**CHAPTER-1**

**INTRODUCTION**

**1.1 INTRODUCTION**

Distributed machine learning (DML) has been widely used in distributed systems [1], [2], where no single node can get the intelligent decision from a massive dataset within an acceptable time [3]–[6]. In a typical DML system [7], a central server has a tremendous amount of data at its disposal. It divides the dataset into different parts and disseminates them to distributed workers who perform the training tasks and return their results to the center [8]–[10]. Finally, the center integrates these results and outputs the eventual model

Unfortunately, with the number of distributed workers increasing, it is hard to guarantee the security of each worker. This lack of security will increase the danger that attackers poison the dataset and manipulate the training result. Poisoning attack is a typical way to tamper the training data in machine learning. Especially in scenarios that newly generated datasets should be periodically sent to the distributed workers for updating the decision model, the attacker will have more chances to poison the datasets, leading to a more severe threat in DML.

Such vulnerability in machine learning has attracted much attention from researchers. Dalvi et al. [14] initially demonstrated that attackers could manipulate the data to defeat the data miner if they have complete information. Then Lowd et al. [15] claimed that the perfect information assumption is unrealistic, and proved the attackers can construct attacks with part of the information. Afterwards, a series of works were conducted [16]–[23], focusing on non-distributed machine learning context. Recently, there are a couple of efforts devoted in preventing data from being manipulated in DML. For example, Zhang et al. [24] and Esposito et al. [25] used game theory to design a secure algorithm for distributed support vector machine (DSVM) and collaborative deep learning, respectively. However, these schemes are designed for specific DML algorithm and cannot be used in general DML situations. Since the adversarial attack can mislead various machine learning algorithms, a widely applicable DML protection mechanism is urgent to be studied.

In this paper, we classify DML into basic distributed machine learning (basic-DML) and semi distributed machine learning (semi-DML), depending on whether the center shares resources in the dataset training tasks. Then, we present data poison detection schemes for basic-DML and semi-DML respectively. The experimental results validate the effect of our proposed schemes. We summary the main contributions of this paper as follows.

We put forward a data poison detection scheme for basic-DML, based on a so-called cross-learning data assignment mechanism. We prove that the cross-learning mechanism would consequently generate training loops, and provide a mathematical model to find the optimal number of training loops which has the highest security.

• We present a practical method to identify abnormal training results, which can be used to find out the poisoned datasets at a reasonable cost.

• For semi-DML, we propose an improved data poison detection scheme, which can provide better learning protection. To efficiently utilize the system resources, an optimal resource allocation scheme is developed.

**CHAPTER-2**

**LITERATURE SURVEY**

**1. “Collaborative task offloading in vehicular edge multi-access networks:**

Mobile edge computing (MEC) has emerged as a promising paradigm to realize user requirements with low-latency applications. The deep integration of multi-access technologies and MEC can significantly enhance the access capacity between heterogeneous devices and MEC platforms. However, the traditional MEC network architecture cannot be directly applied to the Internet of Vehicles (IoV) due to high speed mobility and inherent characteristics. Furthermore, given a large number of resource-rich vehicles on the road, it is a new opportunity to execute task offloading and data processing onto smart vehicles. To facilitate good merging of the MEC technology in IoV, this article first introduces a vehicular edge multi-access network that treats vehicles as edge computation resources to construct the cooperative and distributed computing architecture. For immersive applications, co-located vehicles have the inherent properties of collecting considerable identical and similar computation tasks. We propose a collaborative task offloading and output transmission mechanism to guarantee low latency as well as the application- level performance. Finally, we take 3D reconstruction as an exemplary scenario to provide insights on the design of the network framework. Numerical results demonstrate that the proposed scheme is able to reduce the perception reaction time while ensuring the application-level driving experiences.

**2. Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks:**

The Internet of Things (IoT) platform has played a significant role in improving road transport safety and efficiency by ubiquitously connecting intelligent vehicles through wireless communications. Such an IoT paradigm however, brings in considerable strain on limited spectrum resources due to the need of continuous communication and monitoring. Cognitive radio (CR) is a potential approach to alleviate the spectrum scarcity problem through opportunistic exploitation of the underutilized spectrum. However, highly dynamic topology and time-varying spectrum states in CR-based vehicular networks introduce quite a few challenges to be addressed. Moreover, a variety of vehicular communication modes, such as vehicle-to-infrastructure and vehicle-to-vehicle, as well as data QoS requirements pose critical issues on efficient transmission scheduling. Based on this motivation, in this paper, we adopt a deep Q -learning approach for designing an optimal data transmission scheduling scheme in cognitive vehicular networks to minimize transmission costs while also fully utilizing various communication modes and resources. Furthermore, we investigate the characteristics of communication modes and spectrum resources chosen by vehicles in different network states, and propose an efficient learning algorithm for obtaining the optimal scheduling strategies. Numerical results are presented to illustrate the performance of the proposed scheduling schemes.

**3. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems:**

MXNet is a multi-language machine learning (ML) library to ease the development of ML algorithms, especially for deep neural networks. Embedded in the host language, it blends declarative symbolic expression with imperative tensor computation. It offers auto differentiation to derive gradients. MXNet is computation and memory efficient and runs on various heterogeneous systems, ranging from mobile devices to distributed GPU clusters.  
This paper describes both the API design and the system implementation of MXNet, and explains how embedding of both symbolic expression and tensor operation is handled in a unified fashion. Our preliminary experiments reveal promising results on large scale deep neural network applications using multiple GPU machines.

**4. Machine learning on big data: Opportunities and challenges:**

Machine learning (ML) is continuously unleashing its power in a wide range of applications. It has been pushed to the forefront in recent years partly owing to the advent of big data. ML algorithms have never been better promised while challenged by big data. Big data enables ML algorithms to uncover more fine-grained patterns and make more timely and accurate predictions than ever before; on the other hand, it presents major challenges to ML such as model scalability and distributed computing. In this paper, we introduce a framework of ML on big data (MLBiD) to guide the discussion of its opportunities and challenges. The framework is centered on ML which follows the phases of preprocessing, learning, and evaluation. In addition, the framework is also comprised of four other components, namely big data, user, domain, and system. The phases of ML and the components of MLBiD provide directions for identification of associated opportunities and challenges and open up future work in many unexplored or under explored research areas.

**CHAPTER-3**

**SYSTEM ANALYSIS**

**3.1 EXISTING SYSTEM**

* Unfortunately, with the number of distributed workers increasing, it is hard to guarantee the security of each worker. This lack of security will increase the danger that attackers poison the dataset and manipulate the training result. Poisoning attack is a typical way to tamper the training data in machine learning. Especially in scenarios that newly generated datasets should be periodically sent to the distributed workers for updating the decision model, the attacker will have more chances to poison the datasets, leading to a more severe threat in DML.
* “Support Vector Machine” (SVM) is a supervised machine learning algorithm which can be used for both classification and 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 thehyper-plane that differentiates the two classes very well

**3.2 PROPOSED SYSTEM**

* DML into basic distributed machine learning (basic-DML) and semi distributed machine learning (semi-DML), depending on whether the center shares resources in the dataset training tasks. Then, we present data poison detection schemes for basic-DML and semi-DML respectively. The experimental results validate the effect of our proposed schemes.
* We classify DML into basic-DML and semi-DML, which are shown in Fig.1, respectively. Both of the two scenarios have a center, which contains a database, a computing server, and a parameter server. However, the center provides different functions in these two scenarios. In the basic-DML scenario, the center has no spare computing resource for sub-dataset training, and will send all the sub-datasets to the distributed workers. Therefore, in the basic-DML, the center only integrates the training results from distributed workers by the parameter server.

**3.3 Feasibility Study**

Preliminary investigation examine project feasibility, the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Operational and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

* Economical Feasibility
* Operational Feasibility
* Technical Feasibility

**Economic Feasibility**

A system can be developed technically and that will be used if installed must still be a good investment for the organization. In the economical feasibility, the development cost in creating the system is evaluated against the ultimate benefit derived from the new systems. Financial benefits must equal or exceed the costs.

The system is economically feasible. It does not require any addition hardware or software. Since the interface for this system is developed using the existing resources and technologies available at NIC, There is nominal expenditure and economical feasibility for certain.

**Operational Feasibility**

Proposed projects are beneficial only if they can be turned out into information system. That will meet the organization’s operating requirements. Operational feasibility aspects of the project are to be taken as an important part of the project implementation. Some of the important issues raised are to test the operational feasibility of a project includes the following: -

* Is there sufficient support for the management from the users?
* Will the system be used and work properly if it is being developed and implemented?
* Will there be any resistance from the user that will undermine the possible application benefits?

This system is targeted to be in accordance with the above-mentioned issues. Beforehand, the management issues and user requirements have been taken into consideration. So there is no question of resistance from the users that can undermine the possible application benefits.

The well-planned design would ensure the optimal utilization of the computer resources and would help in the improvement of performance status.

**Technical Feasibility**

The technical issue usually raised during the feasibility stage of the investigation includes the following:

* Does the necessary technology exist to do what is suggested?
* Do the proposed equipments have the technical capacity to hold the data required to use the new system?
* Will the proposed system provide adequate response to inquiries, regardless of the number or location of users?
* Can the system be upgraded if developed?
* Are there technical guarantees of accuracy, reliability, ease of access and data security?

Earlier no system existed to cater to the needs of ‘Secure Infrastructure Implementation System’. The current system developed is technically feasible. It is a web based user interface for audit workflow at NIC-CSD. Thus it provides an easy access to the users. The database’s purpose is to create, establish and maintain a workflow among various entities in order to facilitate all concerned users in their various capacities or roles. Permission to the users would be granted based on the roles specified. Therefore, it provides the technical guarantee of accuracy, reliability and security. The software and hard requirements for the development of this project are not many and are already available in-house at NIC or are available as free as open source. The work for the project is done with the current equipment and existing software technology. Necessary bandwidth exists for providing a fast feedback to the users irrespective of the number of users using the system.

**3.4 INPUT DESIGN**

The input design is the link between the information system and the user. It comprises the developing specification and procedures for data preparation and those steps are necessary to put transaction data in to a usable form for processing can be achieved by inspecting the computer to read data from a written or printed document or it can occur by having people keying the data directly into the system. The design of input focuses on controlling the amount of input required, controlling the errors, avoiding delay, avoiding extra steps and keeping the process simple. The input is designed in such a way so that it provides security and ease of use with retaining the privacy. Input Design considered the following things:

* What data should be given as input?
* How the data should be arranged or coded?
* The dialog to guide the operating personnel in providing input.
* Methods for preparing input validations and steps to follow when error occur.

**OBJECTIVES:**

1. Input Design is the process of converting a user-oriented description of the input into a computer-based system. This design is important to avoid errors in the data input process and show the correct direction to the management for getting correct information from the computerized system.

2. It is achieved by creating user-friendly screens for the data entry to handle large volume of data. The goal of designing input is to make data entry easier and to be free from errors. The data entry screen is designed in such a way that all the data manipulates can be performed. It also provides record viewing facilities.

3. When the data is entered it will check for its validity. Data can be entered with the help of screens. Appropriate messages are provided as when needed so that the user will not be in maize of instant. Thus the objective of input design is to create an input layout that is easy to follow

**3.5 OUTPUT DESIGN**

A quality output is one, which meets the requirements of the end user and presents the information clearly. In any system results of processing are communicated to the users and to other system through outputs. In output design it is determined how the information is to be displaced for immediate need and also the hard copy output. It is the most important and direct source information to the user. Efficient and intelligent output design improves the system’s relationship to help user decision-making.

1. Designing computer output should proceed in an organized, well thought out manner; the right output must be developed while ensuring that each output element is designed so that people will find the system can use easily and effectively. When analysis design computer output, they should Identify the specific output that is needed to meet the requirements.

2. Select methods for presenting information.

3. Create document, report, or other formats that contain information produced by the system.

The output form of an information system should accomplish one or more of the following objectives.

* Convey information about past activities, current status or projections of the
* Future.
* Signal important events, opportunities, problems, or warnings.
* Trigger an action.
* Confirm an action.

