**Experiment No:1**

**12.08.2022**

**Overview of Google Colab**

Colaboratory, or “Colab” for short, is a product from Google Research. Colab allows anybody to write and execute arbitrary python code through the browser, and is especially well suited to machine learning, data analysis and education. Colab is basically a free Jupyter notebook environment running wholly in the cloud. Colab does not require a setup, plus the notebooks that are created can be simultaneously edited by team members. To use Colaboratory, you must have a Google account and then access Colaboratory using your account.

**Advantages**

**1) Sharing:**

Google Colab notebooks can be shared very easily.

## 2) Versioning:

Notebook can be saved to Github with just one simple click on a button. There is no need to write “git add git commit git push git pull” codes.

## 3) Code snippets:

Google Colab has a great collection of snippets that can be plug in on your code.

## 4) Forms for non-technical users:

Not only programmers have to analyze data and Python can be useful for almost everyone in an office job. Non -technical users can change form fields and Google Colab will automatically update the code.

## 5) Performance:

Running python scripts require often a lot of computing power and can take time. By running scripts in the cloud the local machine performance won’t drop while executing Python scripts.

## 6) Price:

## It is free.

**Disadvantages**

**1)Closed-Environment:**

Machine learning practitioners can only run the python package already pre-added on the Colab. There is no way that one can add their own python package and start running the code. Hence, the platform can provide common tools but is not suitable for specialisation.

**2)Repetitive Tasks:**

For every new session in the Google Colab, a programmer must install all of the specific libraries that aren’t included with the standard Python package.

**3)No Live-Editing:**

The option for live editing is completely missing in Google Colab, which restricts two people to write, or edit codes at the same time. Hence, it further leads to a lot of back and forth re-sharing.

**4)Saving & Storage Problems:**

Uploaded files are removed when the session is restarted because Google Colab does not provide a persistent storage facility. So, if the device is turned off, the data can get lost.

**5)Limited Space & Time:** The Google Colab platform stores files in Google Drive with a free space of 15GB; however, working on bigger datasets requires more space, making it difficult to execute.

**The differences between jupyter and google colab are as follows:**

* Jupyter runs on your local machine and your files are saved on your hard disk, while colab runs on a Google server, and your files are stored in your google drive account.
* Jupyter uses processor and no access to external GPU AND TPU, while Colab runs on their server and provide free GPU and TPU.
* Jupyter runtime depends on your system memory limit. In google colab there is a runtime limit of 12/24 hrs and can be interrupted by google.
* In jupyter, all required libraries have to be installed depending on your need, but in colab, most of the libraries are preinstalled in it.
* In Jupyter, notebook files cannot be accessed without hard-drive while in Google Collab they can be accessed from anywhere since they are stored in Google Drive.

## Fundamentals of Python

#### Data Types :

A data type is a set of values and a set of operations defined on data. They can be numbers, strings, lists, dictionaries, tuples, sets etc.

#### Functions:

Python has a lot of built-in functions like print, len, input, etc. Besides the built-in functions of Python, you can also define your functions. A function is a block of code defined to perform a certain action. They are primarily used to replace repetitive statements in your code.

#### Lists:

Lists are ordered collections of items and the most general data type provided by the Python programming language. They are mutable, which means that items stored in a list can be edited. A list is typically used to perform operations on a collection of items at a time.

#### Tuples:

A tuple is also a collection of items very similar to a list in Python. Tuples allow to store an ordered collection of items to perform operations at one time. But unlike lists, a tuple cannot be modified once created. The only advantage of using tuples over lists is that tuples are slightly faster than lists.

#### Dictionary:

A dictionary in Python is completely different from lists and tuples, they are not sequences but mappings. Mappings are also collections of items but in the form of key and value pairs. Simply put, a dictionary contains indexes with keys mapped to certain values

#### Loops:

Loops are statements used to iterate over an object. There are two types of loops in Python:

1.While Loops:

loop is used to iterate through a collection of items.

