Machine Learning and Deep Learning using Software-as-a-Service (via Google™ Colab)

July 9, 2021

Bonaventure Chidube Molokwu
molokwub[at]uwindsor.ca

School of Computer Science
University of Windsor
Ontario - Canada

Introduction

Selected Terminologies
1. Concurrent Systems

University of Windsor

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

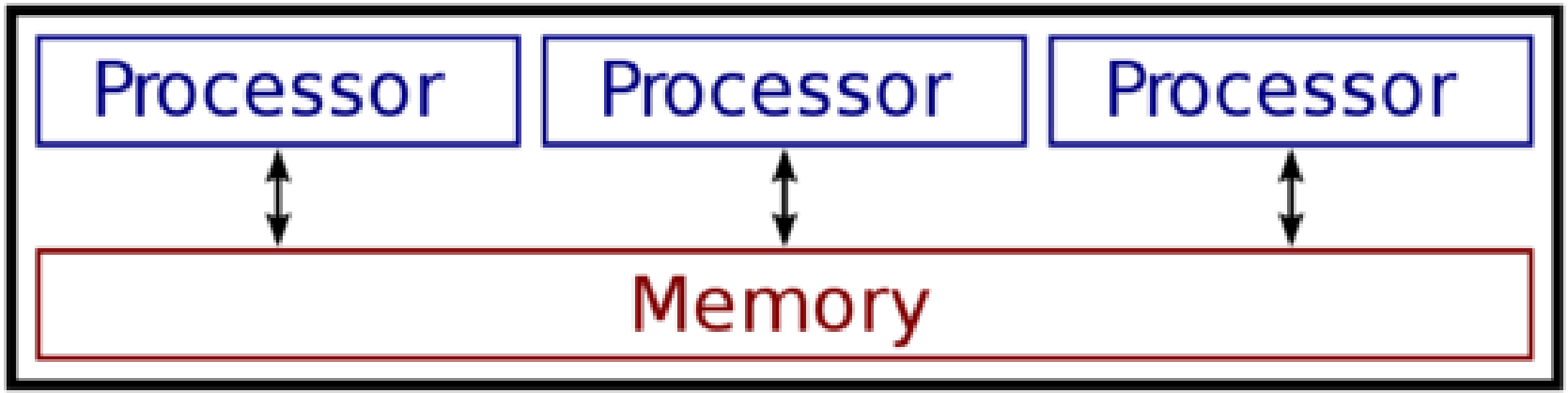


Figure: Parallel computing

1.1 **Parallel Computing:** several processors, which have access to shared memory, execute or process a task.

Selected Terminologies
1.2 Distributed Computing

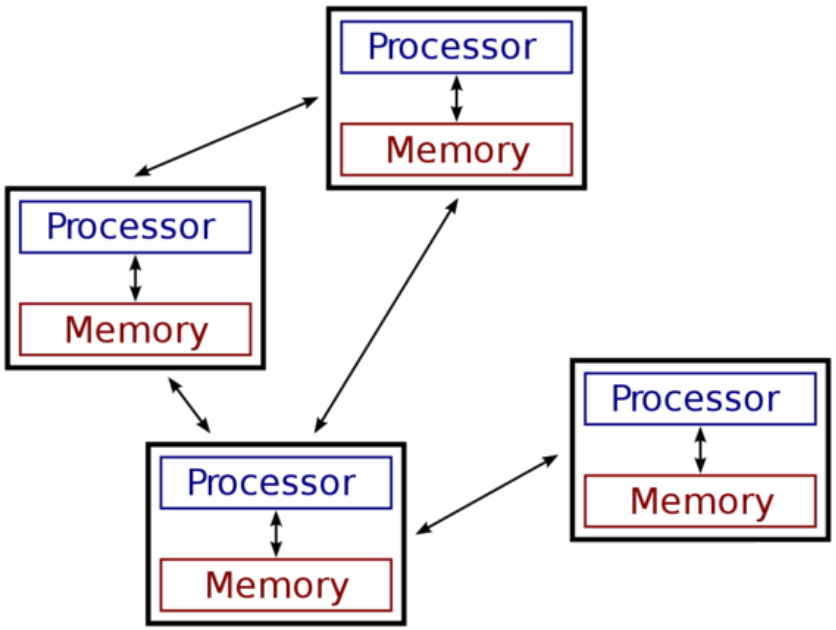
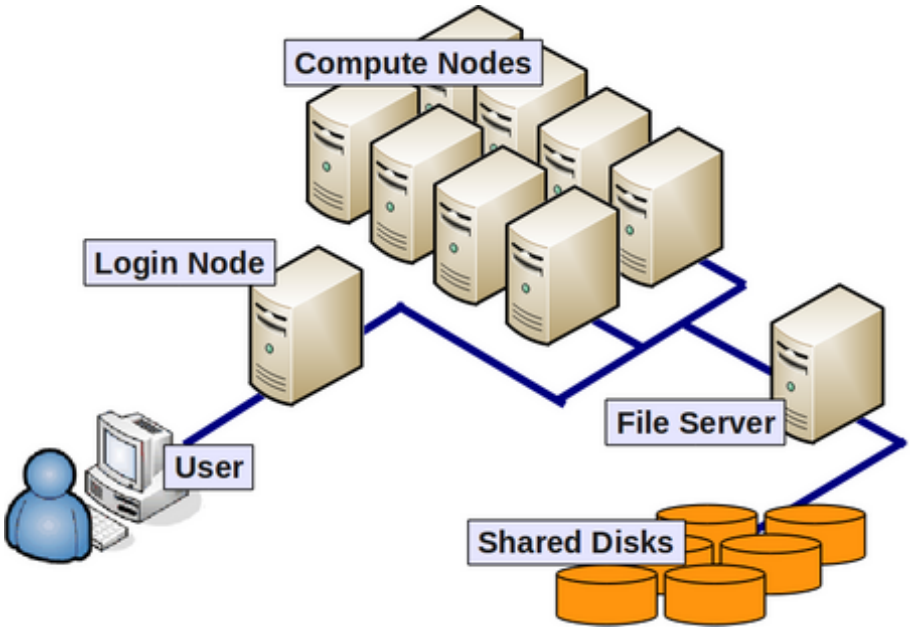


Figure: Distributed computing

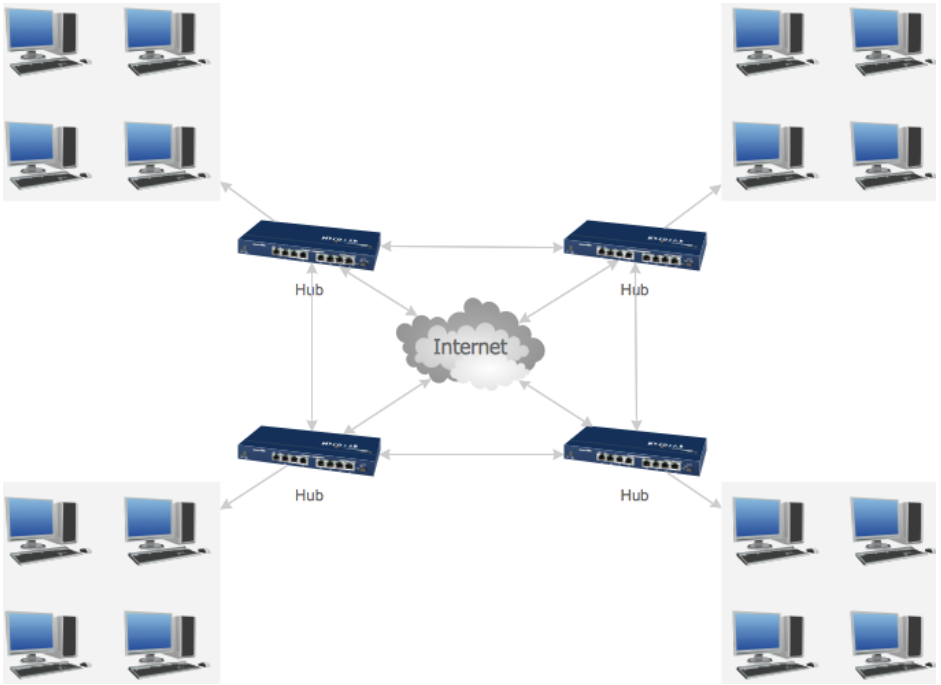
1.2 **Distributed Computing:** each processor has its own private memory for executing or processing a task. Information is exchanged via message passing between processors.

