.

Properties of data

Lets check some properties of dataset :

bikes.columns

18 features are present in dataset, most of which looks integers in nature.

bikes.dtypes

dteday is datetime field so might need some transformation afterwards.

Unique feature values

Let’s explore the unique values present in each feature. These unique values can give us some hints while

doing grouping of the data.

bikes.dteday.unique()

bikes.season.unique()

Same thing can be done using [] operator

bikes["season"].unique() # four seasons

bikes.yr.unique() # two values - 0 for year 2011 and 1 for year 2012

bikes.mnth.unique() # 12 months data stored

temp, attempt, hum, windspeed are real numbers so will not interested in unique values for it.casual, registered and cnt are aggregated user counts for each day, so will not be interested in unique values for it.

Missing values identification

Missing values, null values, NaN needs to be identified.

bikes.isnull().sum()

bikes.isna().sum()

bikes.dteday.isnull().sum()

None of the feature contains missing values or NaN.

If null values are present in the dataframe, following code needs to be executed in order to get rid of them.

Duplicate records identification

Duplicate records needs to be identified and removed.

bikes.duplicated().sum()

There are no duplicate records in the data frame. Same duplicate check can be done on a column as well.

bikes.instant.duplicated().sum()

If duplicate records are present, then following code can be used to remove them.

**1.13 Data transformation**

The datetime column "dte" is captured as object but as it contains date time in it, a lot of interesting things can be derived from it and added to the existing data frame for further analysis.

**Feature Extraction**

Now extract year, month and day from "dte" series.

Same way month, day, weekday features can be extracted from it and added to the dataframe. But our data

frame already contains that information so we will not bother about it.

**Feature Reduction**

Some of the features like "instant" does not do any value addition for the analysis apart from identifying the

unique record. But same thing can be induced from the row label, hence the "instant" feature can be removed.

Similary, "cnt" attribute can be derived from "casual" and "registered" attributes. So "cnt" is redundant

information, hence can be removed.

Filtering

Lets divide the data frame into two different data frames each one containing different year.

Step3 - Group-by and Concate /Merge

Group by based on single feature

There are four seasons for which bike sharing data is collected. Let’s figure out how many users are using this

service as per different seasons.

Step 4 - Writing data frames to CSV file

Let’s utilise the pandas function to write back the data frame to CSV file.

Python - Conditional Executions

Frequently the programs need to follow a certain flow based on the fulfilment of certain condition. If

something is true then follow some path otherwise follow another path.

Simple Example of guessing a number

import random

#generate any random number between 1 to 10

number = random.randint(1,10)

guess = int(input("Enter an integer\n"))

if guess == number :

print("Your guess is correct!")

else:

print("Sorry, you failed to guess it!")The syntax of if statement generally has the following form :

if <condition> :

statements\_to\_be\_executed\_when\_condition\_is\_true

else:

statements\_to\_be\_executed\_when\_condition\_is\_false

Boolean Expression

The expression that returns true or false values as outcome are Boolean expressions. == or != or other logical

operators can be used withing Boolean expression to frame the desired condition.

**Conditional Execution**

Conditional statements give ability to change the flow of the program based on the conditions specified in the

statements.

The simple if statement can have only if block in it. That means, only if the condition is satisfied, then execute

the statements in the if block otherwise ignore those statements.

If block can consist of more than one statements. If the condition is satisfied all the statements in that block

are executed. The statements having same indentation constitutes the block.

**Alternative Execution**

But there will be situations where we have more than one branches of code that needs to be taken up based on

the evaluation of the condition. In that case, else statement and corresponding else block statements can be

added.

number = int(input("Enter the number\n"))

if number % 2 == 0 : # if entered number is even then only message is shown to the user

print(number ," is even number")

else :

print(number ," is odd number")

Chained Conditionals

mark = int(input("Enter the mark\n"))

if mark < 0 or mark > 100:

print("Invalid marks are entered!")

elif mark >= 0 and mark < 35 :

print("Sorry you failed!")

elif mark >= 35 and mark < 80 :

print("You passed in first class")

else :

print("You got distinction!")

Nested Conditionals

number1 = int(input("Enter first number \n"))

number2 = int(input("Enter second number \n"))

if number1 == number2:

print("Both numbers are same")

else:

if number1 > number2:

print("First number is bigger than second")

else:

print("First number is smaller than second")

**1.14 Catching Exceptions with try and except**

If while executing the code, error occurred then execution of code is stopped and error trace is shown to the

user. In order to handle that situation gracefully, the code that is expected to raise exception can be put inside a

try block and the error can be handled without halting the further execution of the program.

For example, you are asking user to enter a number and he has entered the string. When accession this string

value , error will be raised.

number = int(input("Enter a number\n")) # if number entered in string then it will throw er

if number % 2 == 0 :

print("Number is even")

else:

print("Number is odd")

This can be handled with try block and user can be made aware about the actual thing that has happened.

CHAPTER-2

MACHINE LEARNINIG

**2.1 Introduction**

What is Machine Learning?

