

# Machine Learning and Deep Learning using Software-as-a-Service (via Google<sup>TM</sup> Colab)

July 9, 2021

Bonaventure Chidube Molokwu  
molokwub [at] uwindsor . ca

School of Computer Science  
University of Windsor  
Ontario - Canada



University  
of Windsor



# Introduction

## ML & DL using SaaS via Google Colab

1

### Introduction

Terminologies

IaaS

PaaS

SaaS

### Colab<sup>TM</sup>

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

### Q & A



# Selected Terminologies

## 1. Concurrent Systems

### **1. Concurrent Systems:** these are systems

comprising a collection of independent components

(or hardware) which may perform operations at the

same time (or simultaneously) with the goal of

achieving efficiency.

#### 1.1 Parallel Computing.

#### 1.2 Distributed Computing.

# Selected Terminologies

## 1.1 Parallel Computing

ML & DL  
using SaaS  
via Google Colab

Introduction

3  
Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

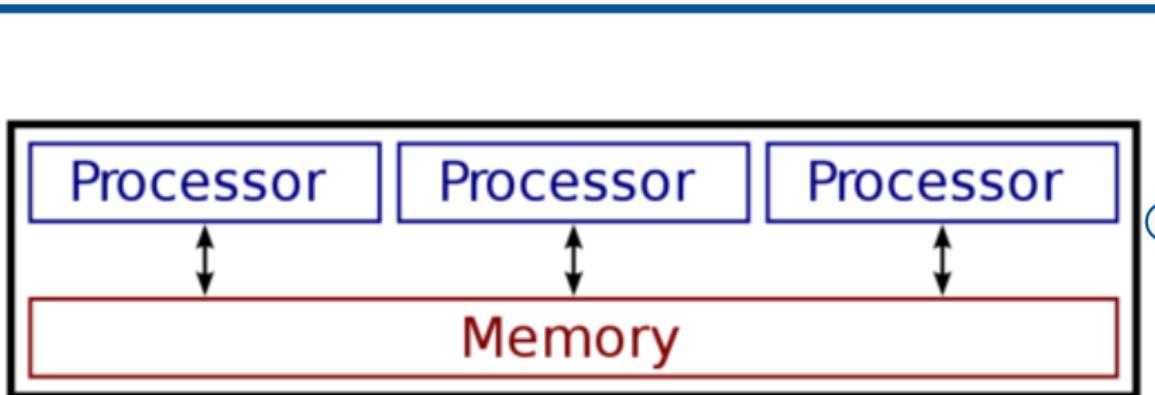


Figure: Parallel computing

**1.1 Parallel Computing:** several processors, which

have access to shared memory, execute or process  
a task.

# Selected Terminologies

## 1.2 Distributed Computing

ML & DL  
using SaaS  
via Google Colab

Introduction

4  
Terminologies  
IaaS  
PaaS  
SaaS

Colab™

Basics  
Code Comments & Documentation  
Save/Store Code  
Share/Teamwork  
Files/Directories  
SQL Databases  
CPU vs. GPU  
Data Forms  
Interactive Widgets  
Linux Commands

Q & A

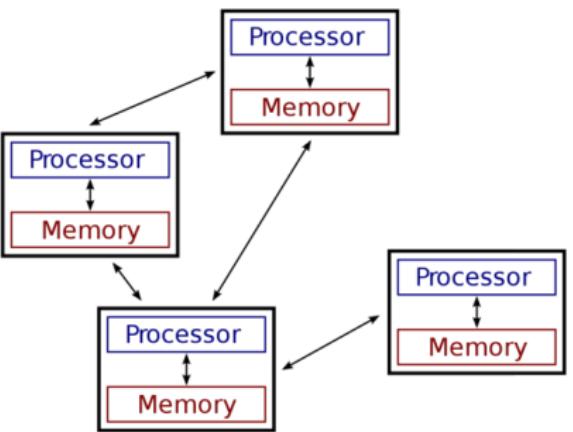


Figure: Distributed computing

**1.2 Distributed Computing:** each processor has its own private memory for executing or processing a task. Information is exchanged via message passing between processors.

# Selected Terminologies

## 2. Cluster Computing

ML & DL  
using SaaS  
via Google  
Colab

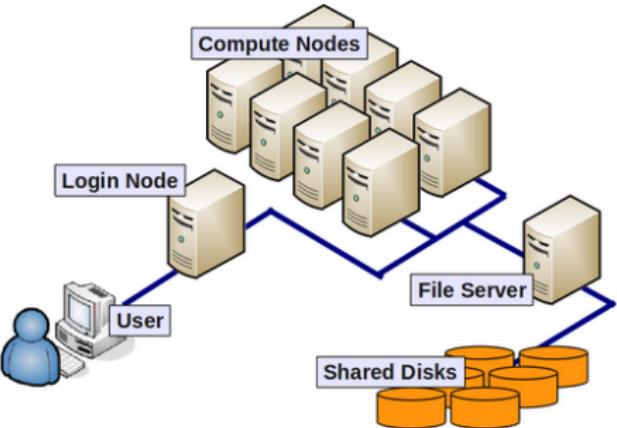
Introduction

5  
Terminologies  
IaaS  
PaaS  
SaaS

Colab™

Basics  
Code Comments & Documentation  
Save/Store Code  
Share/Teamwork  
Files/Directories  
SQL Databases  
CPU vs. GPU  
Data Forms  
Interactive Widgets  
Linux Commands

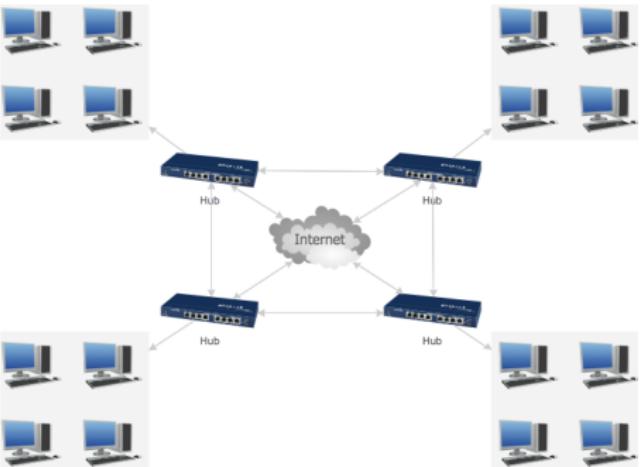
Q & A



**2. Cluster Computing:** A type of **Parallel Computing** comprising several networked computers (nodes) using a shared storage, and all within a specific location (single system). Each node is set to perform the same task controlled by a scheduler.

# Selected Terminologies

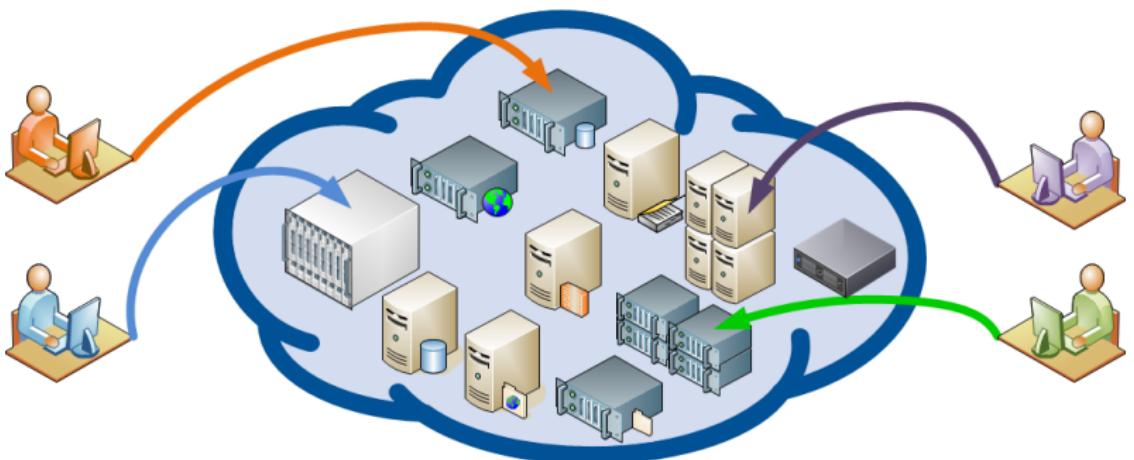
## 3. Grid Computing



**3. Grid Computing:** Type of **Distributed Computing** comprising several networked computers (nodes) having independent storage, and spread across several geographical locations. Each node is set to perform different task.