Selected Terminologies
2. Cluster Computing



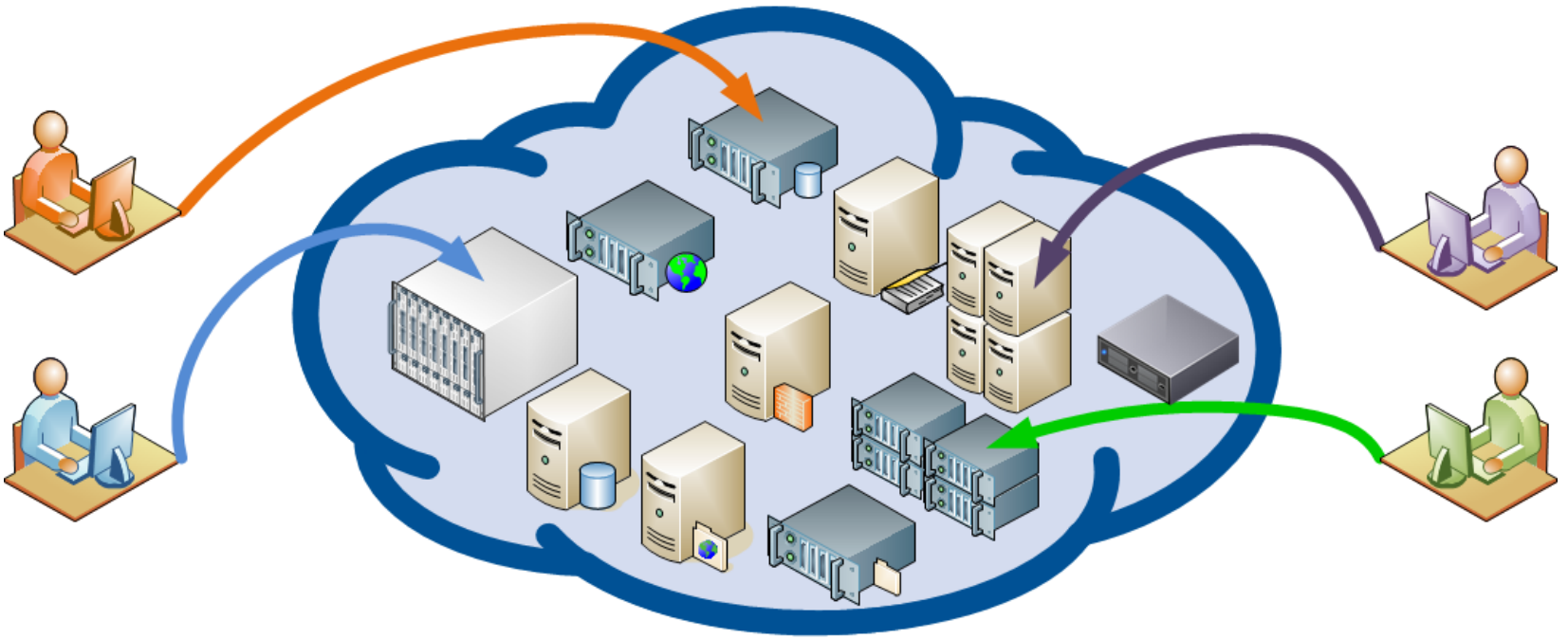
2. **Cluster Computing:** A type of Parallel Computing comprising several networked computers (nodes) using a shared storage, and all within a specific location (single system). Each node is set to perform the same task controlled by a scheduler.

Selected Terminologies
3. Grid Computing



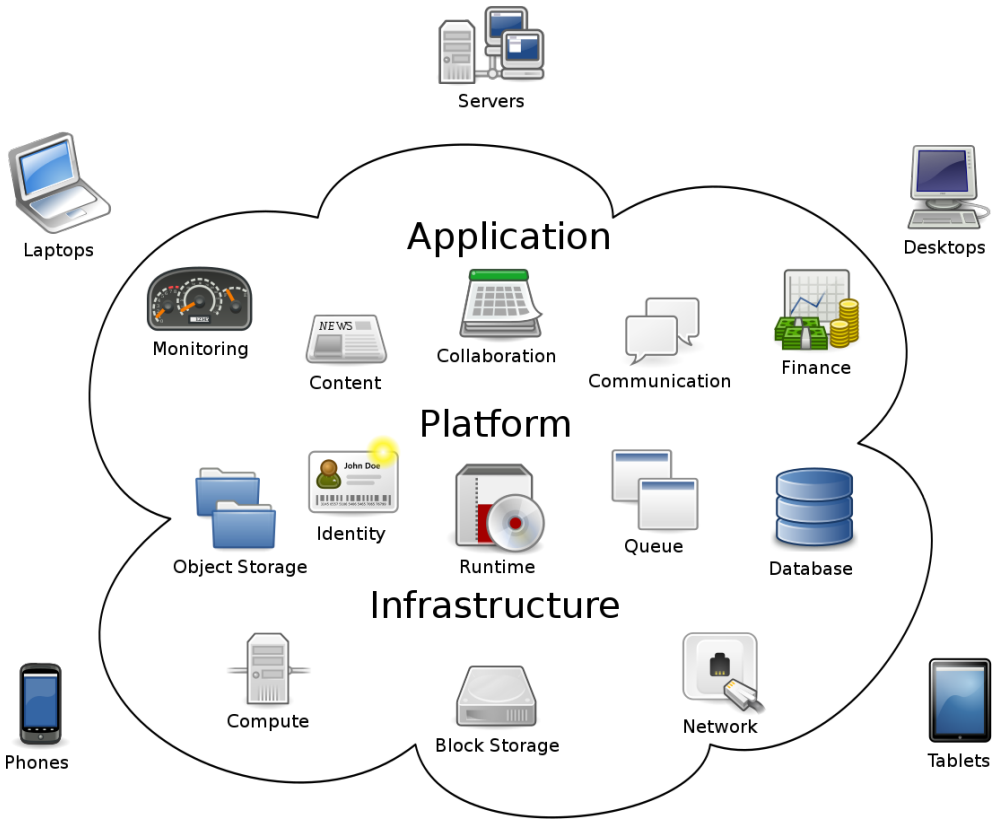
3. **Grid Computing:** Type of Distributed Computing comprising several networked computers (nodes) having independent storage, and spread across several geographical locations. Each node is set to perform different task.

Selected Terminologies
4. Virtual Computing



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



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

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.

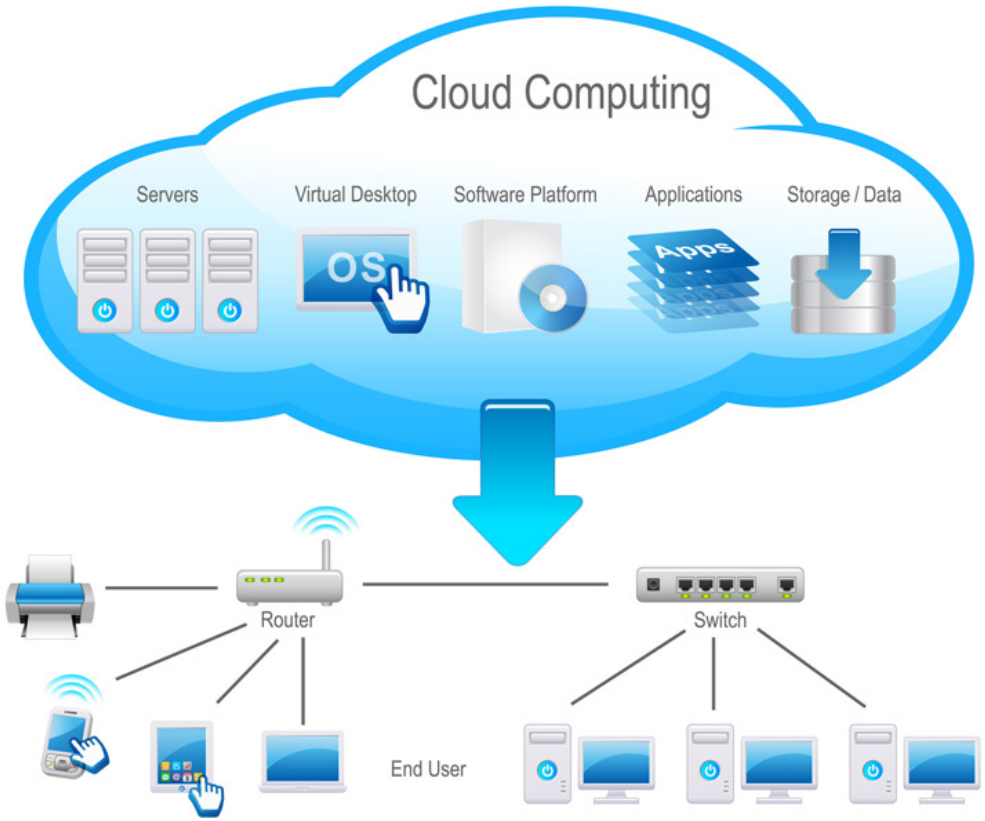
Selected Terminologies
Virtual Computing vs. Cloud Computing