**CHAPTER-4**

**SYSTEM REQUIREMENTS SPECIFICATION**

**4.1 FUNCTIONAL REQUIREMENTS**

In [software engineering](https://en.wikipedia.org/wiki/Software_engineering) and [systems engineering](https://en.wikipedia.org/wiki/Systems_engineering), a **functional requirement** defines a function of a [system](https://en.wikipedia.org/wiki/System) or its component, where a function is described as a specification of behavior between outputs and inputs.[[1]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-FultonAirborne17-1)

Functional requirements may involve calculations, technical details, data manipulation and processing, and other specific functionality that define what a system is supposed to accomplish.[[2]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-2) Behavioral requirements describe all the cases where the system uses the functional requirements, these are captured in [use cases](https://en.wikipedia.org/wiki/Use_case). Functional requirements are supported by [non-functional requirements](https://en.wikipedia.org/wiki/Non-functional_requirement) (also known as "quality requirements"), which impose constraints on the design or implementation (such as performance requirements, security, or reliability). Generally, functional requirements are expressed in the form "system must do <requirement>," while non-functional requirements take the form "system shall be <requirement>." The plan for implementing functional requirements is detailed in the system design, whereas *non-functional* requirements are detailed in the system architecture.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)[[5]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-J%C3%B6nssonImpact06-5)

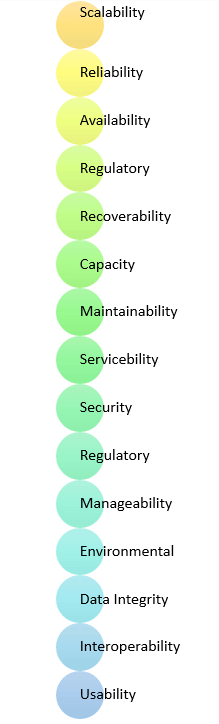
As defined in [requirements engineering](https://en.wikipedia.org/wiki/Requirements_analysis), functional requirements specify particular results of a system. This should be contrasted with non-functional requirements, which specify overall characteristics such as cost and [reliability](https://en.wikipedia.org/wiki/Reliability_engineering). Functional requirements drive the application architecture of a system, while non-functional requirements drive the technical architecture of a system.[[4]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-AdamsNon15-4)

In some cases a requirements analyst generates use cases after gathering and validating a set of functional requirements. The hierarchy of functional requirements collection and change, broadly speaking, is: user/[stakeholder](https://en.wikipedia.org/wiki/Project_stakeholder) request → analyze → use case → incorporate. Stakeholders make a request; systems engineers attempt to discuss, observe, and understand the aspects of the requirement; use cases, entity relationship diagrams, and other models are built to validate the requirement; and, if documented and approved, the requirement is implemented/incorporated.[[6]](https://en.wikipedia.org/wiki/Functional_requirement#cite_note-MITRESys14-6) Each use case illustrates behavioral scenarios through one or more functional requirements. Often, though, an analyst will begin by eliciting a set of use cases, from which the analyst can derive the functional requirements that must be implemented to allow a user to perform each use case.

**4.2 NON-FUNCTIONAL REQUIREMENT**

**NON-FUNCTIONAL REQUIREMENT** (NFR) specifies the quality attribute of a software system. They judge the software system based on Responsiveness, Usability, Security, Portability and other non-functional standards that are critical to the success of the software system. Example of nonfunctional requirement, *“how fast does the website load?”* Failing to meet non-functional requirements can result in systems that fail to satisfy user needs.

Non-functional Requirements allows you to impose constraints or restrictions on the design of the system across the various agile backlogs. Example, the site should load in 3 seconds when the number of simultaneous users are> 10000. Description of non-functional requirements is just as critical as a functional requirement.



* Usability requirement
* Serviceability requirement
* Manageability requirement
* Recoverability requirement
* Security requirement
* Data Integrity requirement
* Capacity requirement
* Availability requirement
* Scalability requirement
* Interoperability requirement
* Reliability requirement
* Maintainability requirement
* Regulatory requirement
* Environmental requirement

**Advantages of Non-Functional Requirement**

Benefits/pros of Non-functional testing are:

* The nonfunctional requirements ensure the software system follow legal and compliance rules.
* They ensure the reliability, availability, and performance of the software system
* They ensure good user experience and ease of operating the software.
* They help in formulating security policy of the software system.

**Disadvantages of Non-functional requirement**

Cons/drawbacks of Non-function requirement are:

* None functional requirement may affect the various high-level software subsystem
* They require special consideration during the software architecture/high-level design phase which increases costs.
* Their implementation does not usually map to the specific software sub-system,
* It is tough to modify non-functional once you pass the architecture phase.

**KEY LEARNING**

* A non-functional requirement defines the performance attribute of a software system.
* Types of Non-functional requirement are Scalability Capacity, Availability, Reliability, Recoverability, Data Integrity, etc.
* Example of Non Functional Requirement is Employees never allowed to update their salary information. Such attempt should be reported to the security administrator.
* Functional Requirement is a verb while Non-Functional Requirement is an attribute
* The advantage of Non-functional requirement is that it helps you to ensure good user experience and ease of operating the software

The biggest disadvantage of Non-functional requirement is that it may affect the various high-level software subsystems.

**CHAPTER-5**

**SYSTEM DESIGN**

**5.1 SYSTEM SPECIFICATIONS**

**Hardware Requirements:**

* System : Pentium i3
* Hard Disk : 500 GB.
* Monitor : 14’ Colour Monitor.
* Mouse : Optical Mouse.
* Ram : 4 GB.

**Software Requirements:**

* Operating system : Windows 8/10.
* Coding Language : PYTHON

**5.2 SYSTEM ARCHITECTURE**



Fig 5.2 System Architecture

**5.2 UML DIAGRAMS**

UML stands for Unified Modeling Language. UML is a standardized general-purpose modeling language in the field of object-oriented software engineering. The standard is managed, and was created by, the Object Management Group.

The goal is for UML to become a common language for creating models of object oriented computer software. In its current form UML is comprised of two major components: a Meta-model and a notation. In the future, some form of method or process may also be added to; or associated with, UML.

The Unified Modeling Language is a standard language for specifying, Visualization, Constructing and documenting the artifacts of software system, as well as for business modeling and other non-software systems.

The UML represents a collection of best engineering practices that have proven successful in the modeling of large and complex systems.

The UML is a very important part of developing objects oriented software and the software development process. The UML uses mostly graphical notations to express the design of software projects.

**GOALS:**

The Primary goals in the design of the UML are as follows:

1. Provide users a ready-to-use, expressive visual modeling Language so that they can develop and exchange meaningful models.
2. Provide extendibility and specialization mechanisms to extend the core concepts.
3. Be independent of particular programming languages and development process.
4. Provide a formal basis for understanding the modeling language.
5. Encourage the growth of OO tools market.
6. Support higher level development concepts such as collaborations, frameworks, patterns and components.
7. Integrate best practices.

**5.2.1 USE CASE DIAGRAM**

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.





Fig 5.2.1 Use Case Diagram

**5.2.2 CLASS DIAGRAM**

In software engineering, a class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.



Fig 5.2.2 Class Diagram

**5.2.3 SEQUENCE DIAGRAM**

A sequence diagram in Unified Modeling Language (UML) is a kind of interaction diagram that shows how processes operate with one another and in what order. It is a construct of a Message Sequence Chart. Sequence diagrams are sometimes called event diagrams, event scenarios, and timing diagrams.



Fig 5.2.3 Sequence Diagram

**5.2.4 ACTIVITY DIAGRAM:**

Activity diagrams are graphical representations of workflows of stepwise activities and actions with support for choice, iteration and concurrency. In the Unified Modeling Language, activity diagrams can be used to describe the business and operational step-by-step workflows of components in a system. An activity diagram shows the overall flow of control.

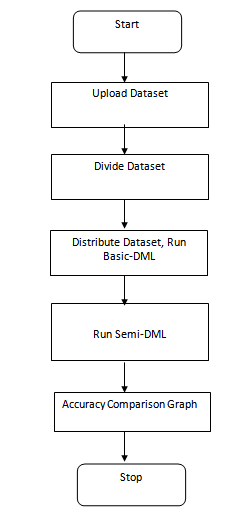


Fig 5.2.4 Activity Diagram

**5.2.5 COLLABORATION DIAGRAM**

The collaboration diagram is used to show the relationship between the objects in a system. Both the sequence and the collaboration diagrams represent the same information but differently. Instead of showing the flow of messages, it depicts the architecture of the object residing in the system as it is based on object-oriented programming. An object consists of several features. Multiple objects present in the system are connected to each other. The collaboration diagram, which is also known as a communication diagram, is used to portray the object's architecture in the system.



Fig 5.2.5 Collaboration Diagram

**CHAPTER-6**

**SYSTEM IMPLEMENTATION**

**6.1 MODULES DESCRIPTION**

**Worker1:**

This is a worker node which accept divided dataset from center server and then build existing SVM model and Basic DML model and then calculate accuracy of both algorithms and send result back to center server

**Worker2:**

This is another worker node which accept other half of dataset and then run existing SVM and Basic DML and send accuracy back to center server.

**CenterServer:**

This is a center server which upload dataset to application and then divide dataset into two equal parts and then distribute each part to worker 1 and 2 and then collect result. This server will run semi DML and calculate its accuracy also.

In machine learning applications we will train model by using dataset and then this model can be used to predict classes from new data. Machine learning model can be trained in two ways such as in local machine or distributed machine and in distributed machines dataset will be divided and share between multiple nodes and all nodes will process the dataset and then build ML model and send result back to centralized server.

In distributed environment sometime attackers may modify training data and then make ML to predict wrong result and to detect and remove such modified data author is using Data Poison Detection technique. This technique will inspect training dataset to identify odd values and then remove it. By applying Data Poison technique we can improve accuracy of ML algorithms.

In propose paper author using two distributed techniques called Basic DML and Semi DML where Basic DML will divide dataset into multiple parts and send to worker nodes and worker nodes build ML model and send result back distributed center server. In Semi DML center server itself will devote resource to ML model to train dataset.

To implement this project we have compared performance of existing SVM and DML where existing SVM will not apply Data Poison Detection technique and DML technique will apply Data Poison technique.

**6.2 SAMPLE CODE**

from tkinter import messagebox

from tkinter import \*

from tkinter import simpledialog

import tkinter

from tkinter import filedialog

import matplotlib.pyplot as plt

import numpy as np

from tkinter.filedialog import askopenfilename

import os

import re

import numpy as np

import pandas as pd

import socket

import json

from sklearn.ensemble import IsolationForest

from sklearn.metrics import accuracy\_score

from sklearn import svm

from sklearn.model\_selection import train\_test\_split

main = tkinter.Tk()

main.title("Data Poison Detection Schemes for Distributed Machine Learning")

main.geometry("1300x1200")

global filename

global svm\_acc,basic\_acc,semi\_acc

global part1,part2

global first,second

def upload():

global filename

filename = filedialog.askopenfilename(initialdir="dataset")

pathlabel.config(text=filename)

text.delete('1.0', END)

text.insert(END,filename+" loaded\n");

def divideDataset():

global first,second

global part1,part2

global header

text.delete('1.0', END)

part1 = ''

part2 = ''

header = ''

dataset = pd.read\_csv(filename)

divide\_size = len(dataset)

text.insert(END,"Dataset contains total records : "+str(divide\_size)+"\n")

p1 = divide\_size / 2

p2 = p1

if (p1+p2) != divide\_size:

p2 = p2 + 1

count = 0

text.insert(END,"Worker1 divided dataset total records : "+str(p1)+"\n")

text.insert(END,"Worker2 divided dataset total records : "+str(p2)+"\n")

with open(filename, "r") as file:

for line in file:

if len(header) == 0:

header = str(line)

print(header)

else:

line = line.strip('\n')

line = line.strip()

if count < p1:

part1+=line+"\n"

else:

part2+=line+"\n"

count = count + 1

file.close()

first = header+part1

second = header+part2

def runBasic():

global first,second

text.delete('1.0', END)

global svm\_acc,basic\_acc

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(('localhost', 2222))

jsondata = json.dumps({"type":"basicDML","dataset": first})

message = client.send(jsondata.encode())

data = client.recv(1000)

data = json.loads(data.decode())

svm\_acc = float(str(data.get("existing")))

basic\_acc = float(str(data.get("dml")))

text.insert(END,"Existing SVM Accuracy Received from Worker1 : "+str(svm\_acc)+"\n")

text.insert(END,"DML SVM Accuracy Received from Worker1 : "+str(basic\_acc)+"\n")

client = socket.socket(socket.AF\_INET, socket.SOCK\_STREAM)

client.connect(('localhost', 3333))

jsondata = json.dumps({"type":"basicDML","dataset": first})

message = client.send(jsondata.encode())

data = client.recv(1000)

data = json.loads(data.decode())

svm\_acc1 = float(str(data.get("existing")))

basic\_acc1 = float(str(data.get("dml")))

text.insert(END,"Existing SVM Accuracy Received from Worker2 : "+str(svm\_acc1)+"\n")

text.insert(END,"DML SVM Accuracy Received from Worker2 : "+str(basic\_acc1)+"\n")

svm\_acc = svm\_acc + svm\_acc1

basic\_acc = basic\_acc + basic\_acc1

svm\_acc = svm\_acc / 2

basic\_acc = basic\_acc / 2

text.insert(END,"Existing SVM Total Accuracy : "+str(svm\_acc)+"\n")

text.insert(END,"DML SVM Total Accuracy : "+str(basic\_acc)+"\n")

def runSemi():

global semi\_acc

dataset = pd.read\_csv(filename)

dataset = dataset.values

X, Y = dataset[:, :-1], dataset[:, -1]