# Selected Terminologies

## 4. Virtual Computing



ML & DL  
using SaaS  
via Google  
Colab

Introduction

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

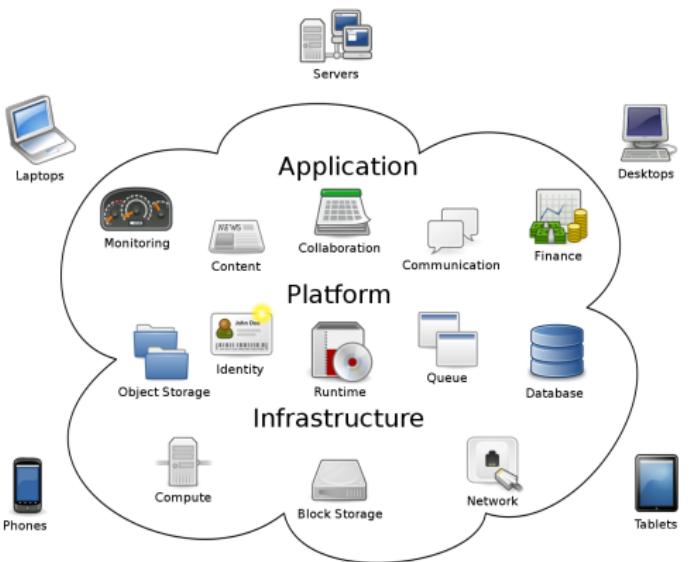
**4. Virtual Computing:** Use of a remote computer,  
remote software, or remote resources from a local  
computer/machine over a computer network.

# Selected Terminologies

## 5. Cloud Computing

ML & DL  
using SaaS  
via Google  
Colab

8



Introduction  
Terminologies  
IaaS  
PaaS  
SaaS

Colab™

Basics  
Code Comments & Documentation  
Save/Store Code  
Share/Teamwork  
Files/Directories  
SQL Databases  
CPU vs. GPU  
Data Forms  
Interactive Widgets  
Linux Commands

Q & A

5. **Cloud Computing:** Provision and transmission of computing services (such as servers, networking, storage, databases, software, analytics, intelligence, etc.) over the Internet (**the cloud**).



# Selected Terminologies

## Virtual Computing vs. Cloud Computing



University  
of Windsor

ML & DL  
using SaaS  
via Google  
Colab

Introduction

9

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

Table: Virtual Computing vs. Cloud Computing

S/N	Virtual Computing	Cloud Computing
1.	Access to remote system(s) or simulated environments.	Provision of pools and automated resources accessed on-demand.
2.	Relatively cheaper, easier, and less complicated setup.	Relatively expensive, tedious, and complicated setup.
3.	Relatively lower scalability and flexibility.	Highly scalable and flexible.

# Selected Terminologies

## Virtual Computing vs. Cloud Computing



University  
of Windsor

ML & DL  
using SaaS  
via Google  
Colab

Introduction

10

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive

Widgets

Linux Commands

Q & A

Table: Virtual Computing vs. Cloud Computing (contd.)

S/N	Virtual Computing	Cloud Computing
4.	Fewer concurrent users.	Many concurrent users.
5.	Access usually based on authentication.	Access based on authentication and paid subscription(s).
6.	Provision of limited storage space.	Provision of unlimited storage space.

# Selected Terminologies

## Service Models in Cloud Computing

ML & DL  
using SaaS  
via Google  
Colab

Introduction

11

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

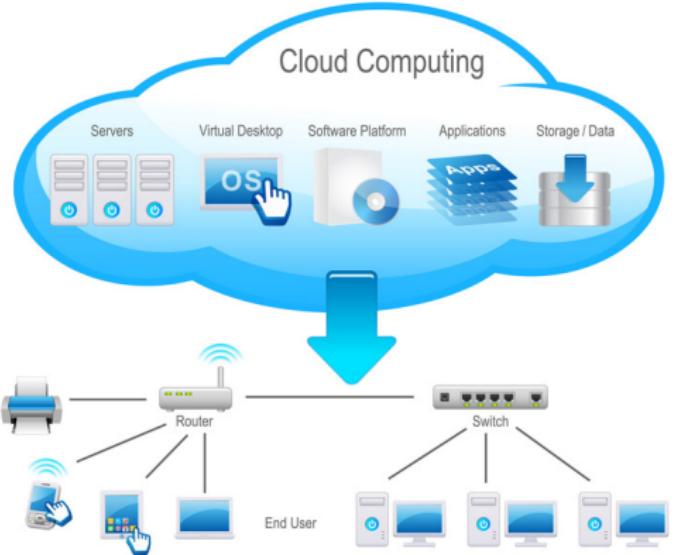
CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A



5.1 **IaaS:** Infrastructure as a Service (IaaS).

5.2 **PaaS:** Platform as a Service (PaaS).

5.3 **SaaS:** Software as a Service (SaaS).

# Selected Terminologies

## IaaS: Infrastructure as a Service

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

### IaaS



Servers and storage



Networking firewalls / security



Data center physical plant /  
building

5.1 **IaaS:** Most basic category of Cloud Computing services. Provides computing hardware (such as servers, storage, networks, virtual machines (VMs), etc.) from a Cloud Service Provider on a **pay-as-you-go** basis.

# Selected Terminologies

## PaaS: Platform as a Service

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

**PaaS**

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

PaaS

IaaS

13



Development tools,  
database management,  
business analytics



Operating systems



Servers and storage



Networking  
firewalls/security



Data centre physical  
facility/building

**5.2 PaaS:** Provides complete development and deployment environment(s) comprising Operating System(s), middleware, and all components of IaaS, on a **pay-as-you-go** basis from a Cloud Service Provider.

# Selected Terminologies

SaaS: Software as a Service

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

14

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

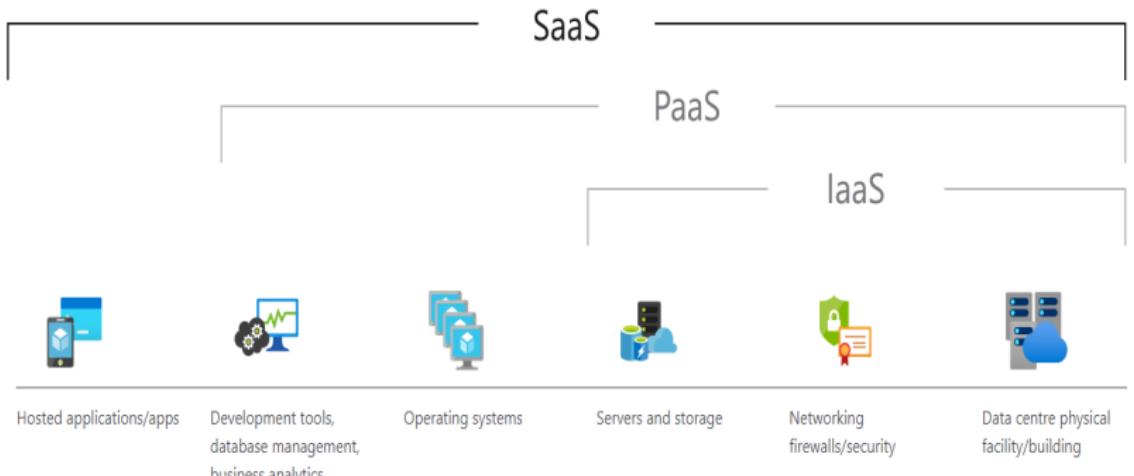
CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A



**5.3 SaaS:** Applications as a Service (AaaS) or On-demand Software. Delivery of software applications (such as email, office tools, etc.), usually on **pay-as-you-go** basis, as a service over the Internet. Includes all of IaaS + PaaS.



Google Colaboratory

1. A SaaS cloud-based tool developed by Google™.
2. It is a free IDE for writing and implementing Python-based codes.
3. Requires relatively no configuration.
4. Provides free access to cloud-based computing resources (CPUs, GPUs, Memory, Storage, etc.).
5. Ideal for team work and collaboration.

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

15 Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

# Google™ Colab

Colab/Colaboratory SaaS: Launching via Web Browser



University  
of Windsor

1. Ensure that you are logged onto your GMail account.
2. Proceed to the following URL:  
<https://colab.research.google.com/>.

The screenshot shows the Google Colab web interface. On the left, there's a sidebar with a 'Table of contents' section containing links like 'Getting started', 'Data science', 'Machine learning', 'More Resources', and 'Machine Learning Examples'. Below this is a 'Section' heading. In the main area, a notebook titled 'What is Colaboratory?' is open. The notebook content includes:

- A heading 'What is Colaboratory?'
- A bulleted list:
  - Zero configuration
  - Free access to GPU
  - Easy sharing
- A section 'Whether you're a student just get started below!'
- A section 'Getting started':
  - The document you are viewing.
  - For example, here is a cell:
  - Code cell content: `[3]: seconds_in_a_day  
seconds_in_a_day  
86400`
  - Text: 'To execute the code in this cell, use the keyboard shortcut "Command+Enter".'
  - Text: 'Variables that you define in this cell are available in all other cells.'
- A code cell at the bottom with the code: `seconds_in_a_week = 7 * seconds_in_a_day  
seconds_in_a_week  
604800`

At the top of the page, the URL is <https://colab.research.google.com/notebooks/intro.ipynb#recent=true>. The top right corner shows a 'Share' button and a user icon. A blue circle highlights the number '16' in the top right corner of the sidebar.