University of Windsor

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

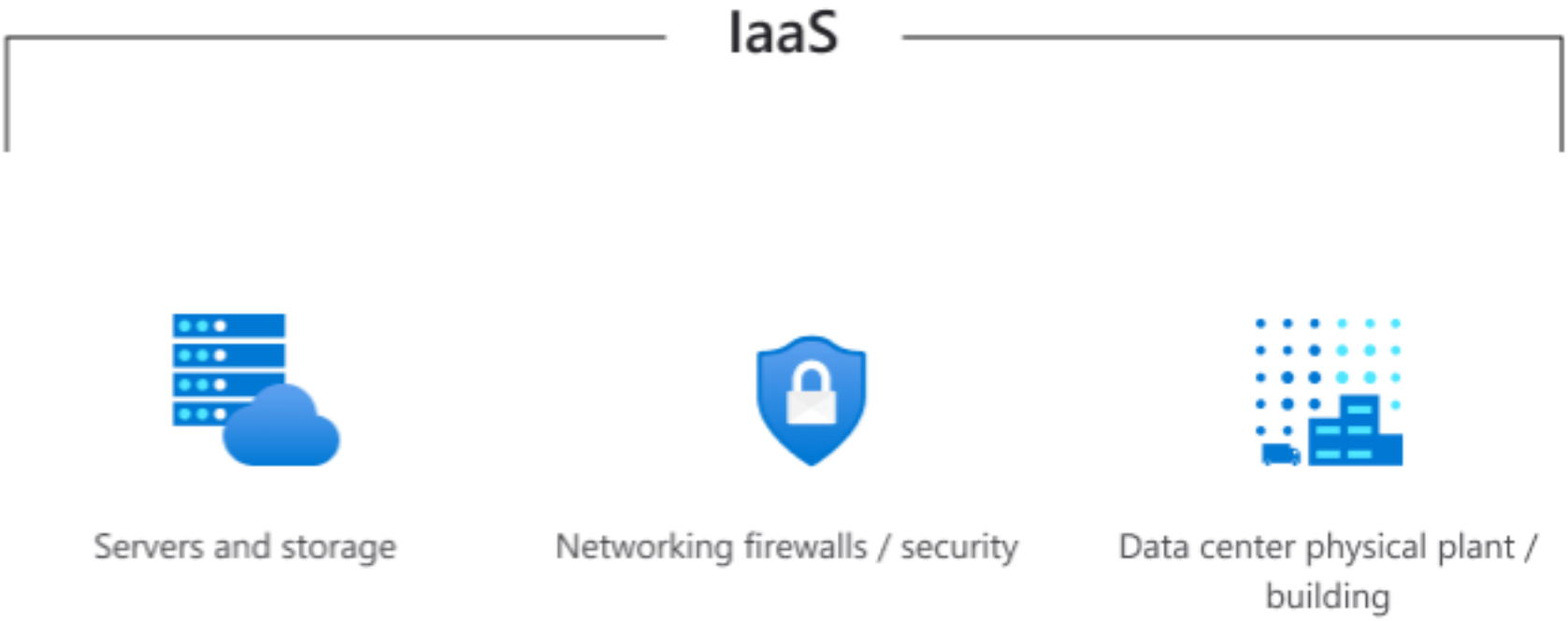


5.1 **IaaS:** Infrastructure as a Service (IaaS).

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

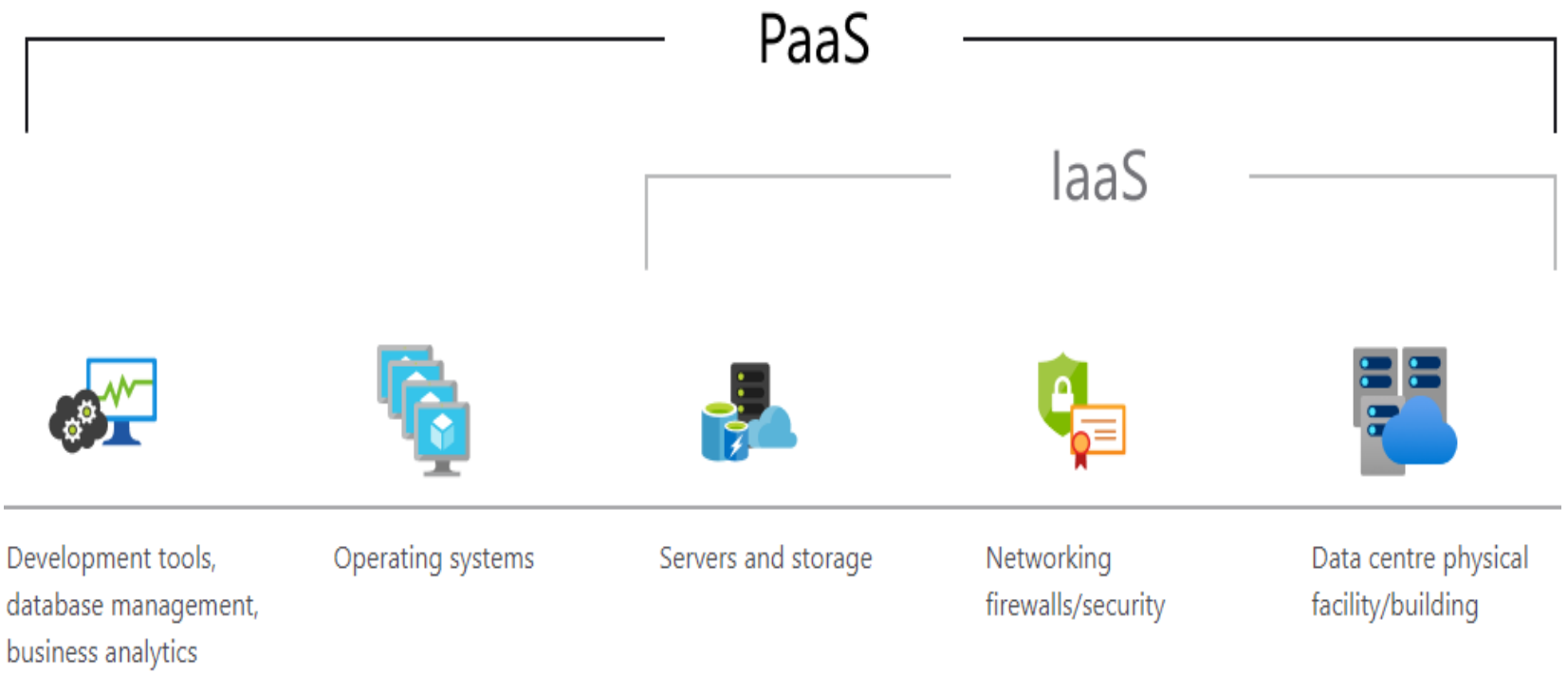
5.3 **SaaS:** Software as a Service (SaaS).

Selected Terminologies
IaaS: Infrastructure as a Service



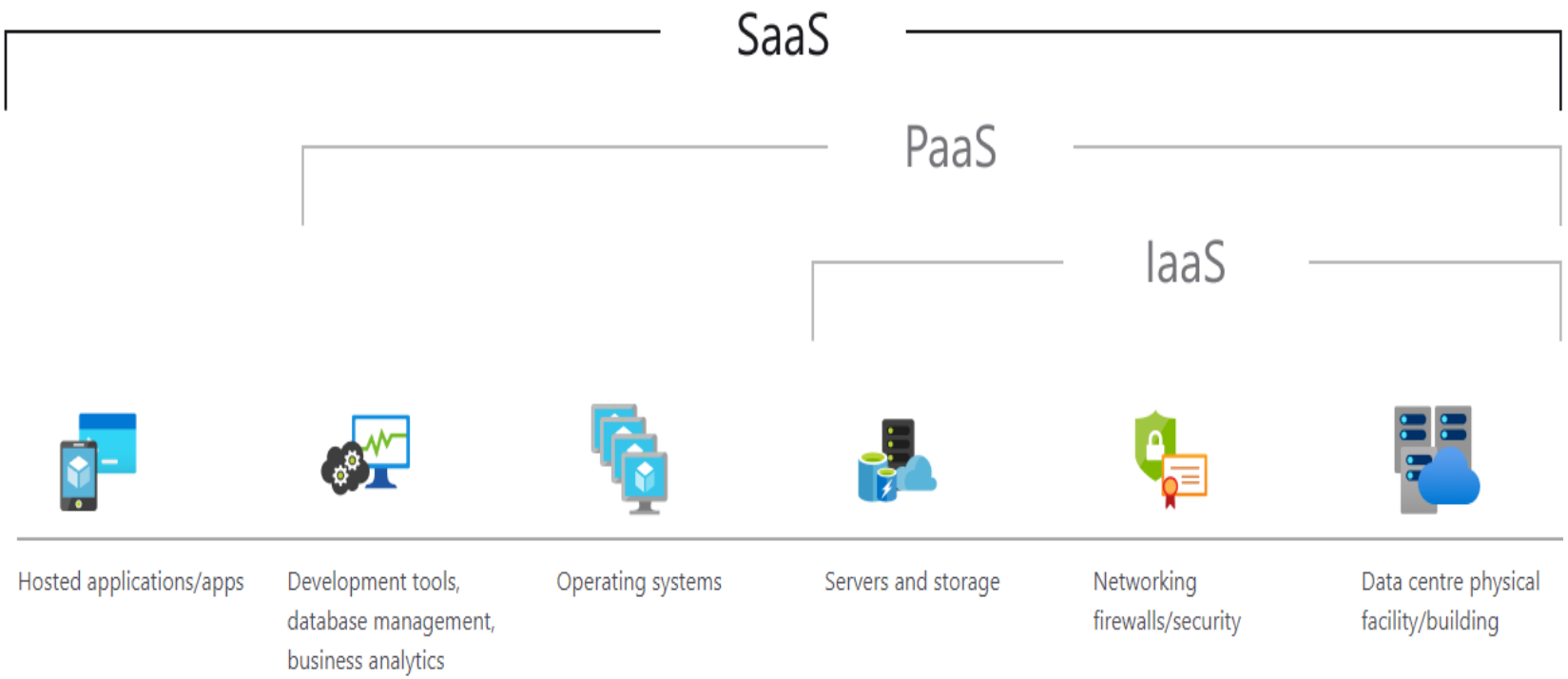
5.1 **IaaS:** Most basic category of Cloud Computing services. Provides computing hardware (such as servers, storage, networks, virtual machines (VMs), etc.) from a Cloud Service Provider on a pay-as-you-go basis.