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, Y, test\_size=0.2)

iso = IsolationForest(contamination=0.1)

yhat = iso.fit\_predict(X\_train)

mask = yhat != -1

X\_train, y\_train = X\_train[mask, :], y\_train[mask]

cls = svm.SVC()

cls.fit(X\_train, y\_train)

prediction\_data = cls.predict(X\_test)

semi\_acc = accuracy\_score(y\_test,prediction\_data)\*100

text.insert(END,"Semi-DML Accuracy : "+str(semi\_acc)+"\n")

def graph():

height = [svm\_acc,basic\_acc,semi\_acc]

bars = ('Existing SVM Accuracy', 'Basic DML Accuracy','Semi DML Accuracy')

y\_pos = np.arange(len(bars))

plt.bar(y\_pos, height)

plt.xticks(y\_pos, bars)

plt.show()

def close():

main.destroy()

font = ('times', 14, 'bold')

title = Label(main, text='Data Poison Detection Schemes for Distributed Machine Learning')

title.config(bg='yellow3', fg='white')

title.config(font=font)

title.config(height=3, width=120)

title.place(x=0,y=5)

font1 = ('times', 13, 'bold')

uploadButton = Button(main, text="Upload Dataset", command=upload)

uploadButton.place(x=50,y=100)

uploadButton.config(font=font1)

pathlabel = Label(main)

pathlabel.config(bg='brown', fg='white')

pathlabel.config(font=font1)

pathlabel.place(x=460,y=100)

divideButton = Button(main, text="Divide Dataset", command=divideDataset)

divideButton.place(x=50,y=150)

divideButton.config(font=font1)

basicButton = Button(main, text="Distribute Dataset & Run Basic-DML", command=runBasic)

basicButton.place(x=310,y=150)

basicButton.config(font=font1)

semi = Button(main, text="Run Semi-DML", command=runSemi)

semi.place(x=650,y=150)

semi.config(font=font1)

graphbutton = Button(main, text="Accuracy Comparison Graph", command=graph)

graphbutton.place(x=50,y=200)

graphbutton.config(font=font1)

exitb = Button(main, text="Exit", command=close)

exitb.place(x=310,y=200)

exitb.config(font=font1)

font1 = ('times', 12, 'bold')

text=Text(main,height=20,width=150)

scroll=Scrollbar(text)

text.configure(yscrollcommand=scroll.set)

text.place(x=10,y=250)

text.config(font=font1)

main.config(bg='burlywood2')

main.mainloop()

### 6.3 SYSTEM TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

**6.4 TYPES OF TESTS**

**Unit testing**

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

**Integration testing**

Integration tests are designed to test integrated software components to determine if they actually run as one program. Testing is event driven and is more concerned with the basic outcome of screens or fields. Integration tests demonstrate that although the components were individually satisfaction, as shown by successfully unit testing, the combination of components is correct and consistent. Integration testing is specifically aimed at exposing the problems that arise from the combination of components.

**Functional test**

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centered on the following items:

Valid Input : identified classes of valid input must be accepted.

Invalid Input : identified classes of invalid input must be rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs must be exercised.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

**System Test**

System testing ensures that the entire integrated software system meets requirements. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

**White Box Testing**

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

**Black Box Testing**

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document. It is a testing in which the software under test is treated, as a black box .you cannot “see” into it. The test provides inputs and responds to outputs without considering how the software works.

**Unit Testing:**

Unit testing is usually conducted as part of a combined code and unit test phase of the software lifecycle, although it is not uncommon for coding and unit testing to be conducted as two distinct phases.

**Test strategy and approach**

Field testing will be performed manually and functional tests will be written in detail.

**Test objectives**

* All field entries must work properly.
* Pages must be activated from the identified link.
* The entry screen, messages and responses must not be delayed.

**Features to be tested**

* Verify that the entries are of the correct format
* No duplicate entries should be allowed
* All links should take the user to the correct page.

# Integration Testing

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects.

The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

**Acceptance Testing**

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

**Test Results:** All the test cases mentioned above passed successfully. No defects encountered.

### 6.5 TESTING METHODOLOGIES

The following are the Testing Methodologies:

* **Unit Testing.**
* **Integration Testing.**
* **User Acceptance Testing.**
* **Output Testing.**
* **Validation Testing.**

**Unit Testing**

Unit testing focuses verification effort on the smallest unit of Software design that is the module. Unit testing exercises specific paths in a module’s control structure to ensure complete coverage and maximum error detection. This test focuses on each module individually, ensuring that it functions properly as a unit. Hence, the naming is Unit Testing.

During this testing, each module is tested individually and the module interfaces are verified for the consistency with design specification. All important processing path are tested for the expected results. All error handling paths are also tested.

**Integration Testing**

Integration testing addresses the issues associated with the dual problems of verification and program construction. After the software has been integrated a set of high order tests are conducted. The main objective in this testing process is to take unit tested modules and builds a program structure that has been dictated by design.

**The following are the types of Integration Testing:**

**1)Top Down Integration**

This method is an incremental approach to the construction of program structure. Modules are integrated by moving downward through the control hierarchy, beginning with the main program module. The module subordinates to the main program module are incorporated into the structure in either a depth first or breadth first manner.

In this method, the software is tested from main module and individual stubs are replaced when the test proceeds downwards.

**2. Bottom-up Integration**

This method begins the construction and testing with the modules at the lowest level in the program structure. Since the modules are integrated from the bottom up, processing required for modules subordinate to a given level is always available and the need for stubs is eliminated. The bottom up integration strategy may be implemented with the following steps:

* The low-level modules are combined into clusters into clusters that perform a specific Software sub-function.
* A driver (i.e.) the control program for testing is written to coordinate test case input and output.
* The cluster is tested.
* Drivers are removed and clusters are combined moving upward in the program

Structure

The bottom up approaches tests each module individually and then each module is module is integrated with a main module and tested for functionality.

**User Acceptance Testing**

User Acceptance of a system is the key factor for the success of any system. The system under consideration is tested for user acceptance by constantly keeping in touch with the prospective system users at the time of developing and making changes wherever required. The system developed provides a friendly user interface that can easily be understood even by a person who is new to the system.

**Output Testing**

After performing the validation testing, the next step is output testing of the proposed system, since no system could be useful if it does not produce the required output in the specified format. Asking the users about the format required by them tests the outputs generated or displayed by the system under consideration. Hence the output format is considered in 2 ways – one is on screen and another in printed format.

**Validation Checking**

Validation checks are performed on the following fields.

**Text Field:**

The text field can contain only the number of characters lesser than or equal to its size. The text fields are alphanumeric in some tables and alphabetic in other tables. Incorrect entry always flashes and error message.

**Numeric Field:**

The numeric field can contain only numbers from 0 to 9. An entry of any character flashes an error messages. The individual modules are checked for accuracy and what it has to perform. Each module is subjected to test run along with sample data. The individually tested modules are integrated into a single system. Testing involves executing the real data information is used in the program the existence of any program defect is inferred from the output. The testing should be planned so that all the requirements are individually tested. A successful test is one that gives out the defects for the inappropriate data and produces and output revealing the errors in the system.

**Preparation of Test Data**

Taking various kinds of test data does the above testing. Preparation of test data plays a vital role in the system testing. After preparing the test data the system under study is tested using that test data. While testing the system by using test data errors are again uncovered and corrected by using above testing steps and corrections are also noted for future use.

**Using Live Test Data:**

Live test data are those that are actually extracted from organization files. After a system is partially constructed, programmers or analysts often ask users to key in a set of data from their normal activities. Then, the systems person uses this data as a way to partially test the system. In other instances, programmers or analysts extract a set of live data from the files and have them entered themselves.

It is difficult to obtain live data in sufficient amounts to conduct extensive testing. And, although it is realistic data that will show how the system will perform for the typical processing requirement, assuming that the live data entered are in fact typical, such data generally will not test all combinations or formats that can enter the system. This bias toward typical values then does not provide a true systems test and in fact ignores the cases most likely to cause system failure.

**Using Artificial Test Data:**

Artificial test data are created solely for test purposes, since they can be generated to test all combinations of formats and values. In other words, the artificial data, which can quickly be prepared by a data generating utility program in the information systems department, make possible the testing of all login and control paths through the program.

The most effective test programs use artificial test data generated by persons other than those who wrote the programs. Often, an independent team of testers formulates a testing plan, using the systems specifications.

The package “Virtual Private Network” has satisfied all the requirements specified as per software requirement specification and was accepted.

**USER TRAINING**

Whenever a new system is developed, user training is required to educate them about the working of the system so that it can be put to efficient use by those for whom the system has been primarily designed. For this purpose the normal working of the project was demonstrated to the prospective users. Its working is easily understandable and since the expected users are people who have good knowledge of computers, the use of this system is very easy.

**MAINTAINENCE**

This covers a wide range of activities including correcting code and design errors. To reduce the need for maintenance in the long run, we have more accurately defined the user’s requirements during the process of system development. Depending on the requirements, this system has been developed to satisfy the needs to the largest possible extent. With development in technology, it may be possible to add many more features based on the requirements in future. The coding and designing is simple and easy to understand which will make maintenance easier.

**TESTING STRATEGY :**

A strategy for system testing integrates system test cases and design techniques into a well planned series of steps that results in the successful construction of software. The testing strategy must co-operate test planning, test case design, test execution, and the resultant data collection and evaluation .A strategy for software testing must accommodate low-level tests that are necessary to verify that a small source code segment has been correctly implemented as well as high level tests that validate major system functions against user requirements.

Software testing is a critical element of software quality assurance and represents the ultimate review of specification design and coding. Testing represents an interesting anomaly for the software. Thus, a series of testing are performed for the proposed system before the system is ready for user acceptance testing.

**SYSTEM TESTING:**

Software once validated must be combined with other system elements (e.g. Hardware, people, database). System testing verifies that all the elements are proper and that overall system function performance is achieved. It also tests to find discrepancies between the system and its original objective, current specifications and system documentation.

**UNIT TESTING:**

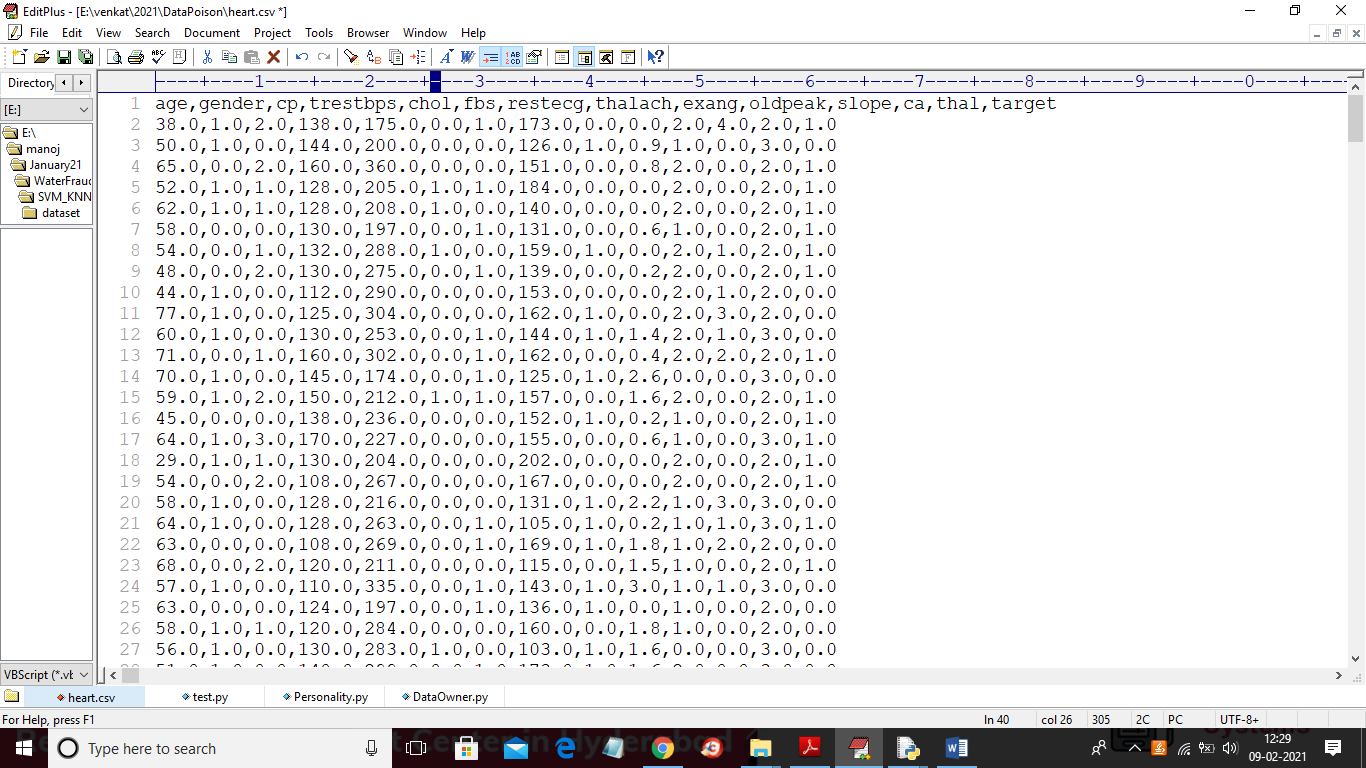
In unit testing different are modules are tested against the specifications produced during the design for the modules. Unit testing is essential for verification of the code produced during the coding phase, and hence the goals to test the internal logic of the modules. Using the detailed design description as a guide, important Conrail paths are tested to uncover errors within the boundary of the modules. This testing is carried out during the programming stage itself. In this type of testing step, each module was found to be working satisfactorily as regards to the expected output from the module.