Q - Your notion of Machine Learning

Machine Learning is the field of study that gives computers the capability to learn without being

explicitly programmed

● Learn what?

#1

In basic words, it is an attempt at enabling machines to think and make decisions like human

beings.

● Examples and Use cases of Machine Learning:

#2

Introduction

What is a Model?

● Basic motive of ML Algorithms

#1

● Concept of Algorithm {y=f(x)}

● Every real life phenomenon is governed by a mathematical formula. We are just trying to estimate

that formula using algos so that we can predict future. Now, obviously, we can not accurately

estimate the formula for every phenomenon in life due to lack of relevant data.

Data Matrix Notation

#2

● What are Predictors/Features

● What is Target/Response

● Concept of Design Matrix

**2.2 Supervised vs Unsupervised Learning**

The main difference between supervised vs unsupervised learning is the need for labelled training data.

Supervised machine learning relies on labelled input (features) and output (response) training data,

whereas unsupervised learning processes unlabelled or raw data. In supervised machine learning the

model learns the relationship between the labelled input and output data. Models are fine tuned until

they can accurately predict the outcomes of unseen data. However, labelled training data will often be

resource intensive to create. Unsupervised machine learning on the other hand learns from unlabelled

raw training data. An unsupervised model will learn relationships and patterns within this unlabelled

dataset, so is often used to discover inherent trends in a given dataset.

● Supervised - when we have both Features and Target available in the training data and we wish to

learn the relationship between these to make predictions in future

● Unsupervised - when we don’t have target labels in the training data and we are trying to find

patterns in the data itself.

Examples and use cases: Clustering vs Predicting weights

#1

Classification vs Regression

Both of these falls under the domain of Supervised Machine Learning

● Regression - We try to Predict a Quantitative variable

● Meaning of Quantitative - Continuous/Real value

● Classification - We try to predict a Qualitative variable

● Meaning of Qualitative - Categorical

● Binary Classification vs Multi-class Classification vs Multi-label Classification (one data point may

belong to more than one output)

Examples: Predicting weights vs Predicting Cancer or Not vs Predicting Polarity of a Review on Amazon

vs Predicting Movie Genre given the description

**2.3 Machine Learning Model**

Before discussing the machine learning model, we must need to understand the following formal definition of ML given by professor Mitchell:

***“***A computer program is said to learn from experience E with respect to some class of tasks T and performance measure P, if its performance at tasks in T, as measured by P, improves with experience E.”

The above definition is basically focusing on three parameters, also the main components of any learning algorithm, namely Task(T), Performance(P) and experience (E). In this context, we can simplify this definition as:

ML is a field of AI consisting of learning algorithms that:

 Improve their performance (P)

 At executing some task (T)

 Over time with experience (E)

**Task(T)**

From the perspective of problem, we may define the task T as the real-world problem to be solved. The problem can be anything like finding best house price in a specific location or to find best marketing strategy etc. On the other hand, if we talk about machine learning, the definition of task is different because it is difficult to solve ML based tasks by conventional programming approach.

A task T is said to be a ML based task when it is based on the process and the system must follow for operating on data points. The examples of ML based tasks are Classification, Regression, Structured annotation, Clustering, Transcription etc.

**Experience (E)**

As name suggests, it is the knowledge gained from data points provided to the algorithm or model. Once provided with the dataset, the model will run iteratively and will learn some inherent pattern. The learning thus acquired is called experience(E). Making an analogy with human learning, we can think of this situation as in which a human being is learning or gaining some experience from various attributes like situation, relationships etc. Supervised, unsupervised and reinforcement learning are some ways to learn or gain experience. The experience gained by out ML model or algorithm will be used to solve the task T.

**Performance (P)**

An ML algorithm is supposed to perform task and gain experience with the passage of time. The measure which tells whether ML algorithm is performing as per expectation or not is its performance (P). P is basically a quantitative metric that tells how a model is performing the task, T, using its experience, E. There are many metrics that help to understand the ML performance, such as accuracy score, F1 score, confusion matrix, precision, recall, sensitivity etc.

**2.4 SVM (Support vector machine)**

Support vector machines (SVMs) are powerful yet flexible supervised machine learning algorithms which are used both for classification and regression. But generally, they are used in classification problems. In 1960s, SVMs were first introduced but later they got refined in 1990. SVMs have their unique way of implementation as compared to other machine learning algorithms. Lately, they are extremely popular because of their ability to handle multiple continuous and categorical variables.

**Working of SVM**

An SVM model is basically a representation of different classes in a hyperplane in multidimensional space. The hyperplane will be generated in an iterative manner by SVM so that the error can be minimized. The goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH).

The followings are important concepts in SVM:

 **Support Vectors:** Datapoints that are closest to the hyperplane is called support vectors. Separating line will be defined with the help of these data points.

 **Hyperplane:** As we can see in the above diagram, it is a decision plane or space which is divided between a set of objects having different classes.

 **Margin:** It may be defined as the gap between two lines on the closet data points of different classes. It can be calculated as the perpendicular distance from the line to the support vectors. Large margin is considered as a good margin and small margin is considered as a bad margin.