Selected Terminologies
PaaS: Platform as a Service



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



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^{TM} Colab

Colab/Colaboratory: A Software as a Service (SaaS)

University of Windsor

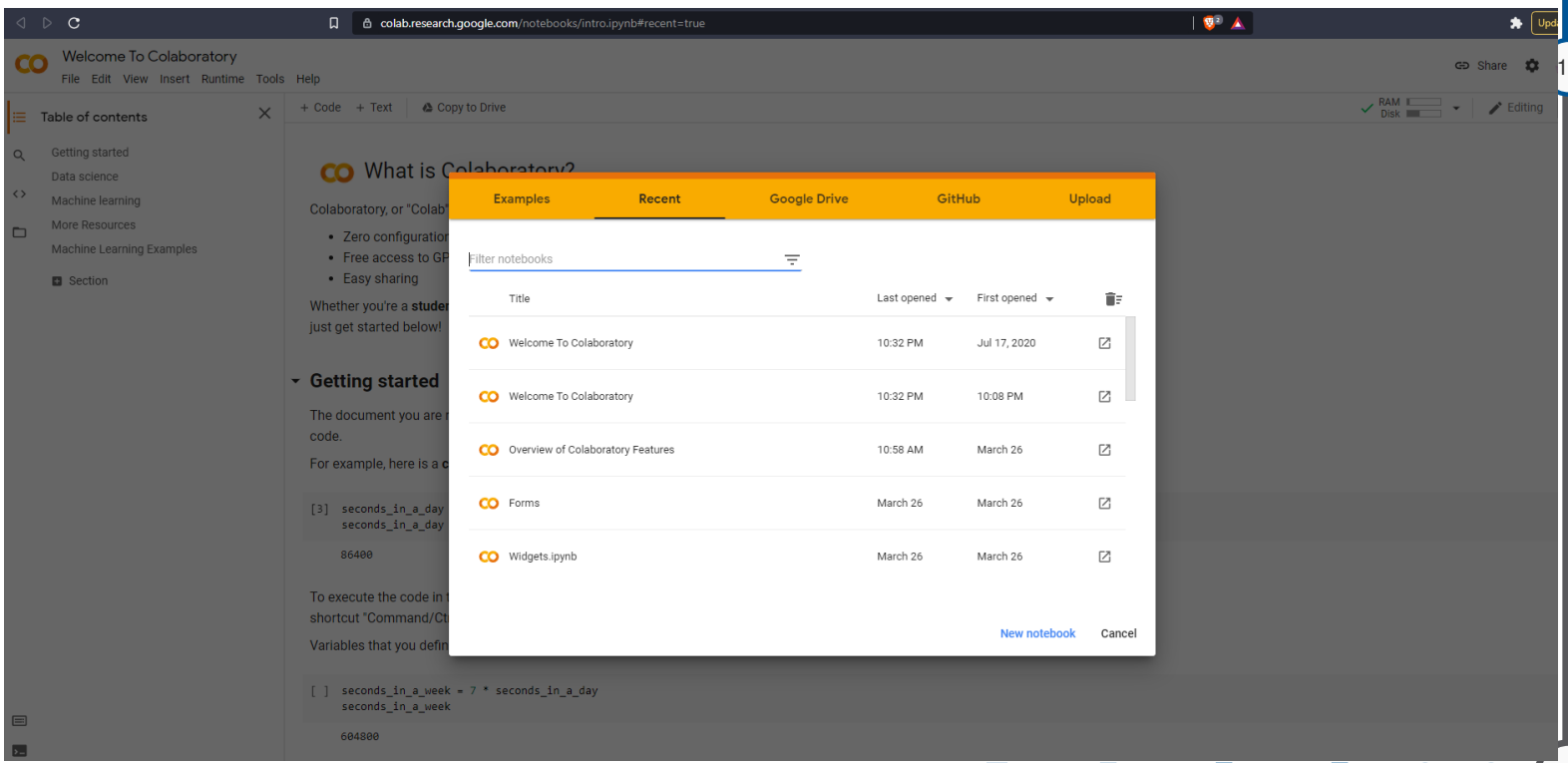
1. A SaaS cloud-based tool developed by Google^{TM}.
2. It is a free IDE for writing and implementing Python-based codes.
3. Requires relatively no configuration.
4. Provides free access to cloud-based computing resources (CPUs, GPUs, Memory, Storage, etc.).
5. Ideal for team work and collaboration.

Google™ Colab

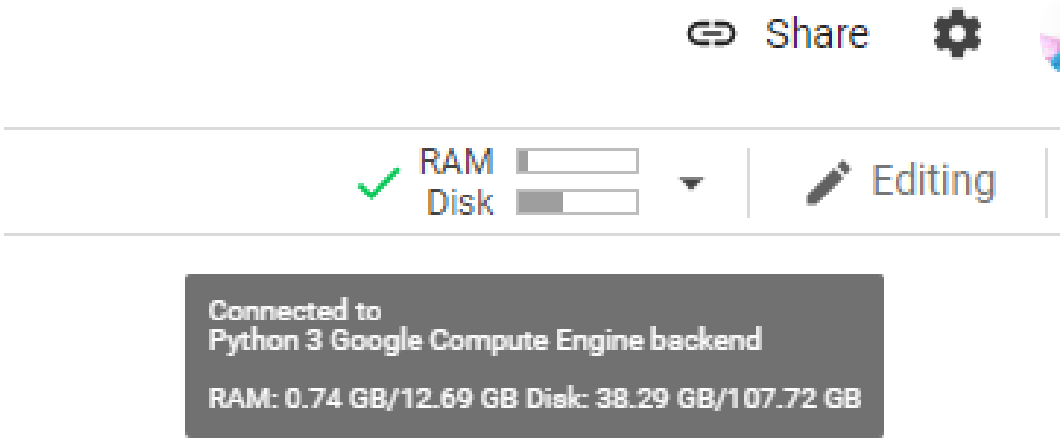
Colab/Colaboratory SaaS: Launching via Web Browser

University of Windsor

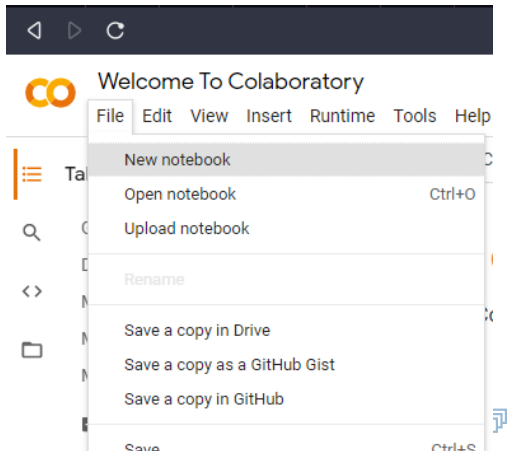
1. Ensure that you are logged onto your GMail account.
2. Proceed to the following URL: `https://colab.research.google.com/`.