In Due Course, latest technology advancements will be taken into consideration. As part of technical build-up many components of the networking system will be generic in nature so that future projects can either use or interact with this.The future holds a lot to offer to the development and refinement of this project.

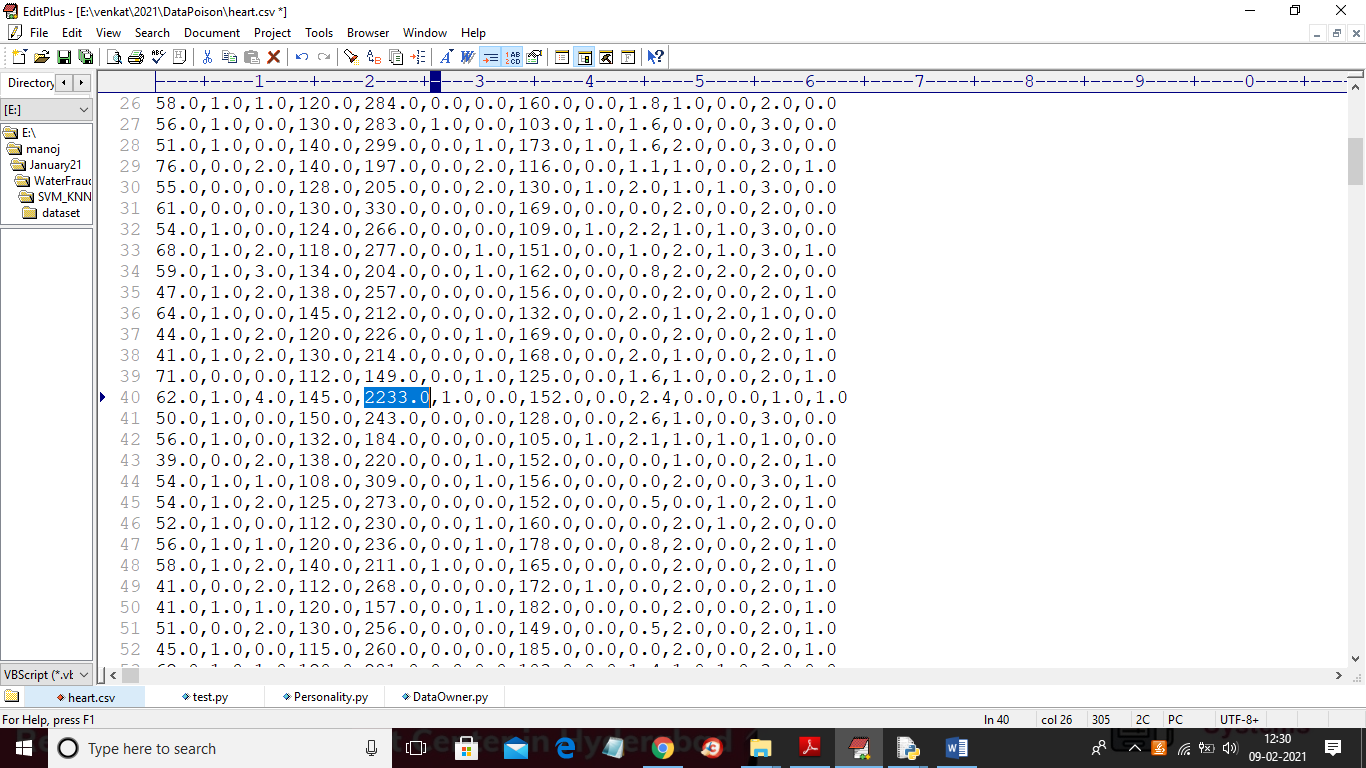
**CHAPTER-7**

**SCREEN SHOTS**

To implement this project we have used heart disease dataset and in below dataset screen we can see dataset contains invalid data which called as Data Poision.

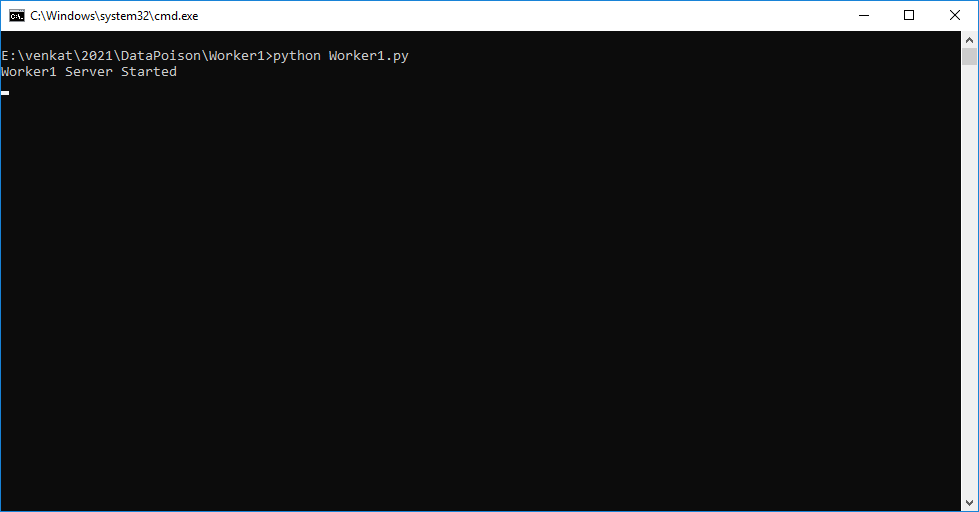


In above screen heart dataset first row contains column names and remaining rows are the column values and in below dataset screen we can see odd or invalid value

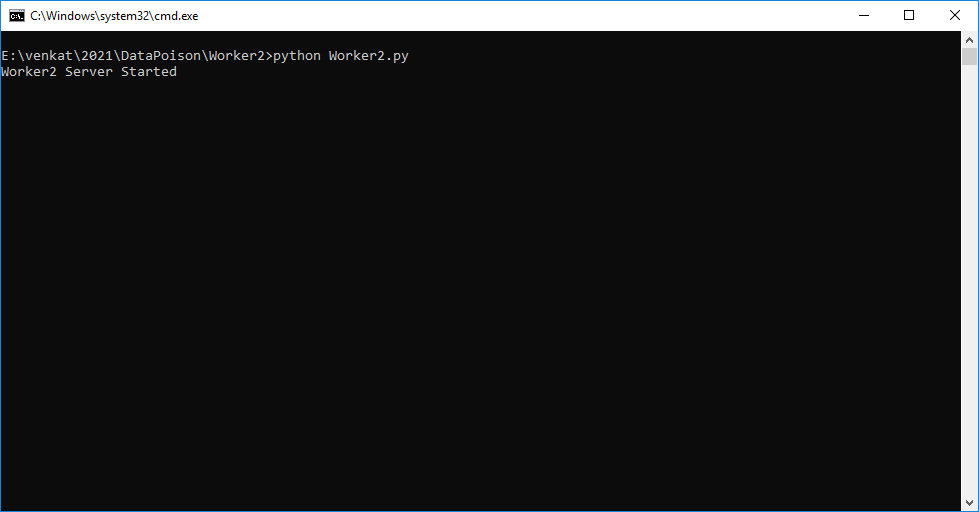


In above screen in selected blue value we can see recorded blood pressure value as 2233 which is wrong value and if ML train on such data then it may predict wrong result and it will reduce prediction accuracy and to avoid such problem we can apply Data Poison Detection technique. In python we can ‘IsolationForest’ class to detect and remove such poison data.

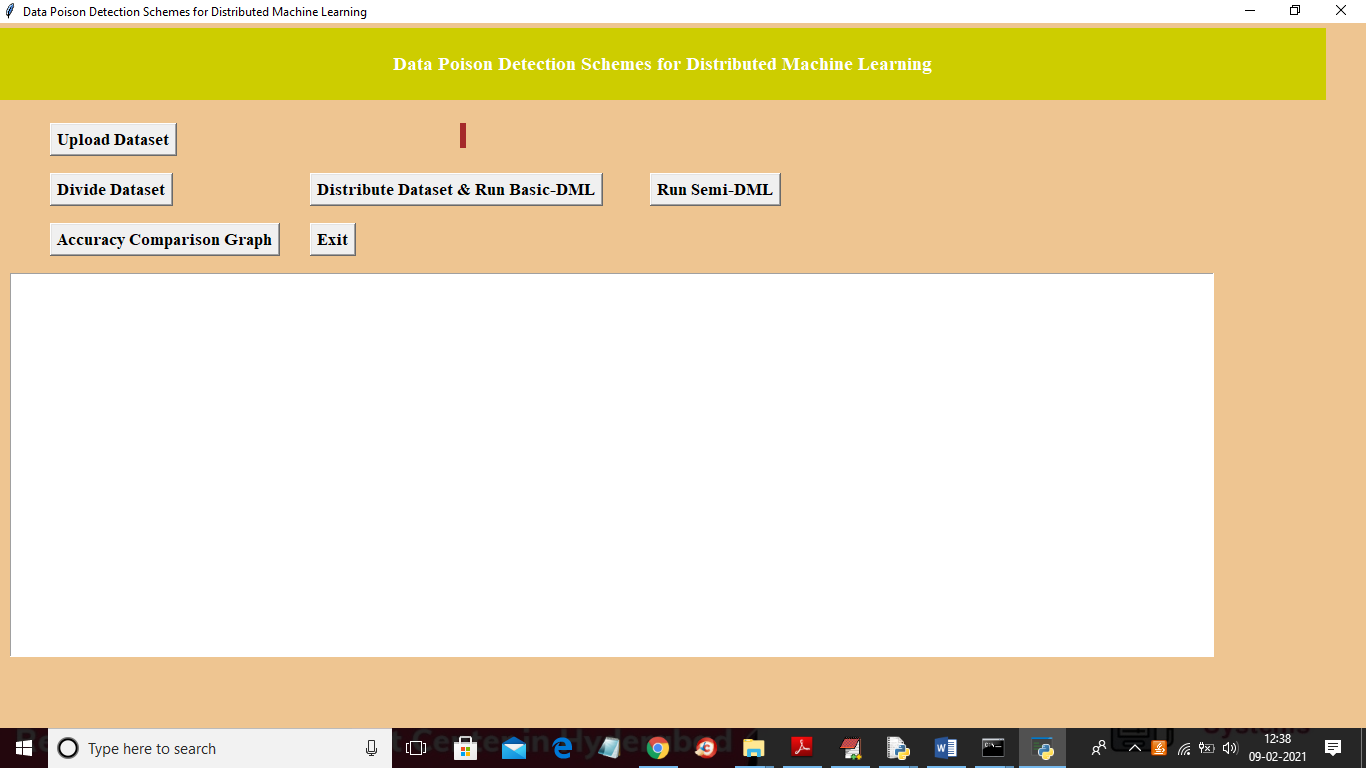
To run project first double click on ‘run.bat’ file from Worker1 folder to start worker 1 node and to get below screen