ML & DL  
using SaaS  
via Google  
Colab

Introduction  
Terminologies  
IaaS  
PaaS  
SaaS

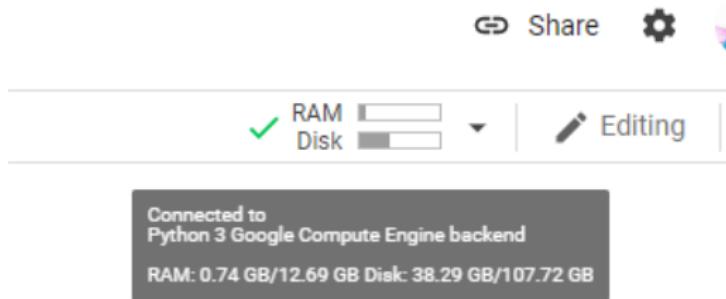
Colab™

Basics

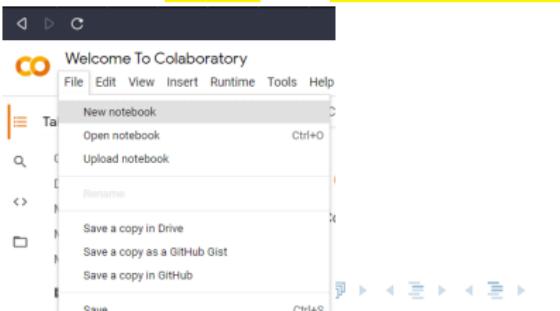
Code Comments & Documentation  
Save/Store Code  
Share/Teamwork  
Files/Directories  
SQL Databases  
CPU vs. GPU  
Data Forms  
Interactive Widgets  
Linux Commands

Q & A

3. View current dynamically allocated resources via *hover* on the right-hand segment of the menu bar.



4. Open a new code editor: **File** → **New notebook**



ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

17

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

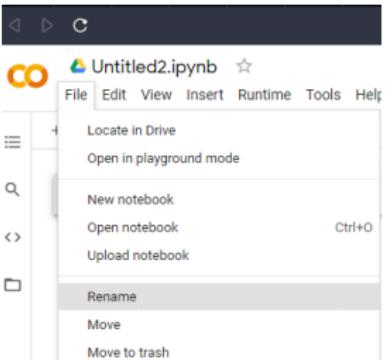
Linux Commands

Q & A

54

### 5. Rename a newly created (source) code editor:

File → Rename



### 6. Alternatively, (source) editor can be renamed via: **Double-click** on the **filename**, and rename.

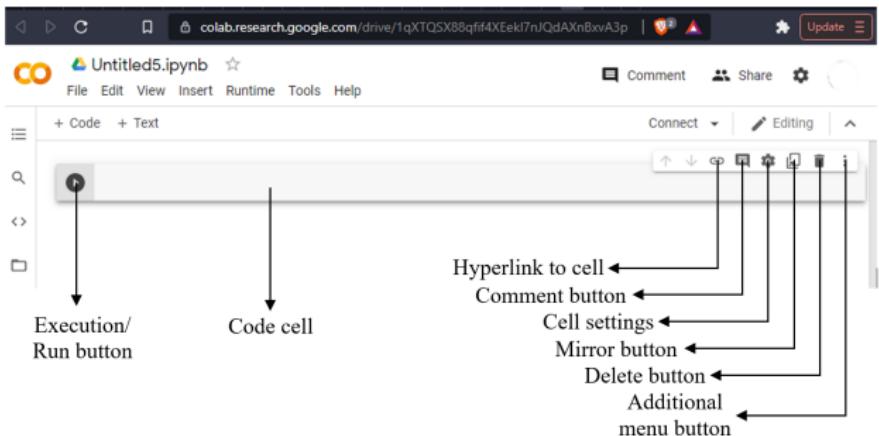


File Edit View Insert Runtime

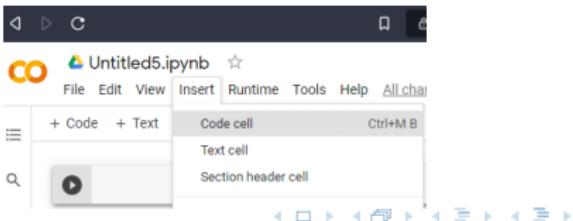
# Google™ Colab

## Colab/Colaboratory SaaS: Basics

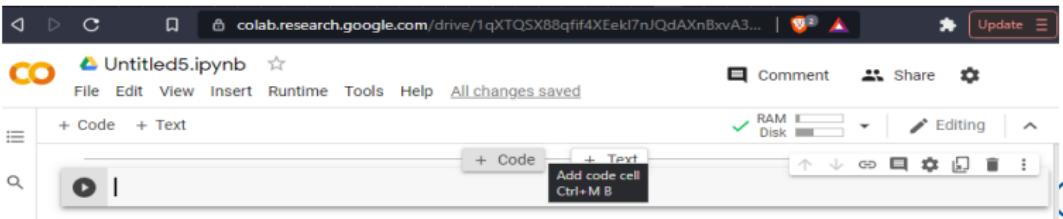
### 7. Code-cell basics:



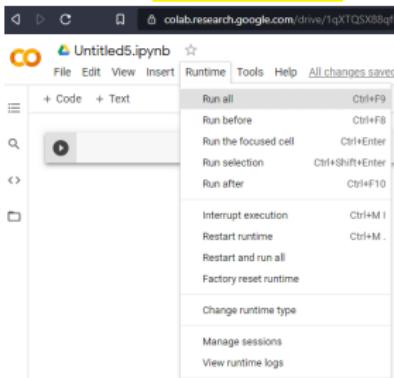
### 8. New code/text cell via: **Insert** → **Code/Text cell**



9. Alternatively, new code/text cell can be invoked via:  
**Mouse-hover** on the **top/bottom edge-line** of cell.

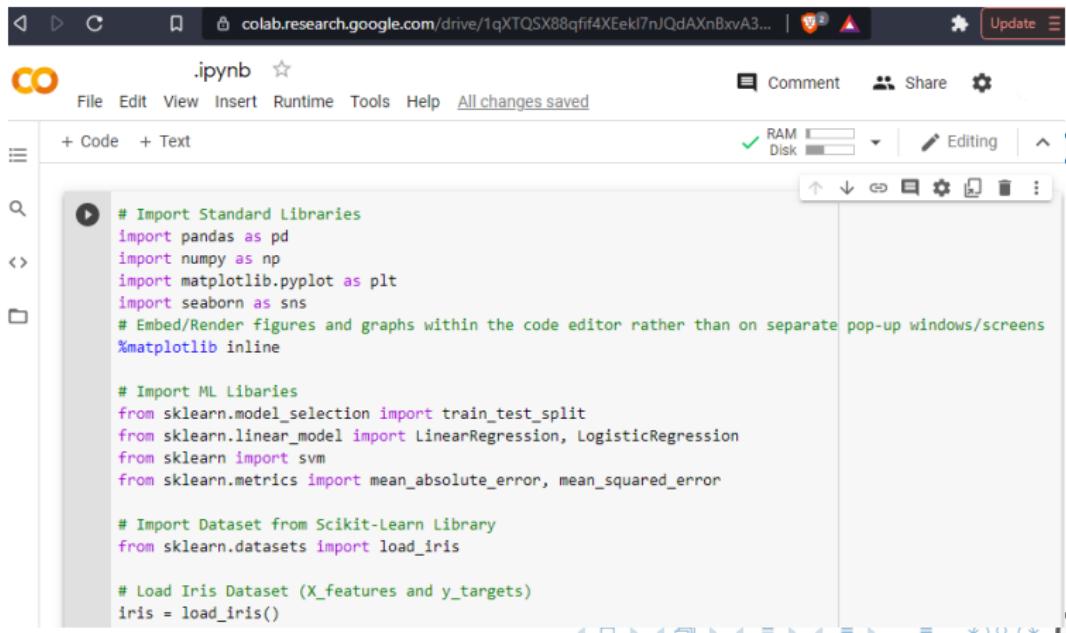


10. New code/text cell via: Runtime → Run all



### 11. Coding and running sample code:

[https://github.com/bhevencious/Workshop\\_ML-and-DL-using-SaaS/blob/main/Workshop\\_ML\\_and\\_DL\\_using\\_SaaS.ipynb](https://github.com/bhevencious/Workshop_ML-and-DL-using-SaaS/blob/main/Workshop_ML_and_DL_using_SaaS.ipynb)