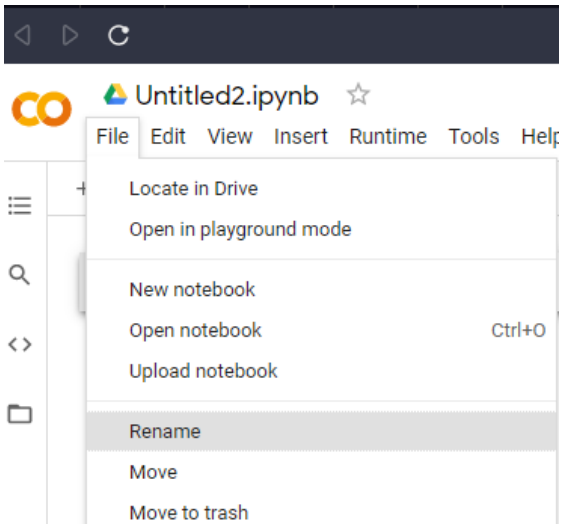
3. View current dynamically allocated resources via *hover* on the right-hand segment of the menu bar.



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

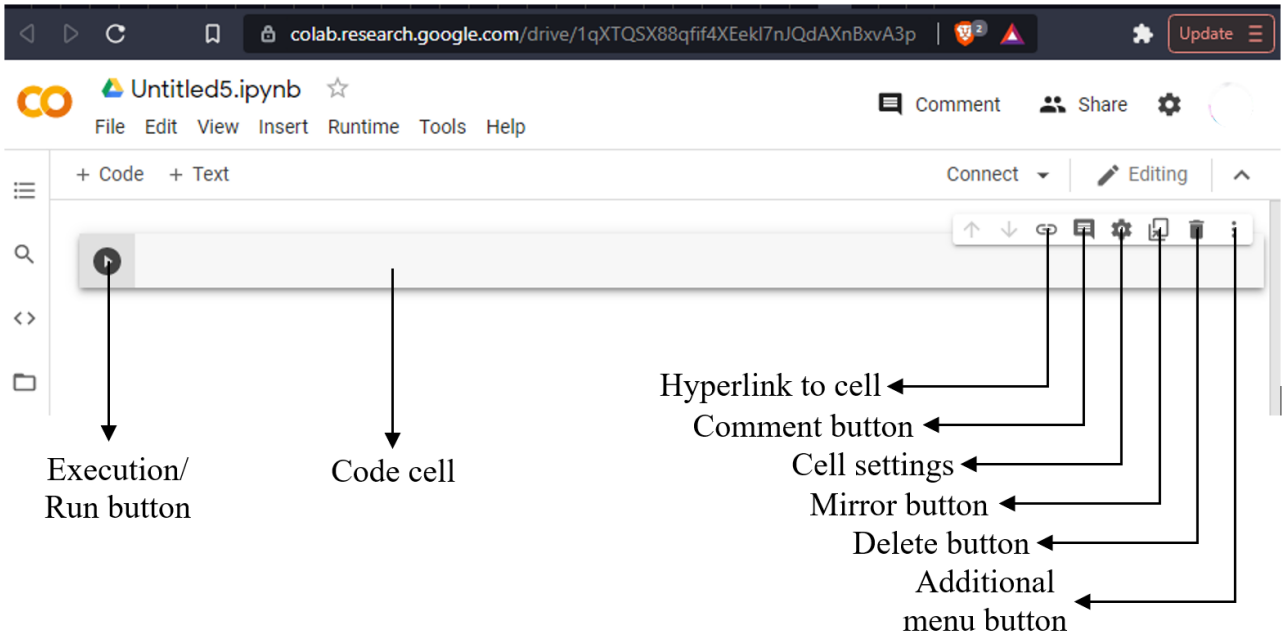


5. Rename a newly created (source) code editor:
   File → Rename

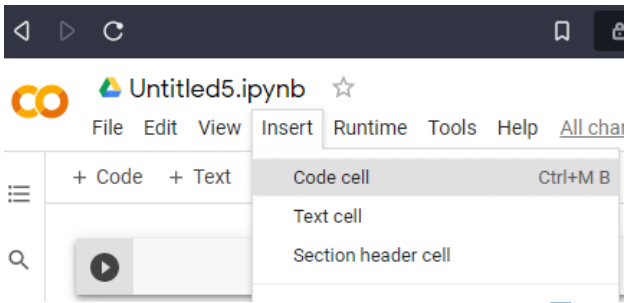


6. Alternatively, (source) editor can be renamed via:
   Double-click on the **filename**, and rename.

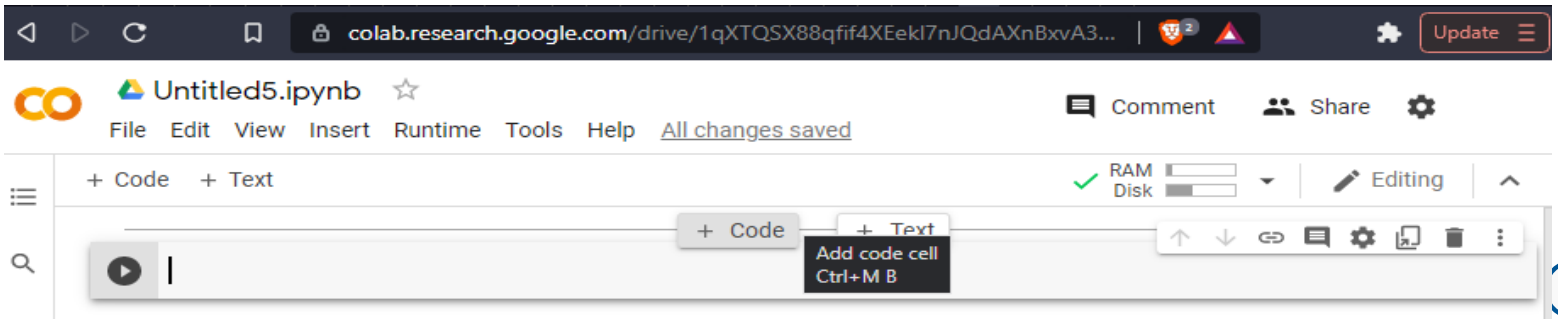
7. Code-cell basics:



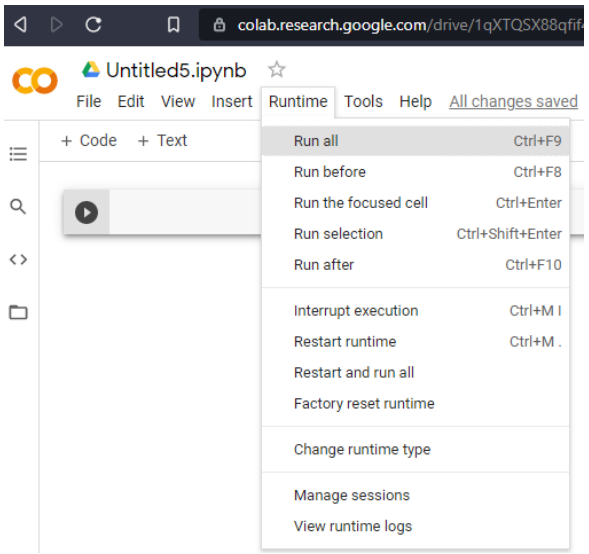
8. New code/text cell via: **Insert** → **Code/Text cell**