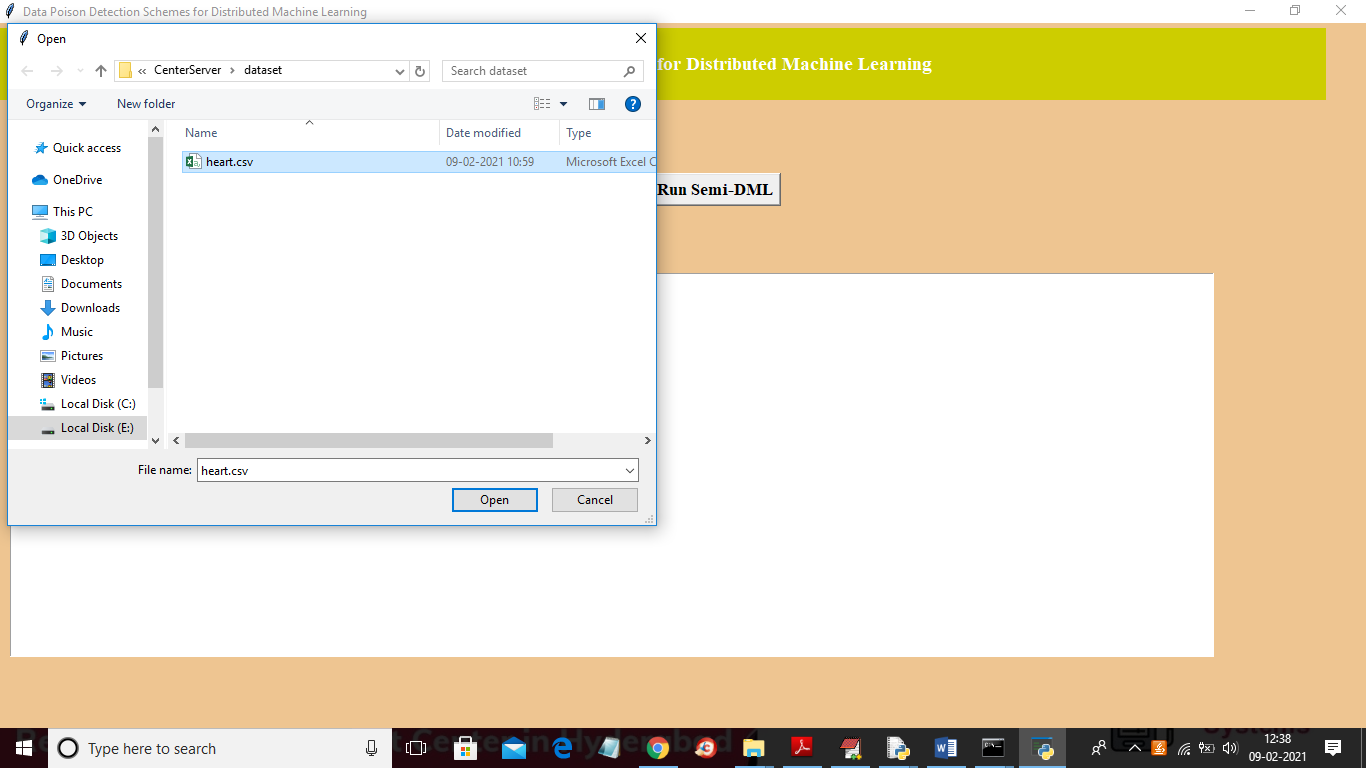
In above screen worker 1 server started and now double click on ‘run.bat’ file from worker2 folder to start worker 2



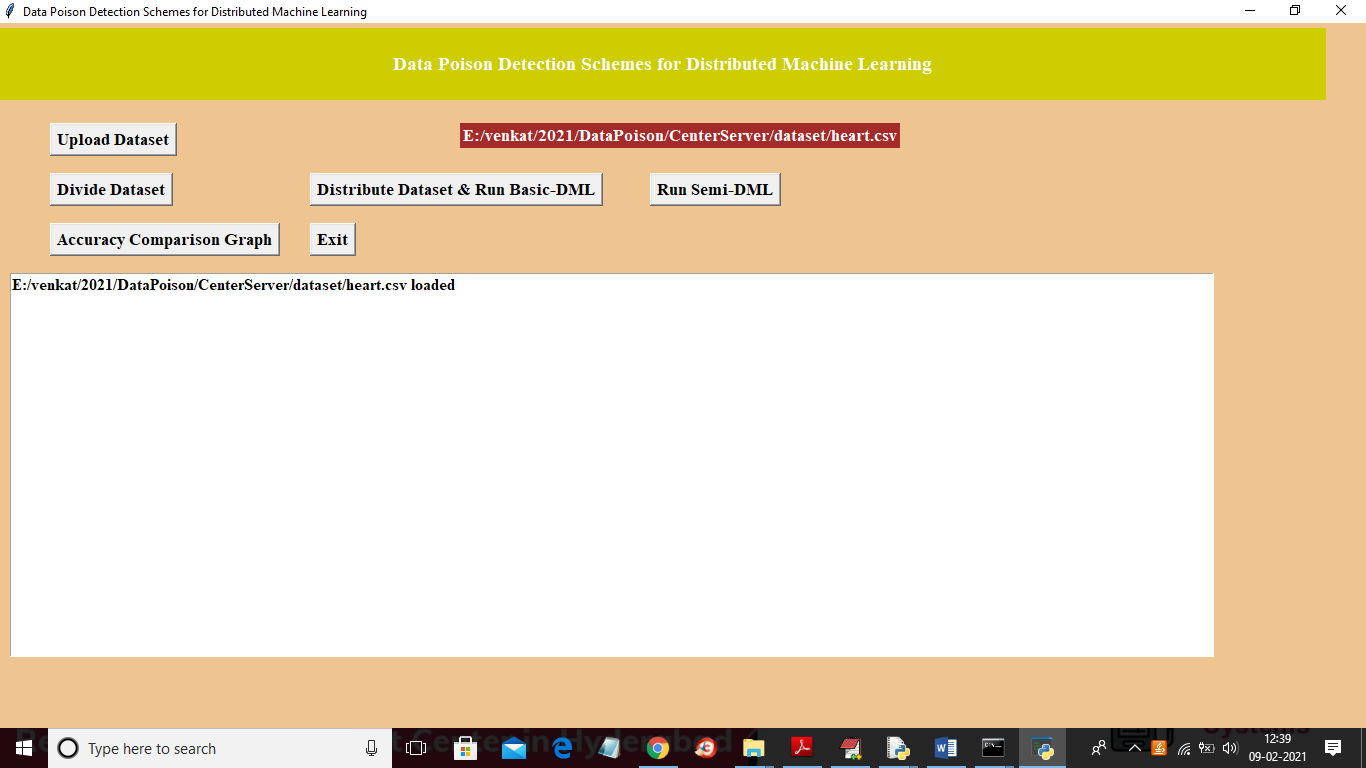
In above screen worker2 server started and now double click on ‘run.bat’ file from ‘CenterServer’ folder to start distributed server and to get below screen



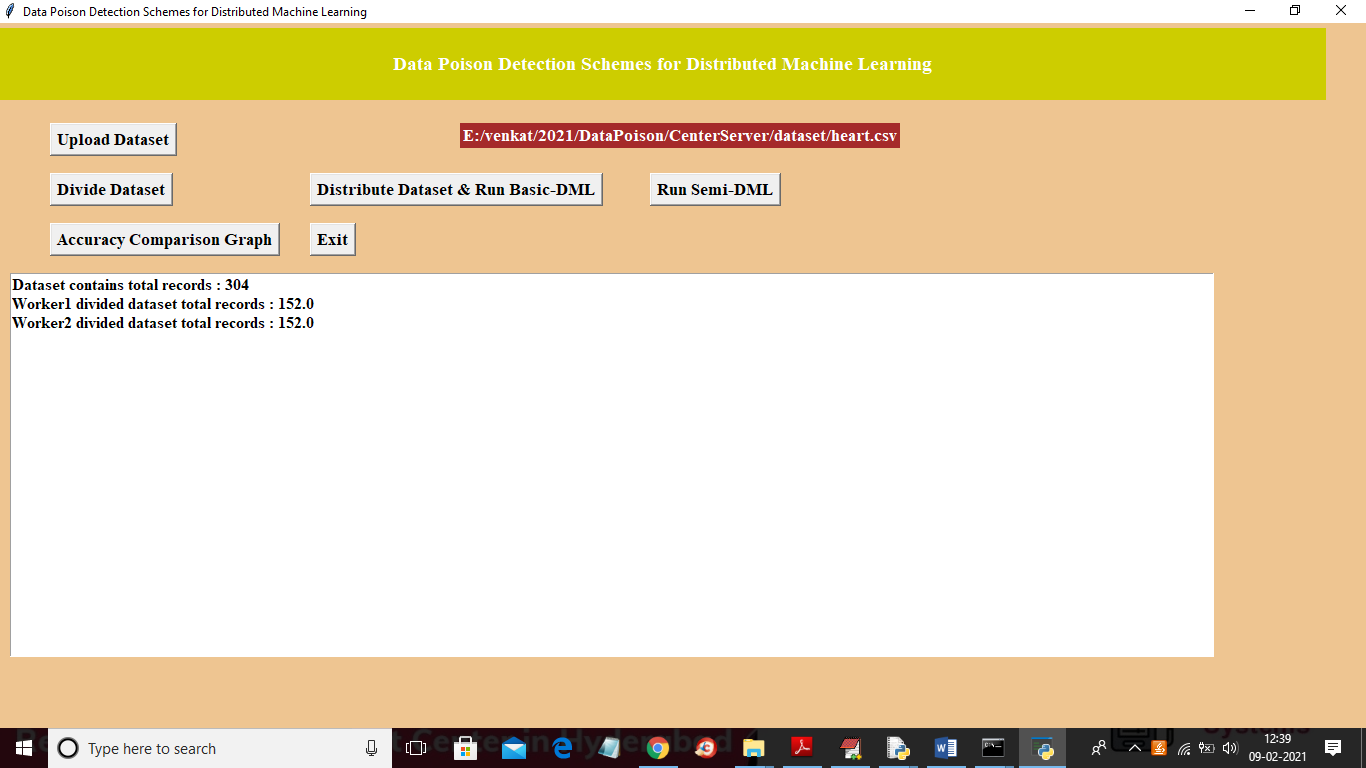
In above screen click on ‘Upload Dataset’ button to upload dataset and to get below screen



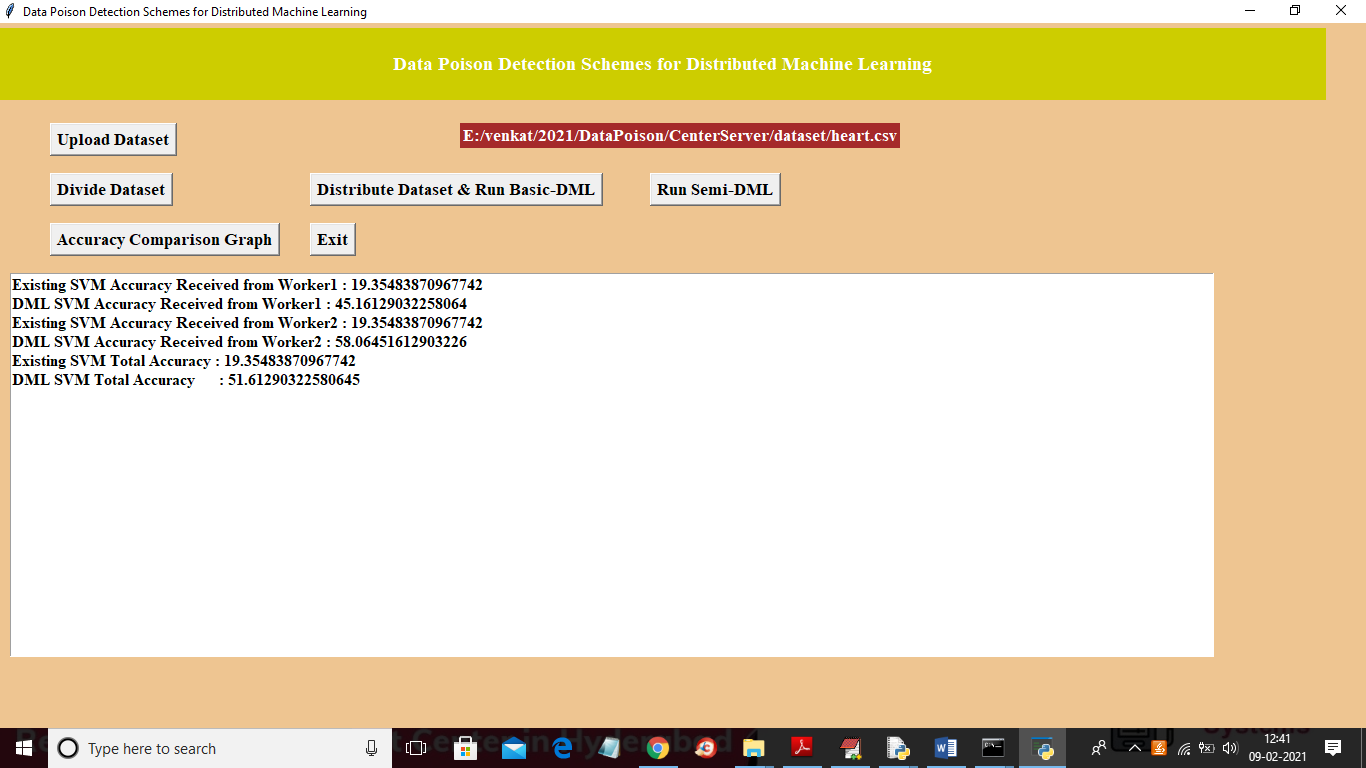
In above screen selecting and uploading ‘heart.csv’ file and then click on ‘Open’ button to load dataset and to get below screen