The screenshot shows the Google Colab interface with a Jupyter notebook titled ".ipynb". The code cell contains Python code for a machine learning project, specifically for loading the Iris dataset and performing initial data processing. The interface includes a toolbar with various icons for file operations, a sidebar with navigation and search tools, and a status bar at the bottom.

```
# Import Standard Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Embed/Render figures and graphs within the code editor rather than on separate pop-up windows/screens
%matplotlib inline

# Import ML Libaries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn import svm
from sklearn.metrics import mean_absolute_error, mean_squared_error

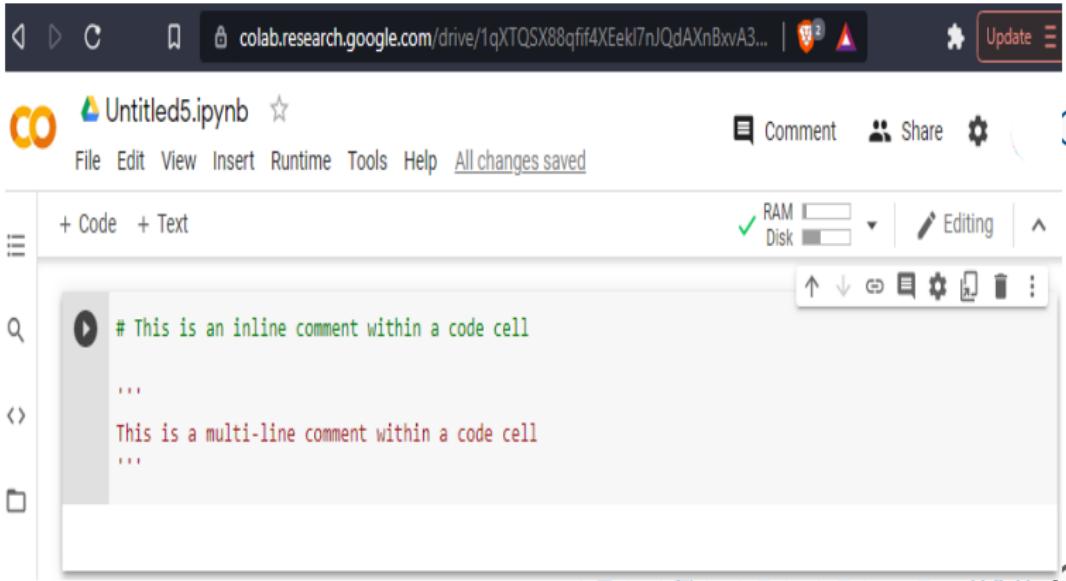
# Import Dataset from Scikit-Learn Library
from sklearn.datasets import load_iris

# Load Iris Dataset (X_features and y_targets)
iris = load_iris()
```

### 12. Code documentation via code cell(s) :

12.1 Inline comment (#).

12.2 Multi-line or Block comment ("").



The screenshot shows a Google Colab notebook titled "Untitled5.ipynb". The top bar includes standard browser controls, a URL bar with "colab.research.google.com", and a toolbar with "Update" and other icons. The main workspace contains two code cells:

```
# This is an inline comment within a code cell
...
This is a multi-line comment within a code cell
...
```

The notebook menu bar includes File, Edit, View, Insert, Runtime, Tools, Help, and a status message "All changes saved". On the right side, there's a toolbar with "Comment", "Share", and "Edit" buttons, along with memory management sliders for RAM and Disk, and a cell control panel with up/down arrows, copy/paste, and delete icons.

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments & Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive Widgets

Linux Commands

Q & A



### 13. Code documentation via **text cell(s)**:

#### 13.1 Advanced code comments - **Markdown** language.

#### 13.2 Basic guide and syntax:

<https://www.markdownguide.org/basic-syntax/>

The screenshot shows the Google Colab interface. At the top, there's a toolbar with icons for back, forward, cell type selection, and a search bar. The URL is colab.research.google.com/drive/1qXTQ5X88qfif4XeekI7nJQdAXnBxvA3... The main menu includes File, Edit, View, Insert, Runtime, Tools, Help, and a status bar indicating "All changes saved". Below the menu is a toolbar with "Code" and "Text" buttons, RAM/Disk status, and an "Editing" mode switch. The left sidebar has sections for "+ Code", "+ Text", "Search", and "Cell". The main area contains two code cells. The first cell contains Python code for linear and logistic regression. The second cell is a text cell containing Markdown and LaTeX text. A vertical dashed line separates the two cells. The text cell content is: "Text cells in Google Colab support **Markdown** and **LaTeX** syntax as well". Below this, a horizontal line separates the text from the equations: "Linear Regression:  $y = mx + c$ " and "Logistic Regression:  $\frac{1}{1+e^{-x}} \equiv \frac{e^x}{e^x+1}$ ". The bottom of the screen shows navigation arrows and other interface elements.

23

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

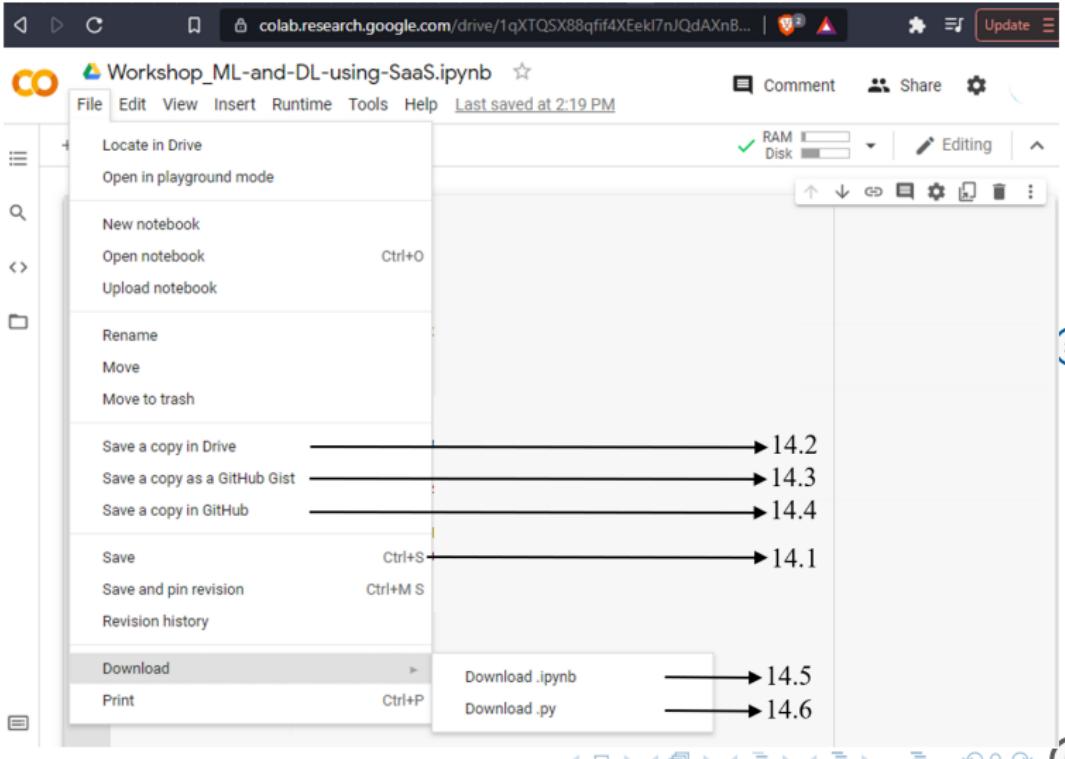
Linux Commands

Q & A

54

14. Save and/or Download source file:
  - 14.1 **File** → **Save** : Saves source file.
  - 14.2 **File** → **Save a copy in Drive** : Saves a copy (duplicate) of the source file to Google™ Drive.
  - 14.3 **File** → **Save a copy as a GitHub Gist** : Saves a copy (duplicate) of the source file to your GitHub™ Gist Timeline/Repository, <https://gist.github.com/>
  - 14.4 **File** → **Save a copy in GitHub** : Saves a copy (duplicate) of source file to a GitHub™ Repository.
  - 14.5 **File** → **Download** → **Download .ipynb** : Download source file (in *.ipynb* format) to your local machine.
  - 14.6 **File** → **Download** → **Download .py** : Download source file (in *.py* format) to your local machine.

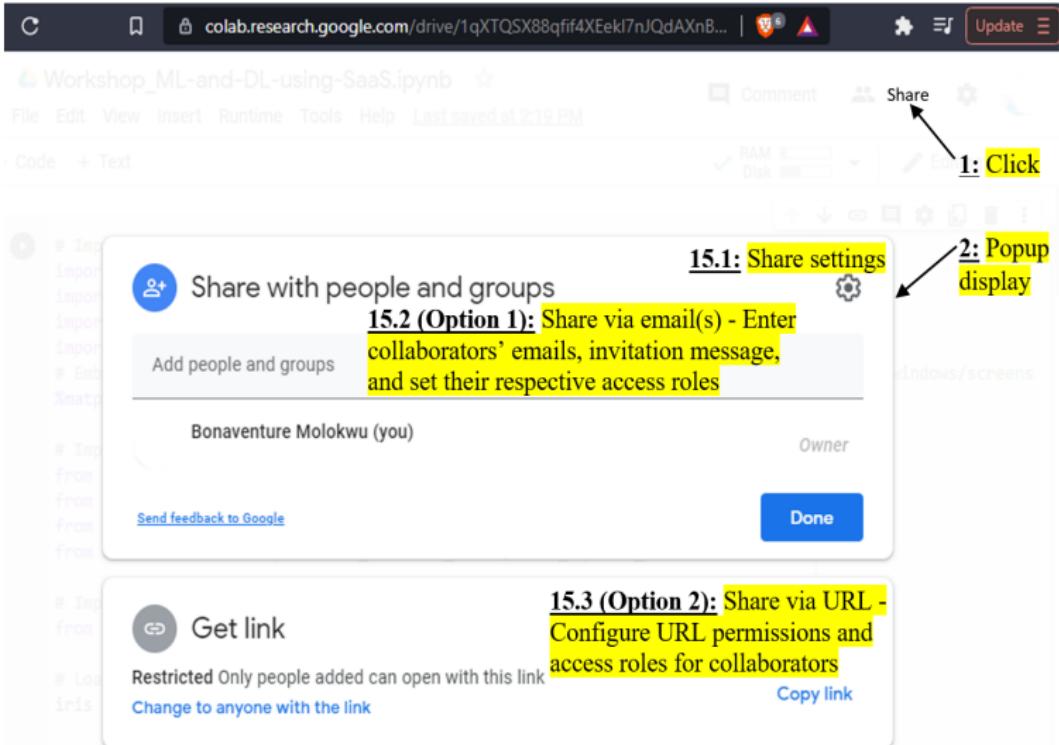
### 14. Save and/or Download source file:



The screenshot shows the Google Colab interface with the 'File' menu open. The menu includes options like 'Locate in Drive', 'Open in playground mode', 'New notebook', 'Open notebook', 'Upload notebook', 'Rename', 'Move', 'Move to trash', 'Save a copy in Drive', 'Save a copy as a GitHub Gist', 'Save a copy in GitHub', 'Save', 'Save and pin revision', 'Revision history', 'Download' (selected), 'Print', and 'Save' (disabled). A context menu also lists 'Download.ipynb' and 'Download.py'. The right side of the screen shows a toolbar with various icons.

- 14.2 Save a copy in Drive
- 14.3 Save a copy as a GitHub Gist
- 14.4 Save a copy in GitHub
- 14.1 Save Ctrl+S
- 14.5 Download.ipynb Download
- 14.6 Download.py Print

### 15. Share/Teamwork on your code:



The screenshot shows the Google Colab interface with a code cell containing Python code. The top navigation bar includes 'File', 'Edit', 'View', 'Insert', 'Runtime', 'Tools', and 'Help'. A message indicates the file was last saved at 2:19 PM. On the right side, there are 'Comment' and 'Share' buttons, with a gear icon for settings. A yellow arrow labeled '1: Click' points to the 'Share' button. Below it, a yellow box highlights the 'Share with people and groups' section. Another yellow box labeled '15.1: Share settings' points to the gear icon. A third yellow box labeled '15.2 (Option 1): Share via email(s) - Enter collaborators' emails, invitation message, and set their respective access roles' contains instructions for adding people. A yellow arrow labeled '2: Popup display' points to a modal window titled 'Share with people and groups' which lists 'Bonaventure Molokwu (you)' as an owner. A 'Done' button is visible in the bottom right of this window. At the bottom, a 'Get link' option is shown with a yellow box labeled '15.3 (Option 2): Share via URL - Configure URL permissions and access roles for collaborators'.

File Edit View Insert Runtime Tools Help Last saved at 2:19 PM

Code Text

# Import libraries

Share

1: Click

15.1: Share settings

15.2 (Option 1): Share via email(s) - Enter collaborators' emails, invitation message, and set their respective access roles

Share with people and groups

Add people and groups

Bonaventure Molokwu (you)

Owner

Done

Send feedback to Google

15.3 (Option 2): Share via URL - Configure URL permissions and access roles for collaborators

Get link

Restricted Only people added can open with this link

Change to anyone with the link

Copy link

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments & Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive Widgets

Linux Commands

Q & A

### 15. Share/Teamwork on your code:

#### 15.1 Alter global settings for these access roles

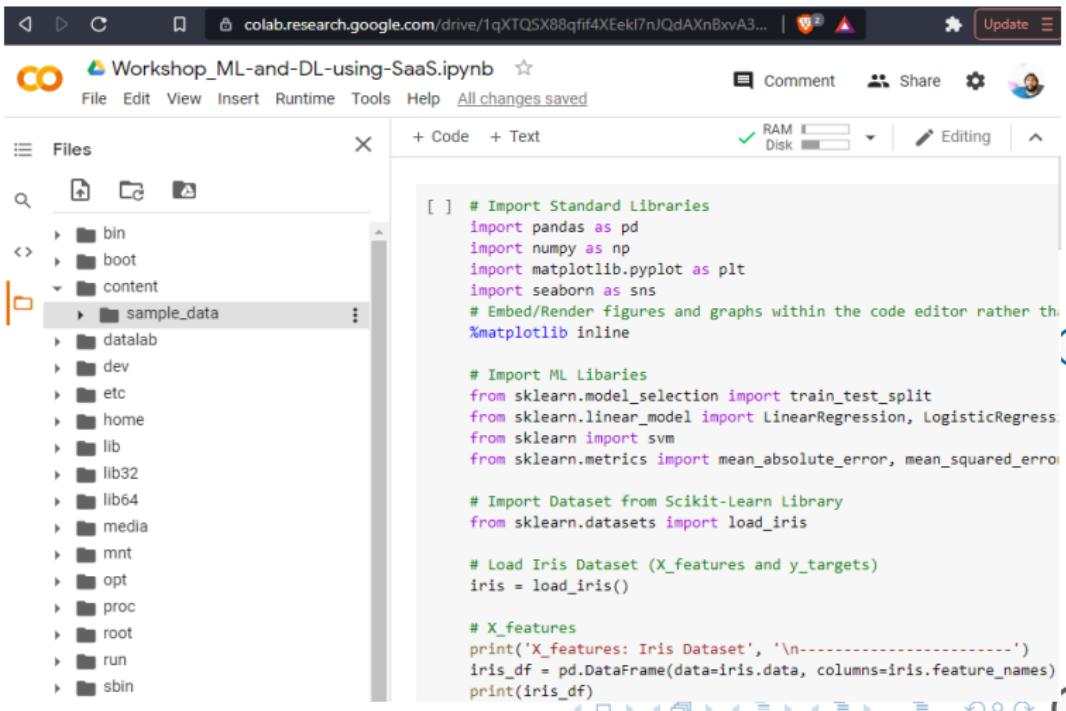
( Editors , Commenters , and Viewers ) with regard to their respective permissions and visible options (such as buttons for *download, print, copy*).

27

#### 15.2 Share/Teamwork on code, with *collaborators*, via email access.

#### 15.3 Share/Teamwork on code, with *collaborators*, via access to URL.

### 16. To explore the directory structure of code repository on Google™ Colab, use the left-hand panel:



The screenshot shows the Google Colab interface. On the left, there's a sidebar titled "Files" which lists various system directories like bin, boot, content, dev, etc. A folder named "sample\_data" is selected. The main area is a code editor with the following Python script:

```
[ ] # Import Standard Libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
# Embed/Render figures and graphs within the code editor rather than
%matplotlib inline

# Import ML Libraries
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LinearRegression, LogisticRegression
from sklearn import svm
from sklearn.metrics import mean_absolute_error, mean_squared_error

# Import Dataset from Scikit-Learn Library
from sklearn.datasets import load_iris

# Load Iris Dataset (X_features and y_targets)
iris = load_iris()

# X_features
print('X_features: Iris Dataset', '\n-----')
iris_df = pd.DataFrame(data=iris.data, columns=iris.feature_names)
print(iris_df)
```

At the bottom of the code editor, there are several navigation icons: back, forward, search, and refresh.