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

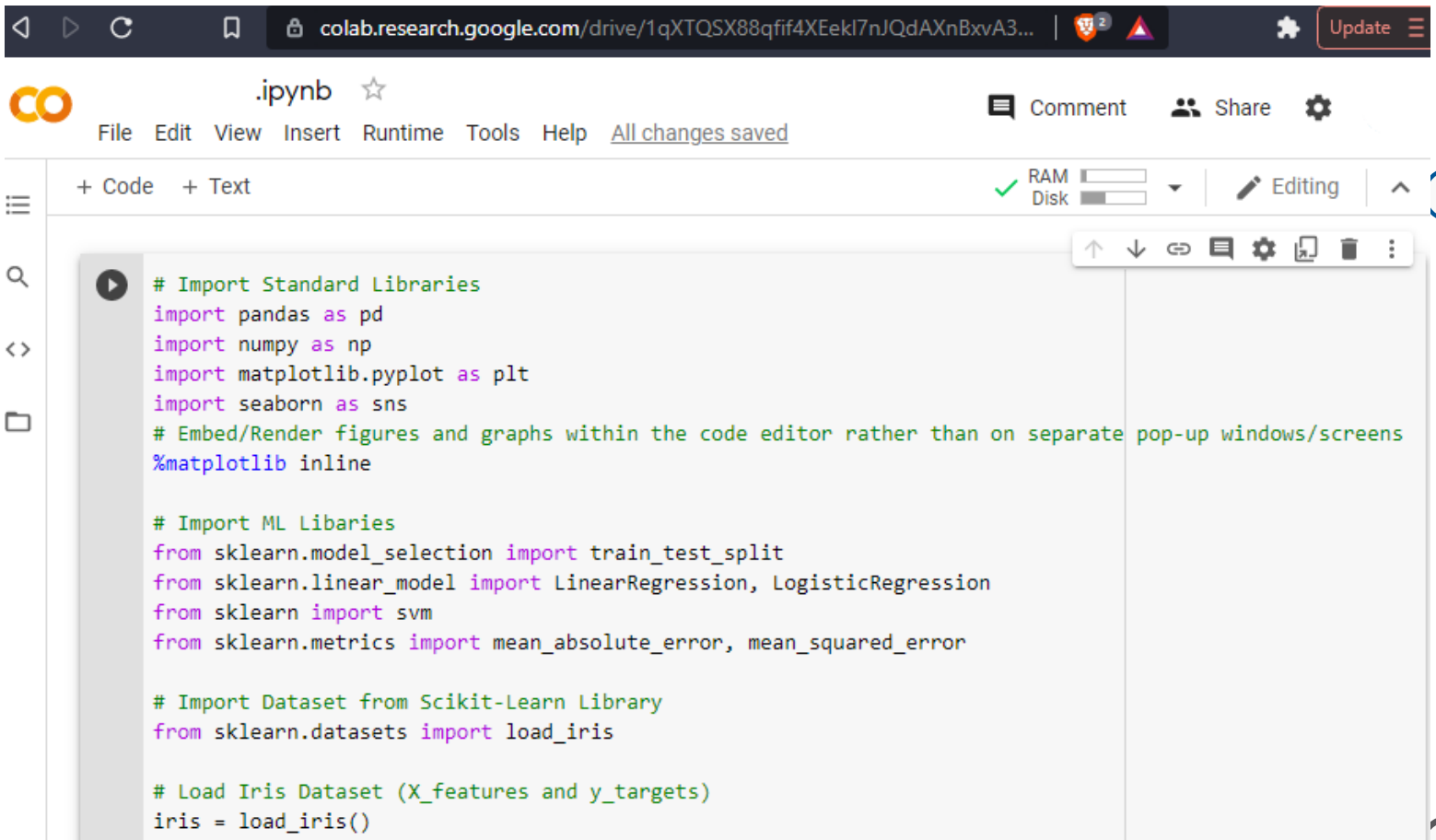


Google™ Colab

Colab/Colaboratory SaaS: Running your Python (.py) Code

University of Windsor

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

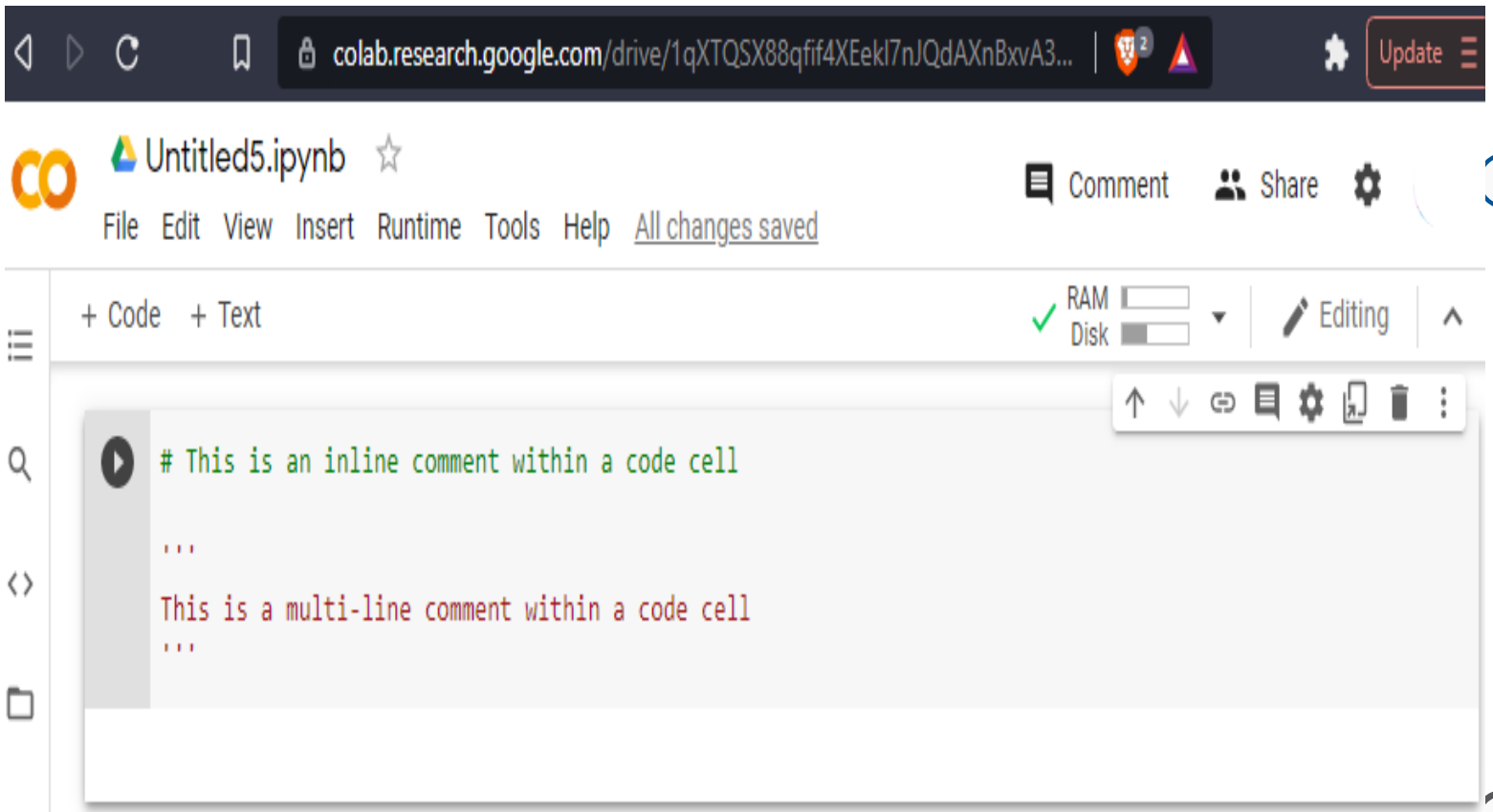


Google™ Colab

Colab/Colaboratory SaaS: Comments and Documentation

University of Windsor

12. Code documentation via code cell(s) :
    - 12.1 Inline comment (#).
    - 12.2 Multi-line or Block comment ('').



Google™ Colab

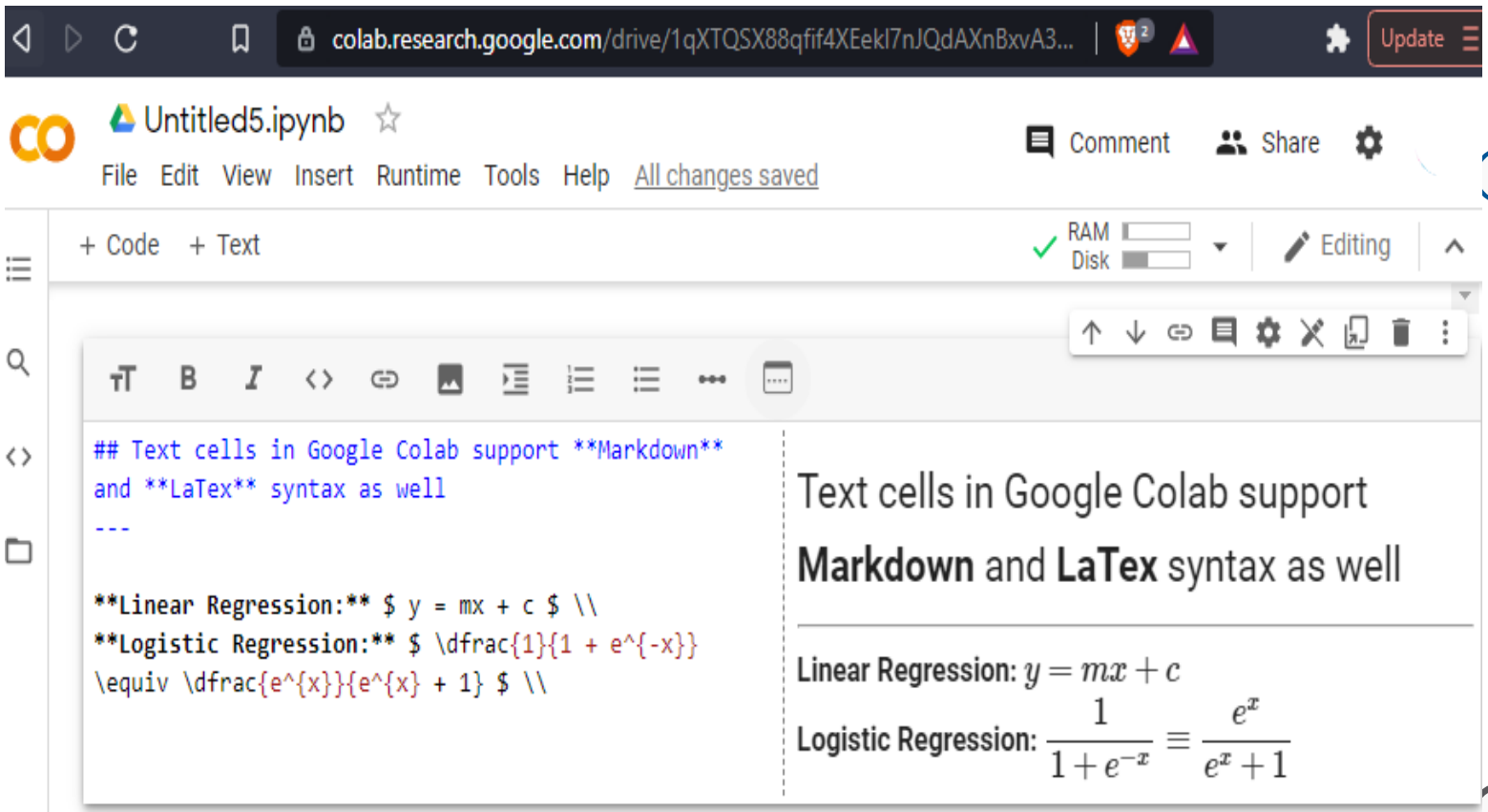
Colab/Colaboratory SaaS: Comments and Documentation