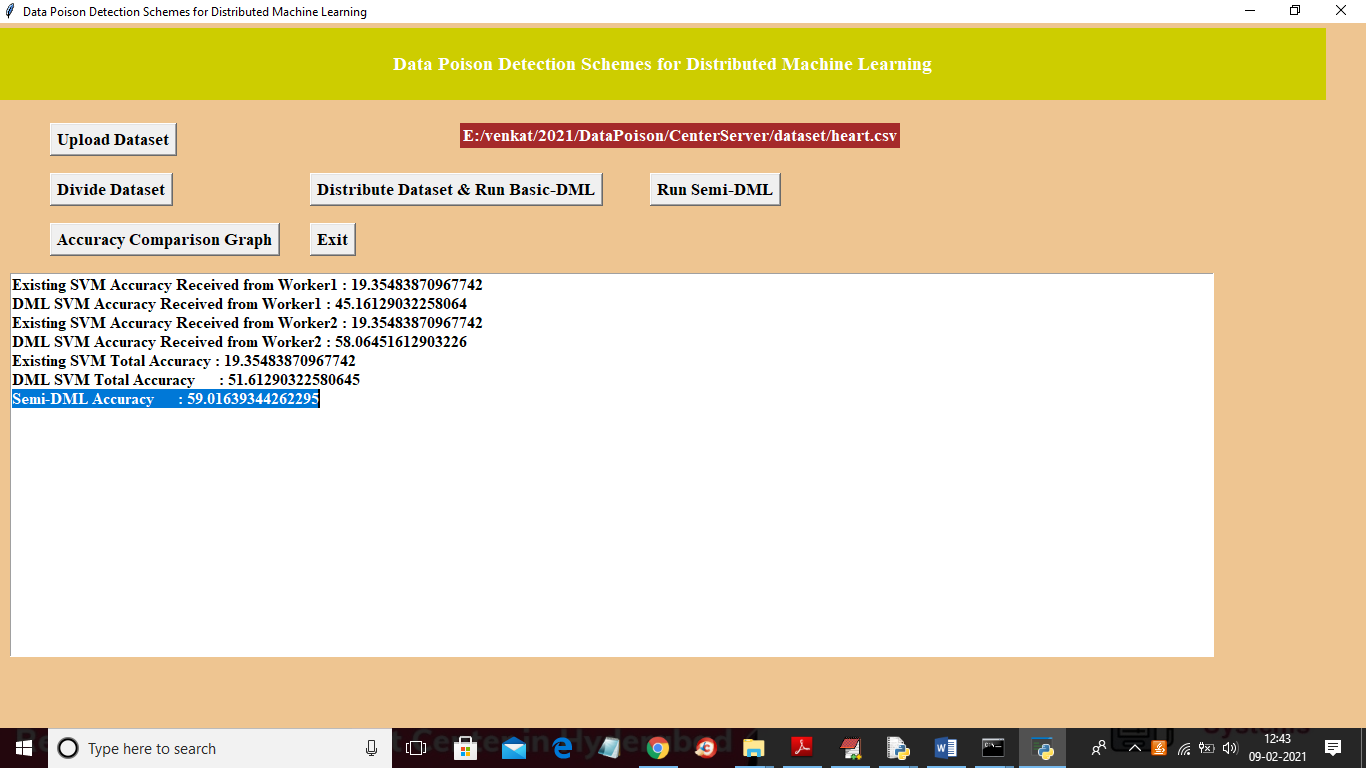
In above screen dataset loaded and now click on ‘Divide Dataset’ button to divide dataset into 2 equal parts



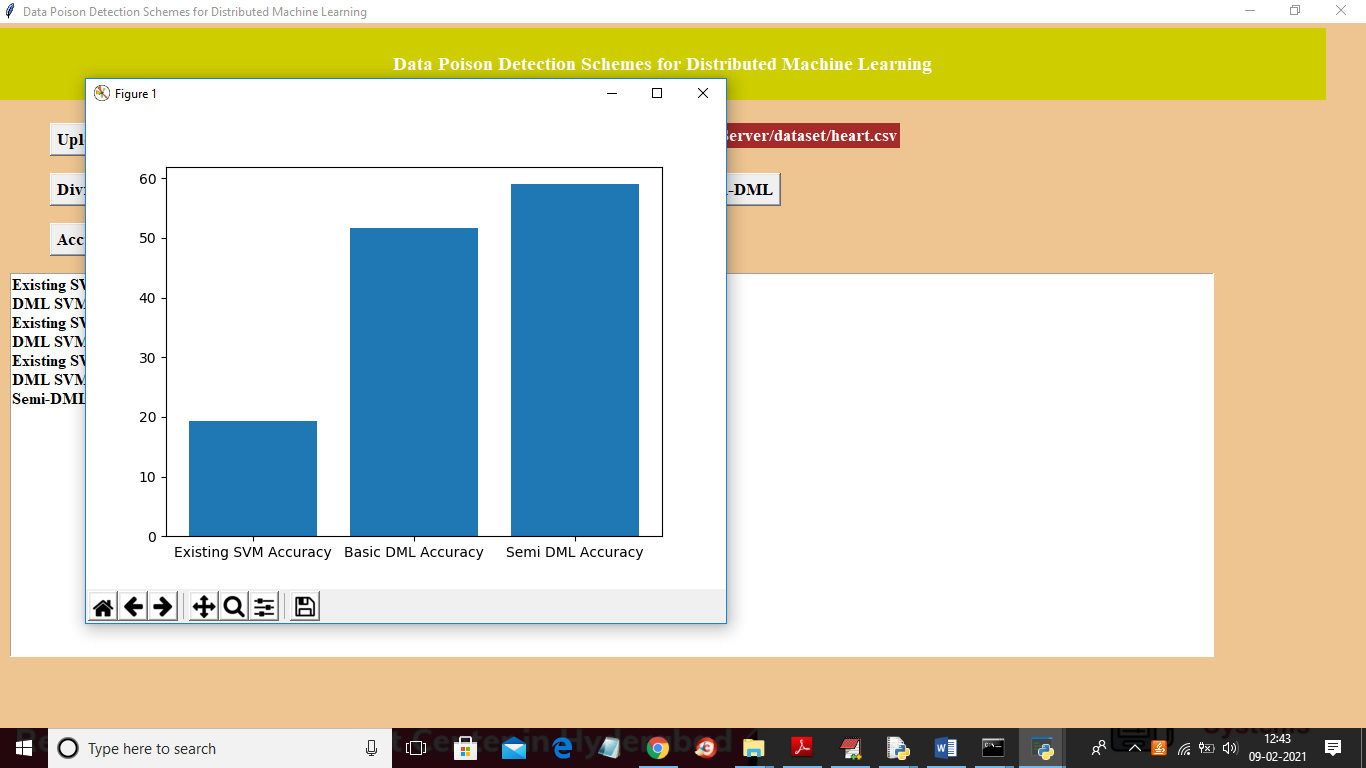
In above screen dataset contains 304 records and equally distributed to 2 parts and now click on ‘Distribute Dataset & Run Basic-DML’ button to distribute dataset to 2 workers and then get accuracy result



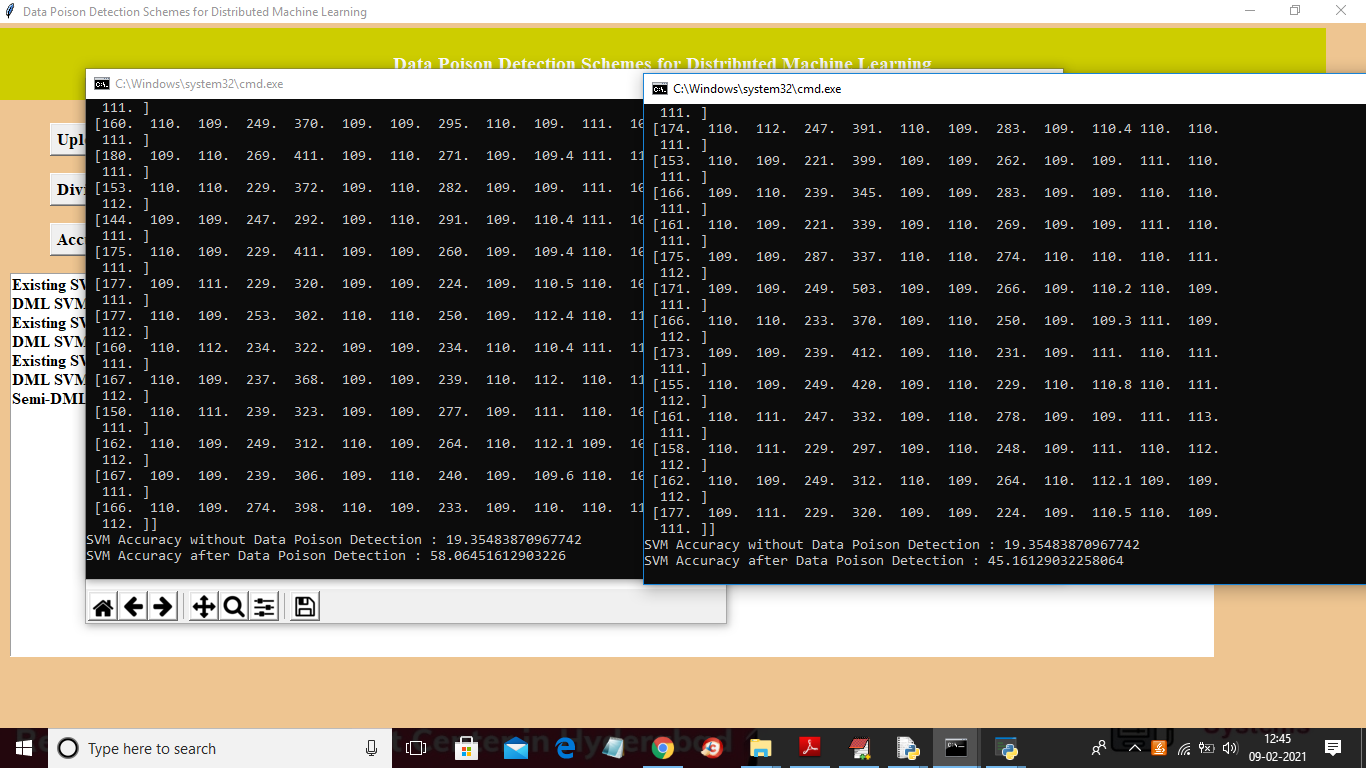
In above screen we got result from 2 worker nodes for existing SVM accuracy and propose DML accuracy and in above screen we can see existing SVM accuracy is 19% when data poison exists in dataset and after removing data poison using DML technique we got 51% accuracy and now click on ‘Run Semi-DML’ button to allow center server to devote resources to DML and then remove poison from dataset and then calculate accuracy.



In above screen Semi-DML accuracy is 59% and now click on ‘Accuracy Comparison Graph’ button to get below graph



In above screen x-axis contains algorithm name and y-axis represents accuracy and from above graph we can conclude that Basic-DML and Semi-DML accuracy is better than existing SVM accuracy. In below worker screens also we can see accuracy values



**CHAPTER-8**

**CONCLUSION**

In this project, we classify DML into basic-DML and semi-DML. In basic-DML, the center server dispatches learning tasks to distributed machines and aggregates their learning results. While in semi-DML, the center server further devotes resources into dataset learning in addition to its duty in basic-DML. We firstly put forward a novel data poison detection scheme for basic-DML, which utilizes a cross-learning mechanism to find out the poisoned data. We prove that the proposed cross-learning mechanism would generate training loops, based on which a mathematical model is established to find the optimal number of training loops. Then, for semi-DML, we present an improved data poison detection scheme to provide better learning protection with the aid of the central resource. To efficiently utilize the system resources, an optimal resource allocation approach is developed. Simulation results show that the proposed scheme can significantly improve the accuracy of the final model by up to 20% for support vector machine and 60% for logistic regression in the basic-DML scenario. Moreover, in the semi-DML scenario, the improved data poison detection scheme with optimal resource allocation can decrease the wasted resources for 20-100%.