The main goal of SVM is to divide the datasets into classes to find a maximum marginal hyperplane (MMH) and it can be done in the following two steps:

 First, SVM will generate hyperplanes iteratively that segregates the classes in best way.

 Then, it will choose the hyperplane that separates the classes correctly.

**2.5 Implementing SVM in Python**

For implementing SVM in Python we will start with the standard libraries import as follows:

import numpy as np

import matplotlib. pyplot as plt

from scipy import stats

import seaborn as sns; sns.set()

Next, we are creating a sample dataset, having linearly separable data, from sklearn.dataset.sample\_generator for classification using SVM:

from sklearn.datasets.samples\_generator import make\_blobs

X, y = make\_blobs(n\_samples=100, centers=2,

random\_state=0, cluster\_std=0.50)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer');

The following would be the output after generating sample dataset having 100 samples and 2 clusters:

We know that SVM supports discriminative classification. it divides the classes from each other by simply finding a line in case of two dimensions or manifold in case of multiple dimensions. It is implemented on the above dataset as follows:

xfit = np.linspace(-1, 3.5)

plt.scatter(X[:, 0], X[:, 1], c=y, s=50, cmap='summer')

plt.plot([0.6], [2.1], 'x', color='black', markeredgewidth=4, markersize=12)

for m, b in [(1, 0.65), (0.5, 1.6), (-0.2, 2.9)]:

plt.plot(xfit, m \* xfit + b, '-k')

plt.xlim(-1, 3.5);

**SVM Kernels**

In practice, SVM algorithm is implemented with kernel that transforms an input data space into the required form. SVM uses a technique called the kernel trick in which kernel takes a low dimensional input space and transforms it into a higher dimensional space. In simple words, kernel converts non-separable problems into separable problems by adding more dimensions to it. It makes SVM more powerful, flexible and accurate. The following are some of the types of kernels used by SVM:

**Linear Kernel**

It can be used as a dot product between any two observations. The formula of linear kernel is as below: 𝐾(𝑥,𝑥𝑖)=𝑠𝑢𝑚(𝑥∗𝑥𝑖)

From the above formula, we can see that the product between two vectors say 𝑥 & 𝑥𝑖 is the sum of the multiplication of each pair of input values.

**Polynomial Kernel**

It is more generalized form of linear kernel and distinguish curved or nonlinear input space. Following is the formula for polynomial kernel:

*K*(x, xi) = 1 + sum(x \* xi)^d

Here d is the degree of polynomial, which we need to specify manually in the learning algorithm.

**Radial Basis Function (RBF) Kernel**

RBF kernel, mostly used in SVM classification, maps input space in indefinite dimensional space. Following formula explains it mathematically:

*K(x,xi) = exp(-gamma \* sum((x – xi^2))*

Here, *gamma* ranges from 0 to 1. We need to manually specify it in the learning algorithm. A good default value of *gamma* is 0.1.

As we implemented SVM for linearly separable data, we can implement it in Python for the data

that is not linearly separable. It can be done by using kernels.

**Pros and Cons of SVM Classifiers**

**Pros of SVM classifiers**

SVM classifiers offers great accuracy and work well with high dimensional space. SVM classifiers basically use a subset of training points hence in result uses very less memory.

**Cons of SVM classifiers**

They have high training time hence in practice not suitable for large datasets. Another disadvantage

is that SVM classifiers do not work well with overlapping classes.

**Decision Tree**

In general, Decision tree analysis is a predictive modelling tool that can be applied across many areas. Decision trees can be constructed by an algorithmic approach that can split the dataset in different ways based on different conditions. Decisions tress are the most powerful algorithms that falls under the category of supervised algorithms.

They can be used for both classification and regression tasks. The two main entities of a tree are decision nodes, where the data is split and leaves, where we got outcome. The example of a binary tree for predicting whether a person is fit or unfit providing various information like age, eating habits and exercise habits.

* Classification decision trees: In this kind of decision trees, the decision variable is categorical. The above decision tree is an example of classification decision tree.
* Regression decision trees: In this kind of decision trees, the decision variable is continuous.

Split Creation A split is basically including an attribute in the dataset and a value. We can create a split in dataset with the help of following three parts:

• Part1: Calculating Gini Score: We have just discussed this part in the previous section.

• Part2: Splitting a dataset: It may be defined as separating a dataset into two lists of rows having index of an attribute and a split value of that attribute. After getting the two groups - right and left, from the dataset, we can calculate the value of split by using Gini score calculated in first part. Split value will decide in which group the attribute will reside.

• Part3: Evaluating all splits: Next part after finding Gini score and splitting dataset is the evaluation of all splits. For this purpose, first, we must check every value associated with each attribute as a candidate split. Then we need to find the best possible split by evaluating the cost of the split. The best split will be used as a node in the decision tree

CONCLUSION:

**Machine learning is a powerful tool for making predictions from data**. However, it is important to remember that machine learning is only as good as the data that is used to train the algorithms.