University of Windsor

13. Code documentation via text cell(s) :

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

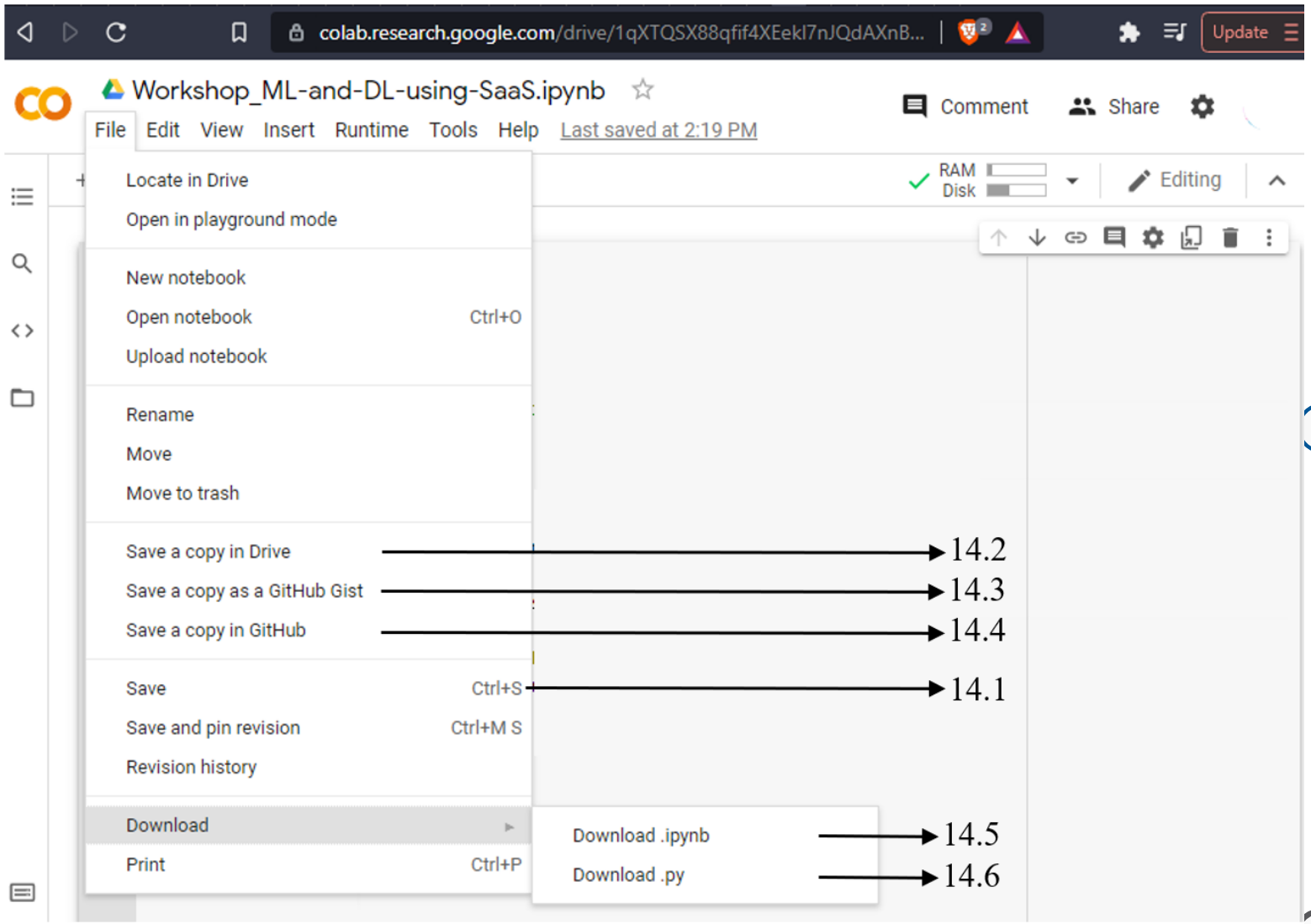
13.2 Basic guide and syntax:
`https://www.markdownguide.org/basic-syntax/`



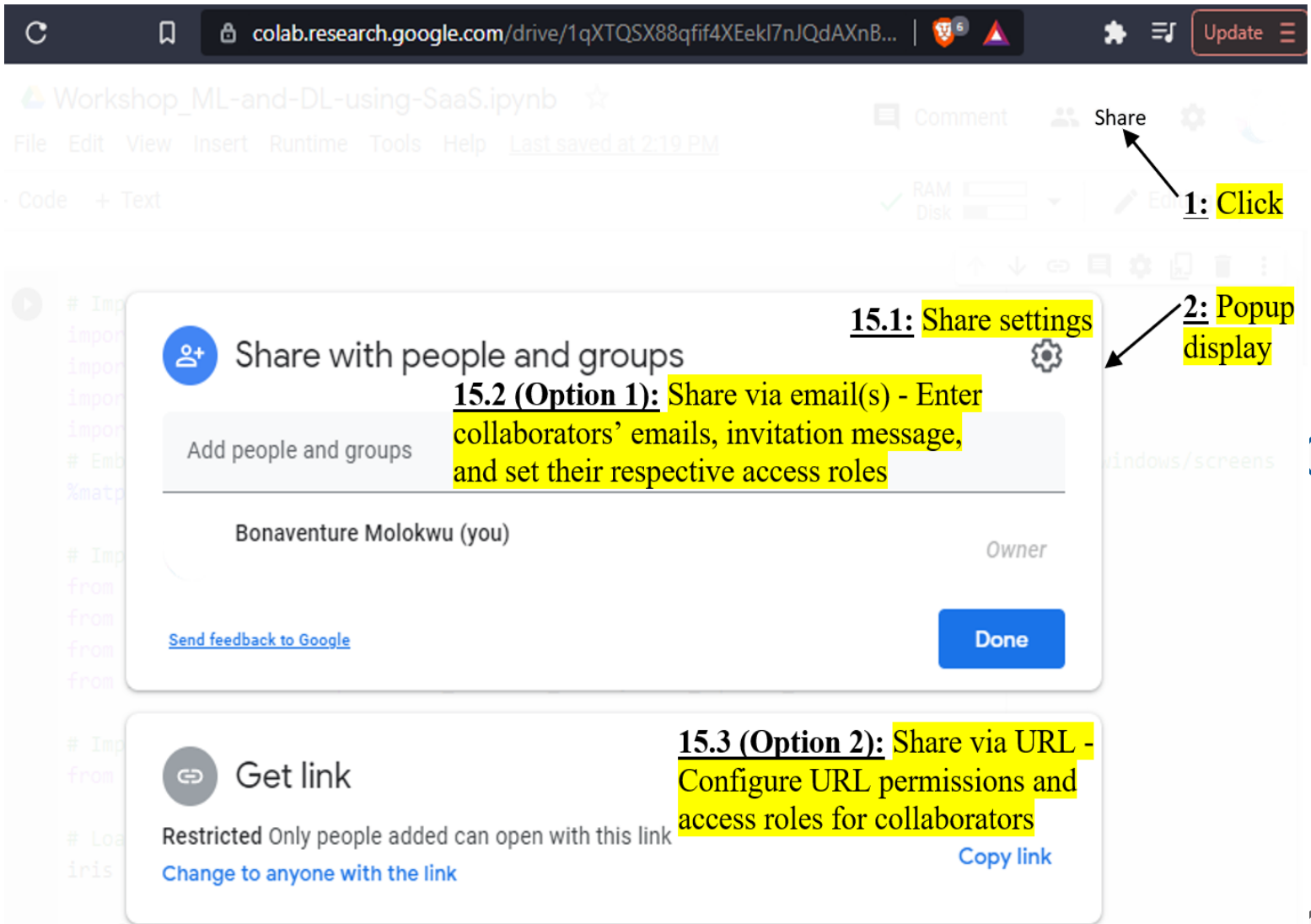
14. Save and/or Download source file:
    - 14.1 File → Save : Saves source file.
    - 14.2 File → Save a copy in Drive : Saves a copy (duplicate) of the source file to Google™ Drive.
    - 14.3 File → Save a copy as a GitHub Gist : Saves a copy (duplicate) of the source file to your GitHub™ Gist Timeline/Repository, `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.