**APPENDIX-A**

**BIBLIOGRAPHY**

[1] G. Qiao, S. Leng, K. Zhang, and Y. He, “Collaborative task offloading in vehicular edge multi-access networks,” IEEE Communications Magazine, vol. 56, no. 8, pp. 48–54, 2018.

[2] K. Zhang, S. Leng, X. Peng, L. Pan, S. Maharjan, and Y. Zhang, “Artificial intelligence inspired transmission scheduling in cognitive vehicular communications and networks,” IEEE Internet of Things Journal, vol. 6, no. 2, pp. 1987–1997, 2019.

[3] M. Abadi, P. Barham, J. Chen, Z. Chen, A. Davis, J. Dean, M. Devin, S. Ghemawat, G. Irving, M. Isard, M. Kudlur, J. Levenberg, R. Monga, S. Moore, D. G. Murray, B. Steiner, P. Tucker, V. Vasudevan, P. Warden, M. Wicke, Y. Yu, and X. Zheng, “Tensorflow: A system for large-scale machine learning.” in 12th USENIX Symposium on Operating Systems Design and Implementation (OSDI), vol. 16. USENIX Association, 2016, pp. 265–283.

[4] T. Chen, M. Li, Y. Li, M. Lin, N. Wang, M. Wang, T. Xiao, B. Xu, C. Zhang, and Z. Zhang, “Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems,” CoRR, vol. abs/1512.01274, 2015.

[5] L. Zhou, S. Pan, J. Wang, and A. V. Vasilakos, “Machine learning on big data: Opportunities and challenges,” Neurocomputing, vol. 237, pp. 350– 361, 2017.

[6] S. Yu, M. Liu, W. Dou, X. Liu, and S. Zhou, “Networking for big data: A survey,” IEEE Communications Surveys & Tutorials, vol. 19, no. 1, pp. 531–549, 2017.

[7] M. Li, D. G. Andersen, J. W. Park, A. J. Smola, A. Ahmed, V. Josifovski, J. Long, E. J. Shekita, and B.-Y. Su, “Scaling distributed machine learning with the parameter server.” in 11th USENIX Symposium on Operating Systems Design and Implementation (OSDI), vol. 14. USENIX Association, 2014, pp. 583–598.

[8] B. Fan, S. Leng, and K. Yang, “A dynamic bandwidth allocation algorithm in mobile networks with big data of users and networks,” IEEE Network, vol. 30, no. 1, pp. 6–10, 2016.

[9] Y. Zhang, R. Yu, S. Xie, W. Yao, Y. Xiao, and M. Guizani, “Home m2m networks: Architectures, standards, and qos improvement,” IEEE Communications Magazine, vol. 49, no. 4, pp. 44–52, 2011.

**TECHNOLOGY USED**

**Python Introduction**

**Python** is a general purpose, dynamic, high level and interpreted programming language. It supports Object Oriented programming approach to develop applications. It is simple and easy to learn and provides lots of high-level data structures.

Python is easy to learn yet powerful and versatile scripting language which makes it attractive for Application Development.

Python's syntax and dynamic typing with its interpreted nature, makes it an ideal language for scripting and rapid application development.

Python supports multiple programming pattern, including object oriented, imperative and functional or procedural programming styles.

Python is not intended to work on special area such as web programming. That is why it is known as multipurpose because it can be used with web, enterprise, 3D CAD etc.

We don't need to use data types to declare variable because it is dynamically typed so we can write a=10 to assign an integer value in an integer variable.

Python makes the development and debugging fast because there is no compilation step included in python development and edit-test-debug cycle is very fast.

# Python History

* Python laid its foundation in the late 1980s.
* The implementation of Python was started in the December 1989 by **Guido Van Rossum** at CWI in Netherland.
* In February 1991, van Rossum published the code (labeled version 0.9.0) to alt.sources.
* In 1994, Python 1.0 was released with new features like: lambda, map, filter, and reduce.
* Python 2.0 added new features like: list comprehensions, garbage collection system.
* On December 3, 2008, Python 3.0 (also called "Py3K") was released. It was designed to rectify fundamental flaw of the language.
* ABC programming language is said to be the predecessor of Python language which was capable of Exception Handling and interfacing with Amoeba Operating System.
* Python is influenced by following programming languages:
  + ABC language.
  + Modula-3

# Python Features

Python provides lots of features that are listed below.

#### 1) Easy to Learn and Use

Python is easy to learn and use. It is developer-friendly and high level programming language.

#### 2) Expressive Language

Python language is more expressive means that it is more understandable and readable.

#### 3) Interpreted Language

Python is an interpreted language i.e. interpreter executes the code line by line at a time. This makes debugging easy and thus suitable for beginners.

#### 4) Cross-platform Language

Python can run equally on different platforms such as Windows, Linux, Unix and Macintosh etc. So, we can say that Python is a portable language.

#### 5) Free and Open Source

Python language is freely available at [offical web address](https://www.python.org/" \t "blank). The source-code is also available. Therefore it is open source.

#### 6) Object-Oriented Language

Python supports object oriented language and concepts of classes and objects come into existence.

#### 7) Extensible

It implies that other languages such as C/C++ can be used to compile the code and thus it can be used further in our python code.

#### 8) Large Standard Library

Python has a large and broad library and prvides rich set of module and functions for rapid application development.

#### 9) GUI Programming Support

Graphical user interfaces can be developed using Python.

#### 10) Integrated

It can be easily integrated with languages like C, C++, JAVA etc.

# Python Applications

Python is known for its general purpose nature that makes it applicable in almost each domain of software development. Python as a whole can be used in any sphere of development.

Here, we are specifing applications areas where python can be applied.

#### 1) Web Applications

We can use Python to develop web applications. It provides libraries to handle internet protocols such as HTML and XML, JSON, Email processing, request, beautifulSoup, Feedparser etc. It also provides Frameworks such as Django, Pyramid, Flask etc to design and delelop web based applications. Some important developments are: PythonWikiEngines, Pocoo, PythonBlogSoftware etc.

#### 2) Desktop GUI Applications

Python provides Tk GUI library to develop user interface in python based application. Some other useful toolkits wxWidgets, Kivy, pyqt that are useable on several platforms. The Kivy is popular for writing multitouch applications.

#### 3) Software Development

Python is helpful for software development process. It works as a support language and can be used for build control and management, testing etc.

#### 4) Scientific and Numeric

Python is popular and widely used in scientific and numeric computing. Some useful library and package are SciPy, Pandas, IPython etc. SciPy is group of packages of engineering, science and mathematics.

#### 5) Business Applications

Python is used to build Bussiness applications like ERP and e-commerce systems. Tryton is a high level application platform.

#### 6) Console Based Application

We can use Python to develop console based applications. For example: **IPython**.

#### 7) Audio or Video based Applications

Python is awesome to perform multiple tasks and can be used to develop multimedia applications. Some of real applications are: TimPlayer, cplay etc.

#### 8) 3D CAD Applications

To create CAD application Fandango is a real application which provides full features of CAD.

#### 9) Enterprise Applications

Python can be used to create applications which can be used within an Enterprise or an Organization. Some real time applications are: OpenErp, Tryton, Picalo etc.

#### 10) Applications for Images

Using Python several application can be developed for image. Applications developed are: VPython, Gogh, imgSeek etc.

There are several such applications which can be developed using Python

# How to Install Python (Environment Set-up)

In this section of the tutorial, we will discuss the installation of python on various operating systems.

### Why Python

* Python works on different platforms (Windows, Mac, Linux, Raspberry Pi, etc).
* Python has a simple syntax similar to the English language.
* Python has syntax that allows developers to write programs with fewer lines than some other programming languages.
* Python runs on an interpreter system, meaning that code can be executed as soon as it is written. This means that prototyping can be very quick.
* Python can be treated in a procedural way, an object-orientated way or a functional way.

### Good to know

* The most recent major version of Python is Python 3, which we shall be using in this tutorial. However, Python 2, although not being updated with anything other than security updates, is still quite popular.
* In this tutorial Python will be written in a text editor. It is possible to write Python in an Integrated Development Environment, such as Thonny, Pycharm, Netbeans or Eclipse which are particularly useful when managing larger collections of Python files.

### 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.

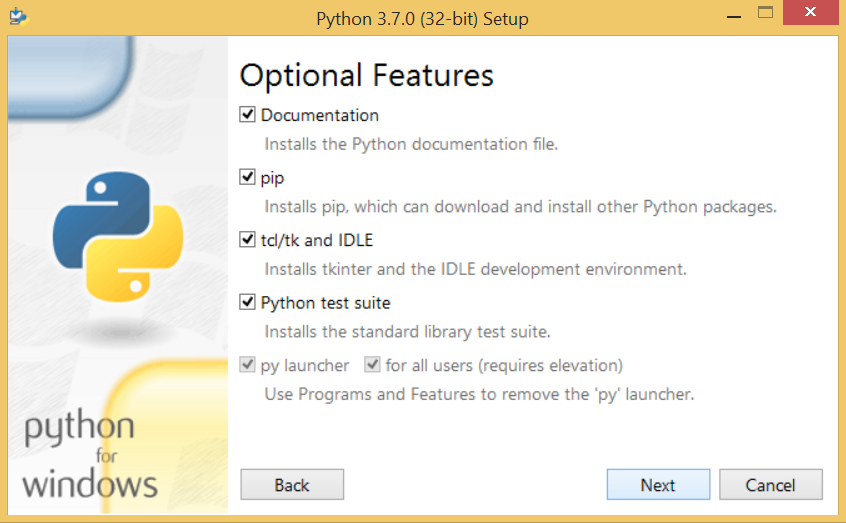
## Installation on Windows

Visit the link <https://www.python.org/downloads/> to download the latest release of Python. In this process, we will install Python 3.6.7 on our Windows operating system.

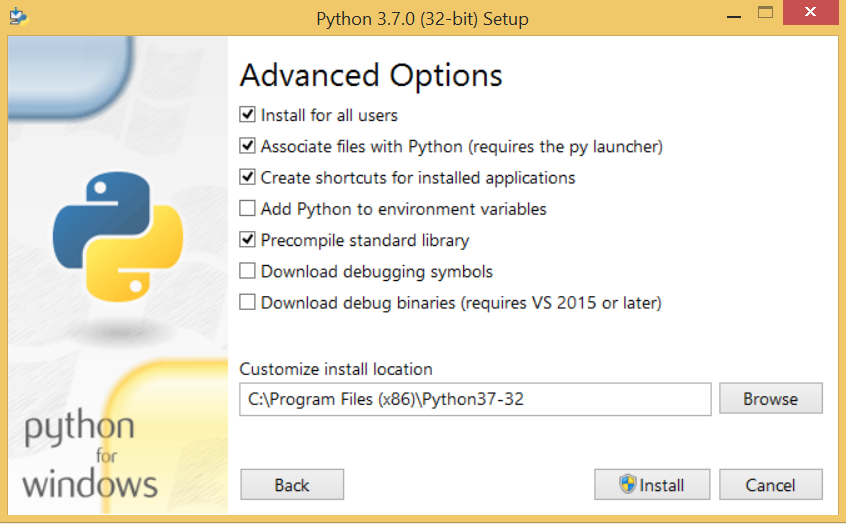
Double-click the executable file which is downloaded; the following window will open. Select Customize installation and proceed.

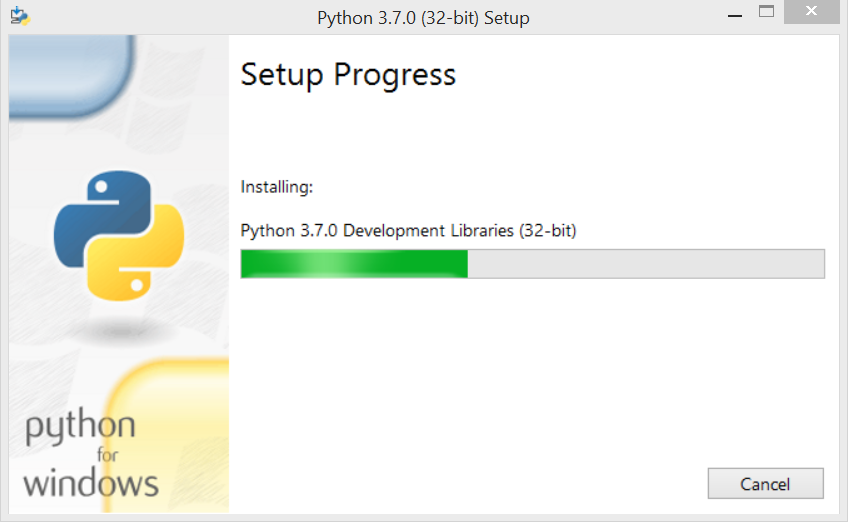
The following window shows all the optional features. All the features need to be installed and are checked by default; we need to click next to continue.