2.For Loops:

The While Loop allows you to execute a set of instructions until the given condition is true.

#### Conditions:

**If, else and elif statements:**

They are conditional statements found in all programming languages. These statements are used to write an event-driven program and are used to execute a set of statements only if the given condition is met.

### Boolean operators (or, and, not):

These are used to evaluate complex assertions together.

* or -One of the many comparisons should be true for the entire condition to be true.
* and -All of the comparisons should be true for the entire condition to be true.
* not -Checks for the opposite of the comparison specified.

## Object-Oriented programming in Python

## In Python, object-oriented Programming (OOPs) is a programming paradigm that uses objects and classes in programming. It aims to implement real-world entities like inheritance, polymorphisms, encapsulation, etc. in the programming.The main concept of OOPs is to bind the data and the functions that work on that together as a single unit so that no other part of the code can access this data.

A class is a collection of objects. A class contains the blueprints or the prototype from which the objects are being created. It is a logical entity that contains some attributes and methods.

The object is an entity that has a state and behavior associated with it.Objects are instance of a class and are defined as an encapsulation of variables (data) and functions into a single entity. They have access to the variables (attributes) and methods (functions) from classes. The attributes and methods can be accessed using the dot(.) operator.

### External Libraries/Modules

One of the main reasons to use Python for data science is that it can develops high-quality packages for different domains and problems. Using external libraries and modules is an integral part of working on projects in Python.

These libraries and modules have defined classes, attributes, and methods that we can use to accomplish our tasks. For example, the math library contains many mathematical functions that we can use to carry out our calculations. The libraries are .py files.

**Program No:1**

**06.10.2022**

**Aim**

Predict class label of a given data point using KNN.

**Source Code**

from sklearn.neighbors import KNeighborsClassifier

x1=[7,7,3,1]

x2=[7,4,4,4]

target=['bad','bad','Good','Good']

from sklearn import preprocessing

le=preprocessing.LabelEncoder()

target\_encoded=le.fit\_transform(target)

print(target\_encoded)

features=zip(x1,x2)

features=list(features)

features

knn = KNeighborsClassifier(n\_neighbors=3)

knn.fit(features,target)

print(knn.predict([[3,7]]))

**Output**

[1 1 0 0]

['Good']

**Program No:2**

**06.10.2022**

**Aim**

Predict the class label of an unseen observation using Naive-bayes.

**Source Code**

Weather=['Sunny','Sunny','Overcast','Rainy','Rainy','Rainy','Overcast','sunny','Sunny','Rainy','Sunny','Overcast','Overcast','Rainy']

temp=['hot','hot','hot','mild','cool','cool','cool','mild','cool','mild','mild','mild','hot','mild']

play=['no','no','yes','yes','yes','no','yes','no','yes','yes','yes','yes','yes','no']

from sklearn import preprocessing

le =preprocessing.LabelEncoder()

weather\_encoded=le.fit\_transform(weather)

print(weather\_encoded)

temp\_encoded=le.fit\_transform(temp)

label=le.fit\_transform(play)

print("Temp:",temp\_encoded)

print("Play:",label)

features=zip(weather\_encoded,temp\_encoded)

features=list(features)

print ("features",features)

from sklearn.naive\_bayes import GaussianNB

model = GaussianNB()

model.fit(features, label)

predicted= model.predict([[0,2]])

print("Predicted Value:", predicted)

**Output**

[2 2 0 1 1 1 0 3 2 1 2 0 0 1]

Temp: [1 1 1 2 0 0 0 2 0 2 2 2 1 2]

Play: [0 0 1 1 1 0 1 0 1 1 1 1 1 0]

features [(2, 1), (2, 1), (0, 1), (1, 2), (1, 0), (1, 0), (0, 0), (3, 2), (2, 0), (1, 2), (2, 2), (0, 2), (0, 1), (1, 2)]

Predicted Value: [1]