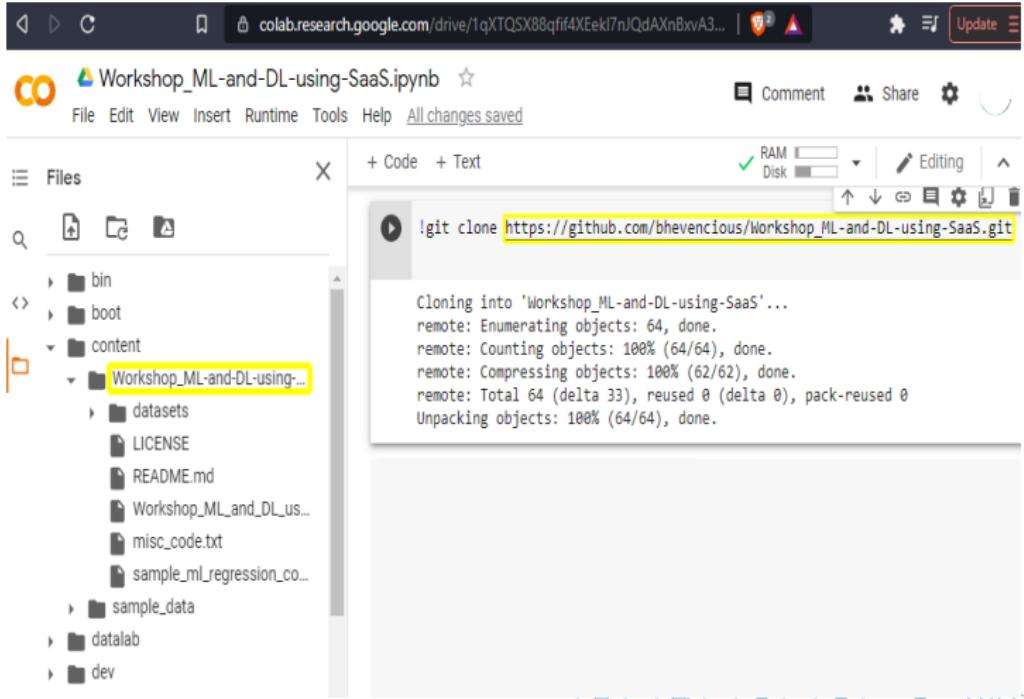


### 17. Use the **bash symbol (!)** prefix, in Google™ Colab, to run shell commands:

The screenshot shows a Google Colab interface with a terminal window. The terminal tab is active, showing the command `!ls /bin`. The output of the command is a list of system binary files, starting with `bash` and ending with `zforce`. The interface includes tabs for Code and Text, and status indicators for RAM and Disk usage.

```
!ls /bin
bash bunzip2 bzcat bzcmp bzdiff bzgrep bzexe bzfgrep bzgrep bzip2 bzip2recover bzless bzmore cat chgrp chmod chown cp dash date dd df dir dmesg dnsdomainname domainname echo egrep false fgrep findmnt fuser
hostname journalctl kill kmod less lessecho lessfile lesskey lesspipe ln login logindctl ls lsblk lsmod mkdir mknod mktemp more mount mountpoint mv networkctl nisdomainname pidof ps pwd rbash readlink rm rmdir run-parts
su sync systemctl systemd less
systemd-ask-password
systemd-escape
systemd-hwdb
systemd-inhibit
systemd-machine-id-setup
systemd-notify
systemd-sysusers
systemd-tmpfiles
systemd-tty-ask-password-agent
tar
tempfile
touch
true
udevadm
unlockmgr_server
umount
uname
uncompress
vdir
wdctl
which
ypdomainname
zcat
zcmp
zdiff
zegrep
zfgrep
zforce
```

18. To clone a GitHub repository, use `!git clone <url>` command:



The screenshot shows a Google Colab notebook titled "Workshop\_ML-and-DL-using-SaaS.ipynb". The left sidebar displays a file tree with a yellow box highlighting the "Workshop\_ML-and-DL-using-SaaS" directory. The main workspace shows a terminal window with the command `!git clone https://github.com/bhevencious/Workshop_ML-and-DL-using-SaaS.git` and its execution output:

```
Cloning into 'Workshop_ML-and-DL-using-SaaS'...
remote: Enumerating objects: 64, done.
remote: Counting objects: 100% (64/64), done.
remote: Compressing objects: 100% (62/62), done.
remote: Total 64 (delta 33), reused 0 (delta 0), pack-reused 0
Unpacking objects: 100% (64/64), done.
```

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

30

Files/Directories

SQL Databases

CPU vs. GPU

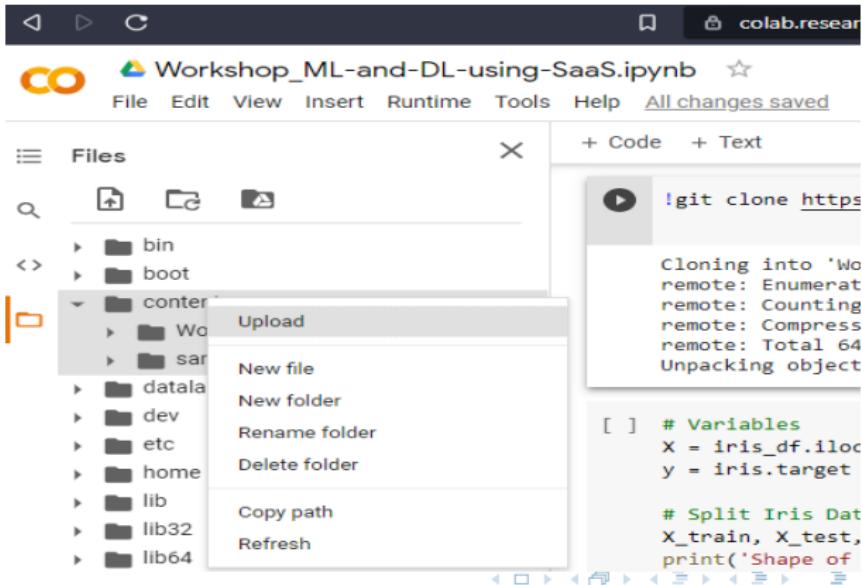
Data Forms

Interactive  
Widgets

Linux Commands

Q & A

19. To upload a dataset/file to your present working directory in Google™ Colab: Right-click on your current directory → Select the Upload menu option to open the upload dialog box.



ML & DL  
using SaaS  
via Google  
Colab

Introduction  
Terminologies  
IaaS  
PaaS  
SaaS

Colab™

Basics  
Code Comments & Documentation  
Save/Store Code  
Share/Teamwork  
Files/Directories  
SQL Databases  
CPU vs. GPU  
Data Forms  
Interactive Widgets  
Linux Commands

Q & A

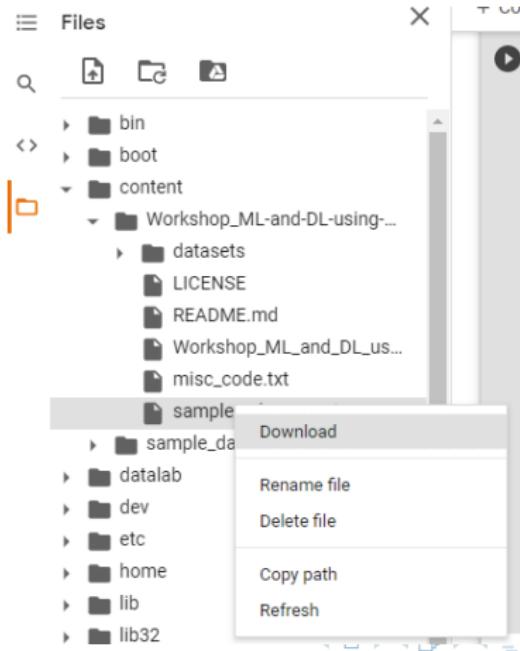
### 20. Load/read a dataset/file present in your Google™ Colab working directory:

```
...
Read a local dataset/file (in .csv format) present in your working directory
USE: pd.read_csv("<LOCAL_URL>")
...
a = pd.read_csv("sample_data/california_housing_test.csv")
a.head()
```

### 21. Load/read a dataset/file present in a remote (GitHub) directory:

```
...
# Read a local dataset/file (in .csv format) present in a remote GitHub directory
USE: pd.read_csv("<REMOTE/RAW_URL>")
...
b = pd.read_csv("https://raw.githubusercontent.com/bhevencious/Workshop_ML-and-\
DL-using-SaaS/main/datasets/california_housing_test.csv") # MUST use the 'raw' link/url from GitHub
b.head()
```

22. File manipulations via your Google™ Colab working directory: Right-click on a file/dataset → Select download (or any menu option) to proceed.



33

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

54

### 23. Load/Read databases from Google™ Colab: MySQL Database .

#### 23.1 Create and use free MySQL databases via:

<https://www.freysqlldatabase.com/register/>

#### 23.2 Thereafter, create a MySQL table within your database. See sample code for MySQL table:

[https://github.com/bhevencious/Workshop-ML-and-DL-using-SaaS/blob/main/misc\\_code.txt](https://github.com/bhevencious/Workshop-ML-and-DL-using-SaaS/blob/main/misc_code.txt)

#### 23.3 Install/Import MySQL database query dependencies.

(a) Python package for MySQL - **pymysql**

(b) Python toolkit for communication with SQL databases - **sqlalchemy**

```
# Install and Import MySQL Database Dependencies
!pip install pymysql
import pymysql
import sqlalchemy

# Initialize Connection Hyperparameters
HOSTNAME = 'sql5.freesqldatabase.com'
USER = 'sql5421285'
PASSWORD = 'hjR44EajY2'
DATABASE = 'sql5421285'
TABLE = 'continents'

# Instantiate a MySQL Connection String
# conn_string = f'mysql+pymysql://<USER>:<PASSWORD>@<HOSTNAME>/<DATABASE>'
conn_string = f'mysql+pymysql://'+USER+':'+PASSWORD+'@'+HOSTNAME+'/'+DATABASE
engine = sqlalchemy.create_engine(conn_string)

# Run a MySQL Query
# query = f'SELECT * FROM <DATABASE>.<TABLE>'
query = f'SELECT * FROM '+DATABASE+'. '+TABLE

# Read MySQL Query Result/Data
import pandas as pd
c = pd.read_sql_query(query, engine)
c.head()
```

Requirement already satisfied: pymysql in /usr/local/lib/python3.7/dist-packages (1.0.2)

	<b>id</b>	<b>uid</b>	<b>c0</b>	<b>flag</b>	<b>authorizer</b>	<b>effdate</b>
<b>0</b>	1	1398491170		1	master	2021-06-26 07:48:34
<b>1</b>	2	1398491180	Africa	1	master	2021-06-26 07:48:34
<b>2</b>	3	1398491190	North America	1	master	2021-06-26 07:48:34
<b>3</b>	4	1398491200	South America	1	master	2021-06-26 07:48:34
<b>4</b>	5	1398491210	Europe	1	master	2021-06-26 07:48:34