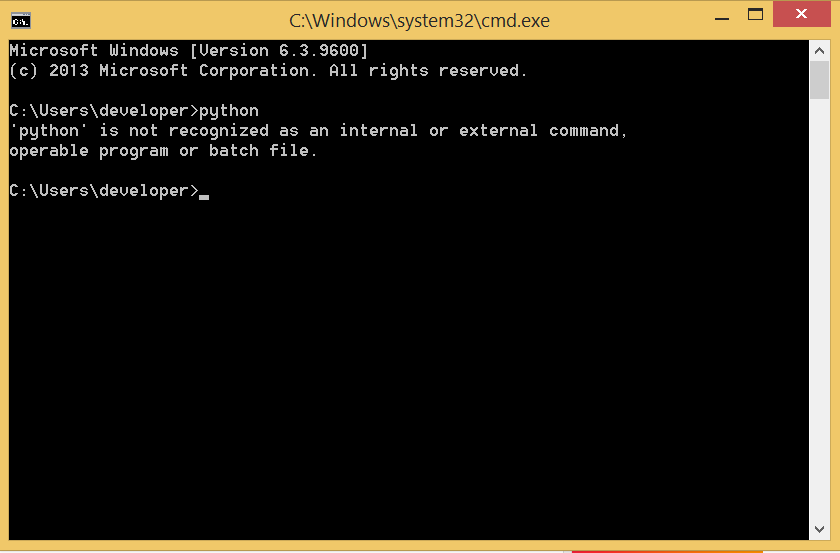
The following window shows a list of advanced options. Check all the options which you want to install and click next. Here, we must notice that the first check-box (install for all users) must be checked.



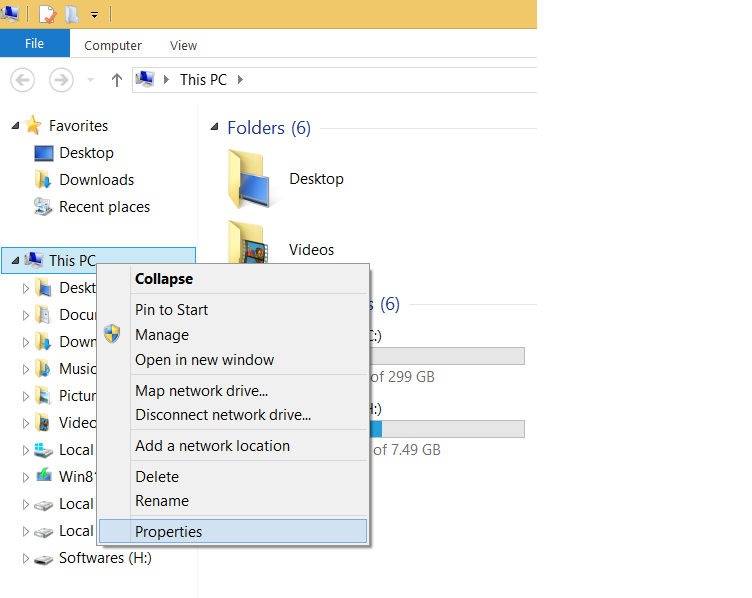
Now, we are ready to install python-3.6.6. Lets install it.

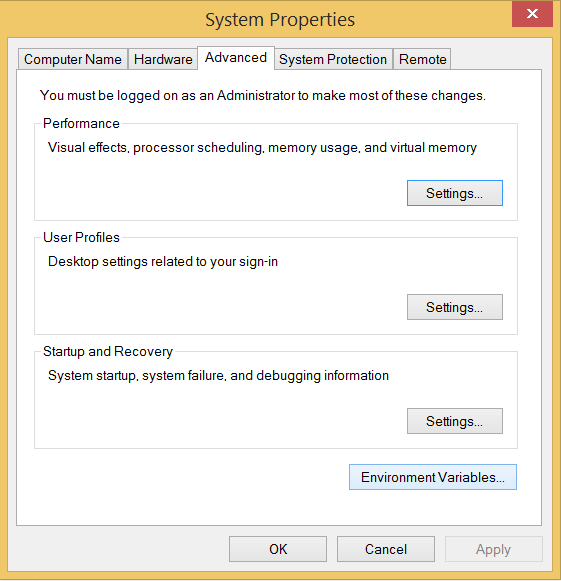


Now, try to run python on the command prompt. Type the command python in case of python2 or python3 in case of python3. It will show an error as given in the below image. It is because we haven't set the path.

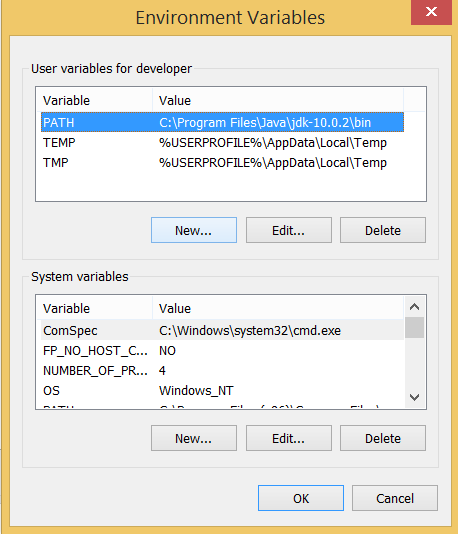


To set the path of python, we need to the right click on "my computer" and go to Properties → Advanced → Environment Variables.

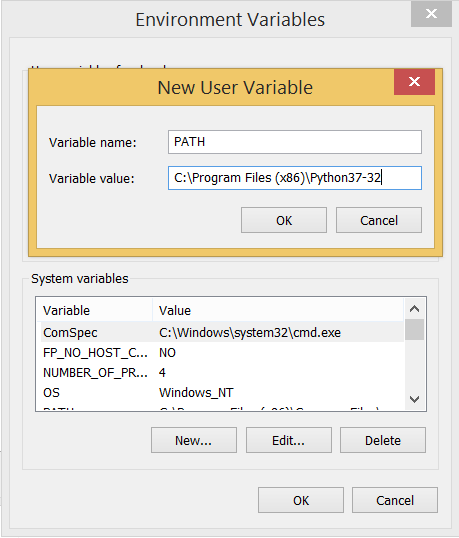




Add the new path variable in the user variable section.



Type PATH as the variable name and set the path to the installation directory of the python shown in the below image.



Now, the path is set, we are ready to run python on our local system. Restart CMD, and type python again. It will open the python interpreter shell where we can execute the python statements.

**Virtual Environments and Packages**

**Introduction**

Python applications will often use packages and modules that don’t come as part of the standard library. Applications will sometimes need a specific version of a library, because the application may require that a particular bug has been fixed or the application may be written using an obsolete version of the library’s interface.

This means it may not be possible for one Python installation to meet the requirements of every application. If application A needs version 1.0 of a particular module but application B needs version 2.0, then the requirements are in conflict and installing either version 1.0 or 2.0 will leave one application unable to run.

The solution for this problem is to create a virtual environment, a self-contained directory tree that contains a Python installation for a particular version of Python, plus a number of additional packages.

Different applications can then use different virtual environments. To resolve the earlier example of conflicting requirements, application A can have its own virtual environment with version 1.0 installed while application B has another virtual environment with version 2.0. If application B requires a library be upgraded to version 3.0, this will not affect application A’s environment.

**Creating Virtual Environments**

The module used to create and manage virtual environments is called venv. venv will usually install the most recent version of Python that you have available. If you have multiple versions of Python on your system, you can select a specific Python version by running python3 or whichever version you want.

To create a virtual environment, decide upon a directory where you want to place it, and run the venv module as a script with the directory path:

python3 -m venv tutorial-env

This will create the tutorial-env directory if it doesn’t exist, and also create directories inside it containing a copy of the Python interpreter, the standard library, and various supporting files.

A common directory location for a virtual environment is .venv. This name keeps the directory typically hidden in your shell and thus out of the way while giving it a name that explains why the directory exists. It also prevents clashing with .env environment variable definition files that some tooling supports.

Once you’ve created a virtual environment, you may activate it.

On Windows, run:

tutorial-env\Scripts\activate.bat

On Unix or MacOS, run:

source tutorial-env/bin/activate

(This script is written for the bash shell. If you use the csh or fish shells, there are alternate activate.csh and activate.fish scripts you should use instead.)

Activating the virtual environment will change your shell’s prompt to show what virtual environment you’re using, and modify the environment so that running python will get you that particular version and installation of Python. For example:

$ source ~/envs/tutorial-env/bin/activate

(tutorial-env) $ python

Python 3.5.1 (default, May 6 2016, 10:59:36)

...

>>> import sys

>>>sys.path

['', '/usr/local/lib/python35.zip', ...,

'~/envs/tutorial-env/lib/python3.5/site-packages']

>>>

**Managing Packages with pip**

You can install, upgrade, and remove packages using a program called pip. By default pip will install packages from the Python Package Index, <https://pypi.org>. You can browse the Python Package Index by going to it in your web browser, or you can use pip’s limited search feature:

(tutorial-env) $ pip search astronomy

skyfield - Elegant astronomy for Python

gary - Galactic astronomy and gravitational dynamics.

novas - The United States Naval Observatory NOVAS astronomy library

astroobs - Provides astronomy ephemeris to plan telescope observations

PyAstronomy - A collection of astronomy related tools for Python.

...

pip has a number of subcommands: “search”, “install”, “uninstall”, “freeze”, etc. (Consult the Installing Python Modules guide for complete documentation for pip.)

You can install the latest version of a package by specifying a package’s name:

(tutorial-env) $ pip install novas

Collecting novas

Downloading novas-3.1.1.3.tar.gz (136kB)

Installing collected packages: novas

Running setup.py install for novas

Successfully installed novas-3.1.1.3

You can also install a specific version of a package by giving the package name followed by == and the version number:

(tutorial-env) $ pip install requests==2.6.0

Collecting requests==2.6.0

Using cached requests-2.6.0-py2.py3-none-any.whl

Installing collected packages: requests

Successfully installed requests-2.6.0

If you re-run this command, pip will notice that the requested version is already installed and do nothing. You can supply a different version number to get that version, or you can run pip install --upgrade to upgrade the package to the latest version:

(tutorial-env) $ pip install --upgrade requests

Collecting requests

Installing collected packages: requests

Found existing installation: requests 2.6.0

Uninstalling requests-2.6.0:

Successfully uninstalled requests-2.6.0

Successfully installed requests-2.7.0

pip uninstall followed by one or more package names will remove the packages from the virtual environment.

pip show will display information about a particular package:

(tutorial-env) $ pip show requests

---

Metadata-Version: 2.0

Name: requests

Version: 2.7.0

Summary: Python HTTP for Humans.

Home-page: http://python-requests.org

Author: Kenneth Reitz

Author-email: me@kennethreitz.com

License: Apache 2.0

Location: /Users/akuchling/envs/tutorial-env/lib/python3.4/site-packages

Requires:

pip list will display all of the packages installed in the virtual environment:

(tutorial-env) $ pip list

novas (3.1.1.3)

numpy (1.9.2)

pip (7.0.3)

requests (2.7.0)

setuptools (16.0)

pip freeze will produce a similar list of the installed packages, but the output uses the format that pip install expects. A common convention is to put this list in a requirements.txt file:

(tutorial-env) $ pip freeze > requirements.txt

(tutorial-env) $ cat requirements.txt

novas==3.1.1.3

numpy==1.9.2

requests==2.7.0

The requirements.txt can then be committed to version control and shipped as part of an application. Users can then install all the necessary packages with install -r:

(tutorial-env) $ pip install -r requirements.txt

Collecting novas==3.1.1.3 (from -r requirements.txt (line 1))

...

Collecting numpy==1.9.2 (from -r requirements.txt (line 2))

...

Collecting requests==2.7.0 (from -r requirements.txt (line 3))

...

Installing collected packages: novas, numpy, requests

Running setup.py install for novas

Successfully installed novas-3.1.1.3 numpy-1.9.2 requests-2.7.0

pip has many more options. Consult the Installing Python Modules guide for complete documentation for pip. When you’ve written a package and want to make it available on the Python Package Index, consult the Distributing Python Modules guide.