University  
of Windsor

# Google<sup>TM</sup> Colab

Colab/Colaboratory SaaS: CPU Battle - Intel<sup>TM</sup> vs. AMD<sup>TM</sup>

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab<sup>TM</sup>

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

36 CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

Q & A

### ML & DL using SaaS via Google Colab

#### Introduction

Terminologies

IaaS

PaaS

SaaS

#### Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

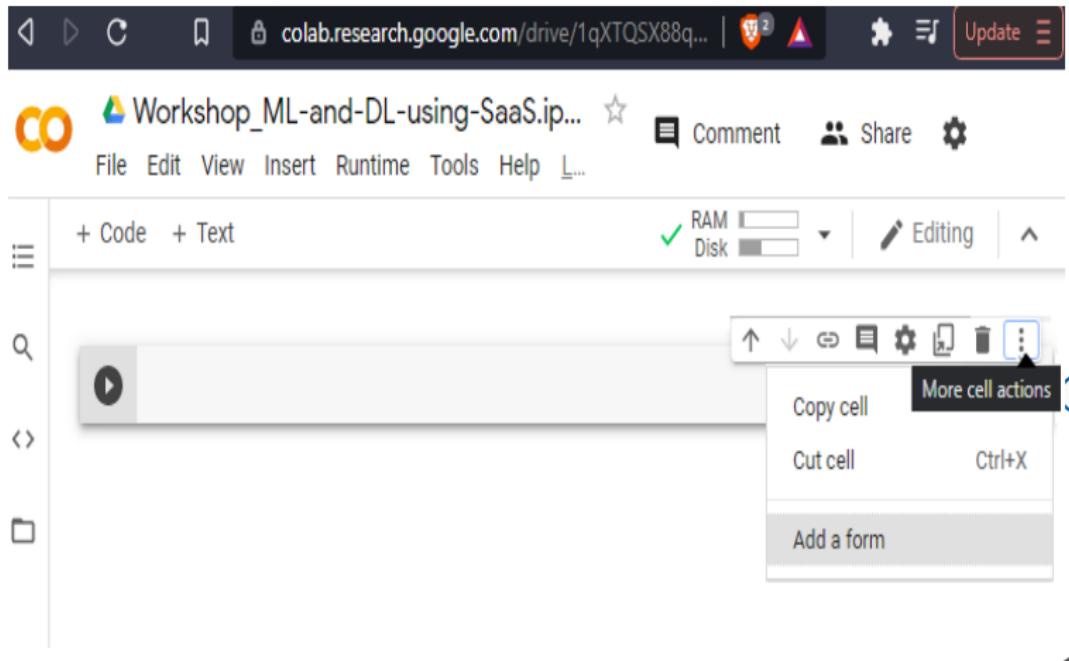
Linux Commands

#### Q & A

37

54

24. Google™ Colab Data Forms provide dynamic data communication during execution-time.



The screenshot shows the Google Colab interface. At the top, there's a toolbar with icons for back, forward, and search, followed by the URL 'colab.research.google.com/drive/1qXTQSX88q...', a progress bar, and an 'Update' button. Below the toolbar is the Colab logo and the title 'Workshop\_ML-and-DL-using-SaaS.ip...'. The main menu includes File, Edit, View, Insert, Runtime, Tools, Help, and a language selector. On the right side of the menu bar are 'Comment', 'Share', and settings icons. The main workspace has two tabs: '+ Code' and '+ Text'. In the top right corner of the workspace, there are RAM and Disk status indicators and an 'Editing' mode switch. A context menu is open over a cell, showing options like 'Copy cell', 'Cut cell' (with a keyboard shortcut 'Ctrl+X'), and 'Add a form'. The 'Add a form' option is highlighted. A blue vertical line with a circular arrow at the bottom right indicates a loop or continuation. The page number '54' is at the bottom right.

### 25. Alter properties of a Data Form as shown below:



colab.research.google.com/drive/1qXTQSX88qfif4X... | 🔍 ⚡ ⚡ Update

Workshop\_ML-and-DL-using-SaaS.ip...

File Edit View Insert Runtime Tools Help AI...

+ Code + Text

RAM Disk

Editing

#@title Default title text

Default title text

Edit form attributes

Edit this selection to alter the title of the form

Click here to alter/set form properties

### 26. Data-Form properties dialog/window:

ML & DL  
using SaaS  
via Google  
Colab

## Edit form attributes

Title text

Input (Data) Form for Testing our Iris-ML Model

Initial form visibility

Last shown (default)

Auto-execute cell when fields change

Display output below form

Form width

px

40

Cancel

Save

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

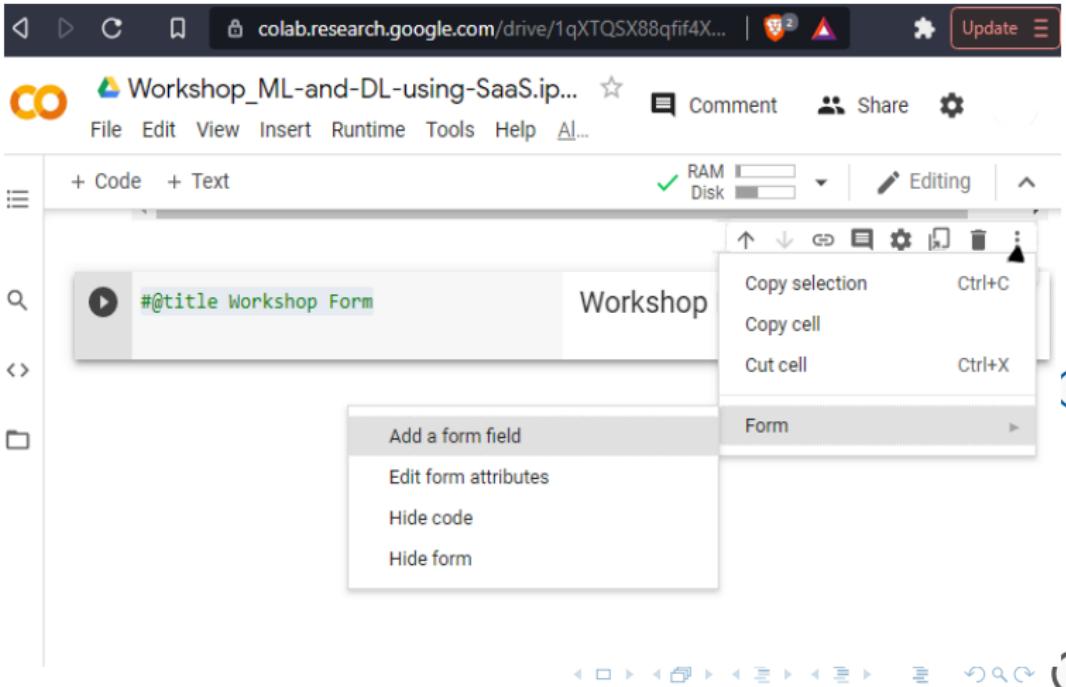
Interactive  
Widgets

Linux Commands

Q & A

27.1 Edit attributes → Form → Add a form field OR

27.2 Insert → Add a form field :



The screenshot shows the Google Colab interface with a code cell containing the code `#@title Workshop Form`. A context menu is open over the code cell, with the "Add a form field" option highlighted. The menu also includes options like "Edit form attributes", "Hide code", and "Hide form". The Colab toolbar at the top shows the file name "Workshop\_ML-and-DL-using-SaaS.ipynb" and various tool icons. The sidebar on the left shows a search bar, a folder icon, and a list of recent files.

### 28. Form fields:

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

42 Data Forms

Interactive  
Widgets

Linux Commands

Q & A

### Add new form field

Form field type: input

Variable name: variable\_name

Variable type: string

Form field type dropdown menu:

- dropdown
- input** (selected)
- slider
- markdown

Cancel Save

## 29. Implementation of interactive fields via Data Forms (Code):

```
① #@title Input (Data) Form for Testing our Iris-ML Model { run: "auto", vertical-output: true }
#@markdown ---

#@markdown Petal Length (cm)
petal_length = 4.0 #@param {type:"number"}

#@markdown Petal Width (cm)
petal_width = "2.6" #@param ["0.2", "0.4", "0.6", "0.8", "1.0", "1.2", "1.4", "1.6", "1.8", "2.0", "2.2", "2.4", "2.6"]

#@markdown Sepal Length (cm)
sepal_length = 3 #@param {type:"slider", min:0, max:6, step:0.1}

#@markdown Sepal Width (cm)
sepal_width = 1.8 #@param {type:"slider", min:0, max:3, step:0.1}

# Packaging Input for ML-Model Evaluation
temp_df_row = pd.DataFrame([[sepal_length, sepal_width, petal_length, petal_width]])
temp_pred = model.predict(temp_df_row)

result = 'setosa'
if round(temp_pred[0]) == 0:
    result = 'setosa'
elif round(temp_pred[0]) == 1:
    result = 'versicolor'
elif round(temp_pred[0]) == 2:
    result = 'virginica'

#@markdown **The resultant type of Iris flower is:**
print(result)
```

43

Data Forms  
Interactive Widgets  
Linux Commands

Q & A

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

### 30. Implementation of interactive fields via Data Forms (Output):

ML & DL  
using SaaS  
via Google  
Colab

Input (Data) Form for Testing our Iris-ML Model

Petal Length (cm)

`petal_length:` 4.0

Petal Width (cm)

`petal_width:` 2.6

Sepal Length (cm)

`sepal_length:` 3

Sepal Width (cm)

`sepal_width:` 1.8

The resultant type of Iris flower is:

virginica

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

44 Data Forms

Interactive  
Widgets

Linux Commands

Q & A

31. Alternative for providing dynamic data communication during execution-time.

### 31.1 Interactive Python Widgets == **ipywidgets**

### 31.2 Detailed documentation ==

<https://ipywidgets.readthedocs.io/en/latest/examples/Widget%20Basics.html>

### 31.3 Dependencies:

31.3.1 **import ipywidgets as widgets**

31.3.2 **from IPython.display import display**

45

Interactive  
Widgets

Linux Commands

Q & A

### 32. Interactive Python Widgets for dynamic data communication during execution-time ( **Code** ).

ML & DL  
using SaaS  
via Google  
Colab

```
import ipywidgets as widgets
from IPython.display import display

# Petal Length (cm)
petal_length = widgets.FloatText(value=0.2, description='Petal Length (cm):', continuous_update=True, disabled=False)

# Petal Width (cm)
petal_width = widgets.Dropdown(options=["0.2", "0.4", "0.6", "0.8", "1.0", "1.2", "1.4", "1.6", "1.8", "2.0", "2.2", "2.4"],
                                value="0.2", description='Petal Width (cm):', continuous_update=True, disabled=False)

# Sepal Length (cm)
sepal_length = widgets.FloatSlider(value=1.5, min=0, max=6.0, step=0.1, description='Sepal Length (cm):', disabled=False,
                                    continuous_update=True, orientation='horizontal', readout=True, readout_format='.1f')

# Sepal Width (cm)
sepal_width = widgets.FloatSlider(value=0.5, min=0, max=3.0, step=0.1, description='Sepal Width (cm):', disabled=False,
                                    continuous_update=True, orientation='horizontal', readout=True, readout_format='.1f')

# Display all Interactive Widget Tools to Screen
display(petal_length, petal_width, sepal_length, sepal_width)
```

Petal Length...	0.2
Petal Width...	2.4
Sepal Length...	<input type="range" value="1.5"/> 1.3
Sepal Width...	<input type="range" value="0.5"/> 0.7

46

Interactive  
Widgets

Linux Commands

Q & A

### 33. Interactive Python Widgets for dynamic data communication during execution-time ( Output ).

```
[97] # Packaging Input for ML-Model Evaluation
      temp_df_row = pd.DataFrame([[sepal_length.value, sepal_width.value, petal_length.value, petal_width.value]])
      temp_pred = model.predict(temp_df_row)

      result = 'setosa'
      if round(temp_pred[0]) == 0:
          result = 'setosa'
      elif round(temp_pred[0]) == 1:
          result = 'versicolor'
      elif round(temp_pred[0]) == 2:
          result = 'virginica'
      else:
          result = 'UNKNOWN'

      print('The resultant type of Iris flower is: ', result)
```

The resultant type of Iris flower is: versicolor



# Basic Commands

## Basic Unix-based (Linux) Commands

1. **cat** (*!cat ./sample\_data/README.md*) - alias for “concatenate”, and it is majorly used to manipulate the content of files in Unix-based environments.
2. **pwd** (*!pwd*) - prints the name of the current working directory.
3. **cd** (*!cd sample\_data/*) - used to change/switch your current (working) directory.
4. **ls** (*!/s*) - list contents of the current working directory.
5. **exit** (*exit*) - used to close a terminal window, end the execution of a shell script, or log you out of an SSH remote access session.



# Basic Commands

## Basic Unix-based (Linux) Commands

6. **history** (*%history*) - lists all the commands you have previously issued via terminal.
7. **clear** (*!clear*) - clears the current screen content of any text/command.
8. **mkdir** (*!mkdir newFolder*) - create new directories in the filesystem.
9. **cp** (*!cp sample\_data/README.md newFolder/*) - to copy files and/or directories from one directory to another directory. `cp <source> <dest>`.
10. **rm** (*!rm newFolder/README.md*) - removes or deletes a file from a directory within the filesystem.

# Basic Commands

## Basic Unix-based (Linux) Commands



University  
of Windsor

11. **mv** (*!mv sample\_data/mnist\_test.csv newFolder*)  
- to move files and/or directories from one directory to another directory. mv <source> <dest>.
12. **rmdir** (*!rmdir newFolder*) - removes/deletes an EMPTY directory from the filesystem.
13. **rm -r** (*!rm -r newFolder*) - removes or deletes a directory and its content from the filesystem.
14. **man <command>** (*!man ls*) - displays the manual with respect to any command.
15. **<command> --help** (*!ls --help*) - displays the help information with regard to the usage of a command.

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab™

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms  
Interactive  
Widgets

50 Linux Commands

Q & A

# Basic Commands

## Basic Unix-based (Linux) Commands



University  
of Windsor

16. **curl** (*!curl https://www.uwindsor.ca*) - used to retrieve files and information from URLs.
17. **find** (*!find sample\_data/ -name \*c\**) - track down files that you know exist but cannot remember where you kept them. **-name** = filename.
18. **free** (*!free -h*) - gives a summary of your memory usage for both the main Random Access Memory (RAM) and swap memory. **-h** = human-friendly.
19. **gzip** (*!gzip -k bobo.txt*) - compress files. **-k** = keep main.
20. **head** (*!head -n 7 bobo.txt*) - listing of the first 10 lines of a file by default. **-n** = number of lines.

ML & DL  
using SaaS  
via Google  
Colab

Introduction

Terminologies

IaaS

PaaS

SaaS

Colab<sup>TM</sup>

Basics

Code Comments  
& Documentation

Save/Store Code

Share/Teamwork

Files/Directories

SQL Databases

CPU vs. GPU

Data Forms

Interactive  
Widgets

Linux Commands

51

Q & A



# Basic Commands

## Basic Unix-based (Linux) Commands

21. **tail** (*!tail -n 7 bobo.txt*) - listing of the last 10 lines of a file by default. *-n* = number of lines.
22. **ps** (*!ps -A*) - shows all running processes. *-A* = show information for all processes.
23. **kill** (*!kill 17*) - terminates a process via the process ID (PID).
24. **top** (*!top*) - real-time display of all the processes related to your Linux machine.
25. **wget <file\_URL>** (*!wget <file\_URL>*) - downloads the content of the file specified via *file\_URL*.



# Basic Commands

## Basic Unix-based (Linux) Commands

26. **uname** (*!uname -a*) - obtain system information about your Linux machine. -a = all basic info.
27. **w** (*!w*) - list all the users who are currently logged-in.
28. **whoami** (*!whoami*) - find out whom you are currently logged-in as.
29. **lsb\_release** (*!lsb\_release -a*) - provides certain LSB (Linux Standard Base) and distribution-specific information. -a = all information.
30. **lscpu** (*!lscpu*) - displays information about the CPU(s) architecture of your machine.

**Detailed list of Linux commands:** [https://man7.org/linux/man-pages/dir\\_all\\_alpha.html](https://man7.org/linux/man-pages/dir_all_alpha.html)





# References

1. Bisong E. (2019) Google Colaboratory. In: Building Machine Learning and Deep Learning Models on Google Cloud Platform. Apress, Berkeley, CA. [https://doi.org/10.1007/978-1-4842-4470-8\\_7](https://doi.org/10.1007/978-1-4842-4470-8_7) [https://colab.research.google.com/notebooks/basic\\_features\\_overview.ipynb](https://colab.research.google.com/notebooks/basic_features_overview.ipynb)
2. Barkai, N. (2016, October 1). Distributed computing solution. Retrieved from <https://asknature.org/idea/distributed-computing-solution/>
3. NetApp. (2020, March 25). What Is High-Performance Computing. Retrieved from <https://www.netapp.com/us/info/what-is-high-performance-computing.aspx>
4. ComputeCanada. (2020, April 28). Compute Canada Documentation. Retrieved from [https://docs.computecanada.ca/wiki/Compute\\_Canada\\_Documentation](https://docs.computecanada.ca/wiki/Compute_Canada_Documentation)
5. Mckay, D. (2019, May 8). 37 Important Linux Commands You Should Know. Retrieved from <https://www.howtogeek.com/412055/37-important-linux-commands-you-should-know/>
6. Mc Lay, M. (2018, July 22). Lmod: A New Environment Module System. Retrieved from <https://lmod.readthedocs.io/en/latest/>
7. Globus. (2020, July 31). Globus. Retrieved from <https://docs.computecanada.ca/wiki/Globus>

# Questions? & Answers!



University  
of Windsor