University of Windsor

14. Save and/or Download source file:



15. Share/Teamwork on your code:



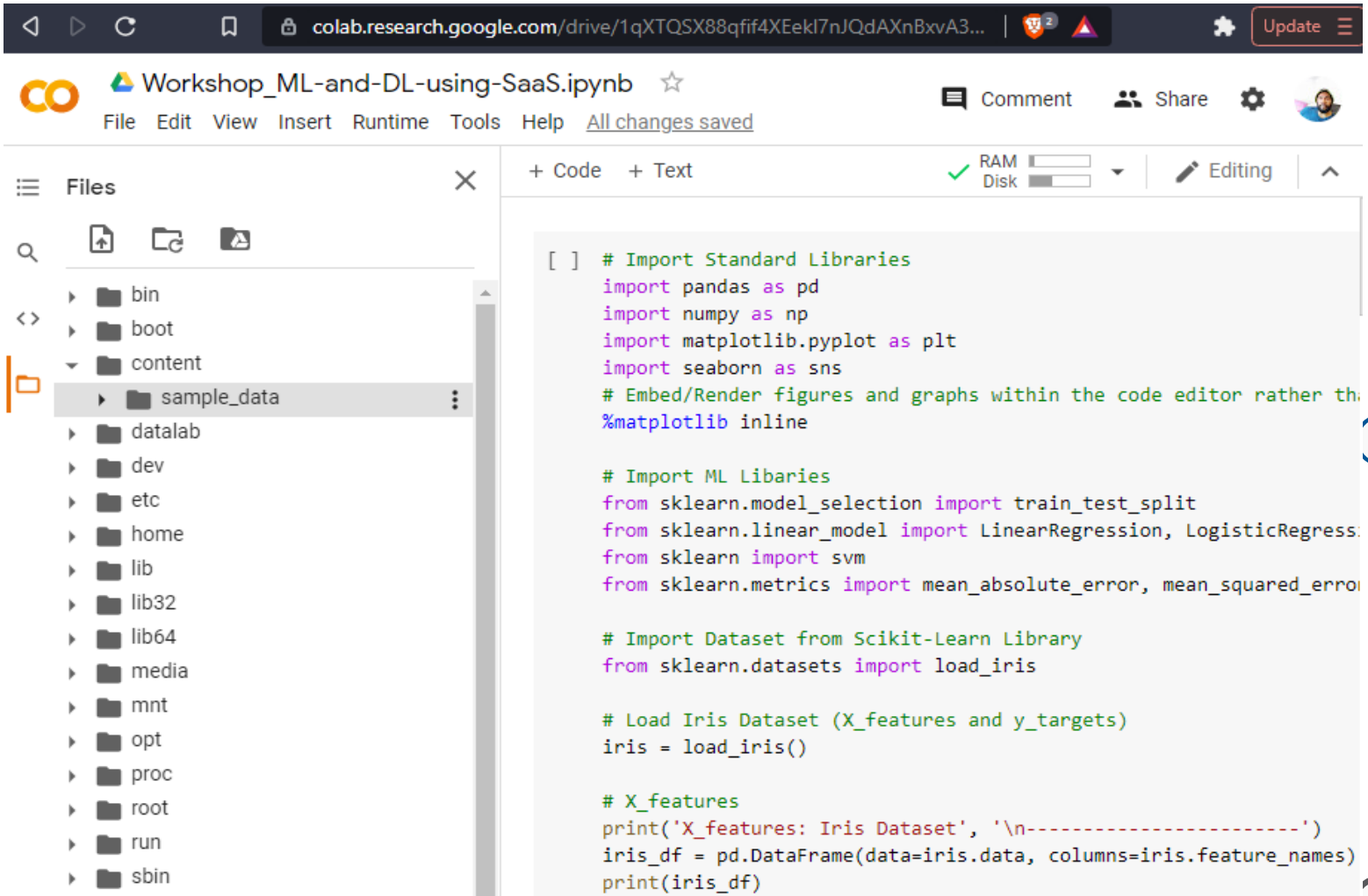
15. Share/Teamwork on your code:

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*).

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:

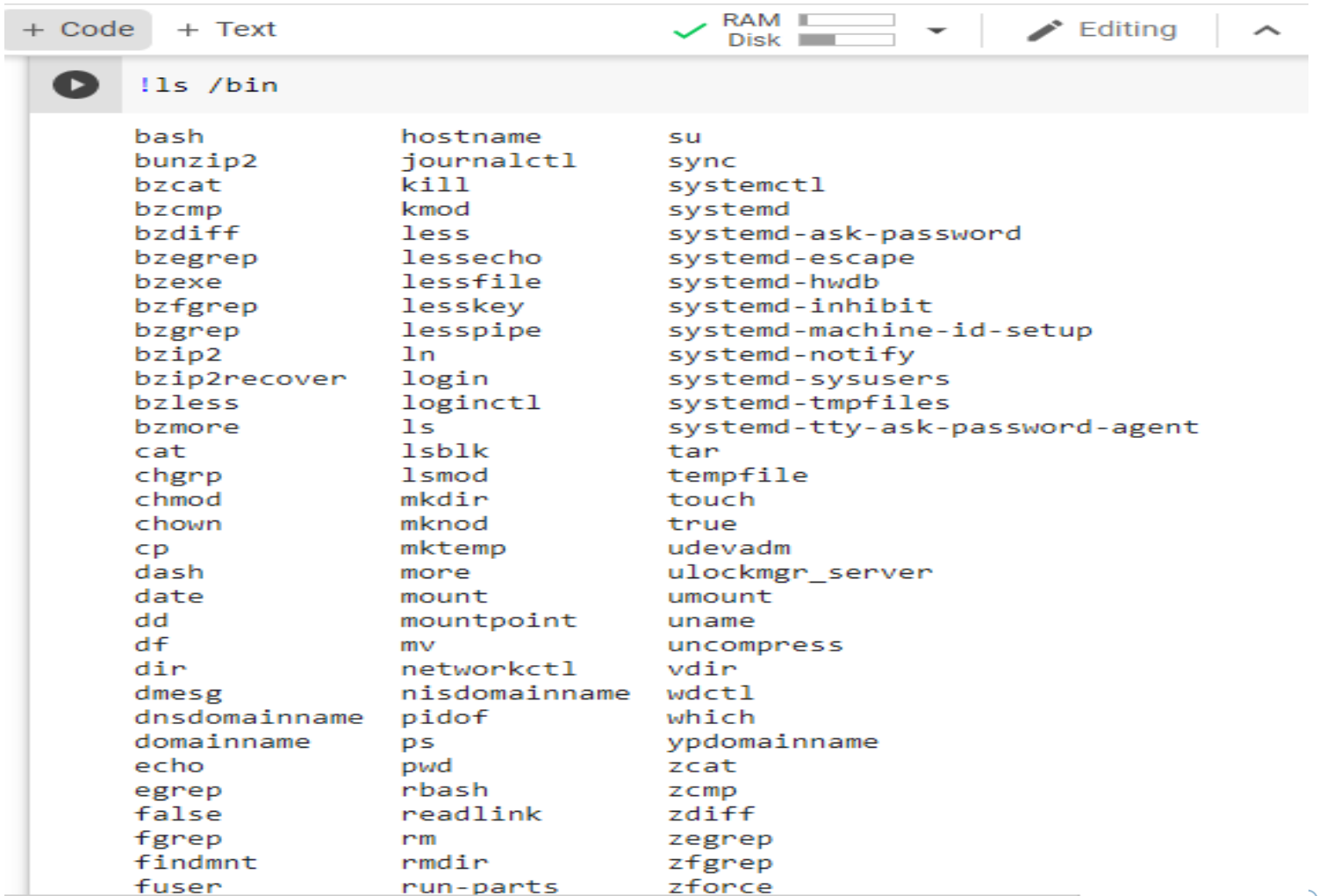


Google™ Colab

Colab/Colaboratory SaaS: Running Shell Commands

University of Windsor

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

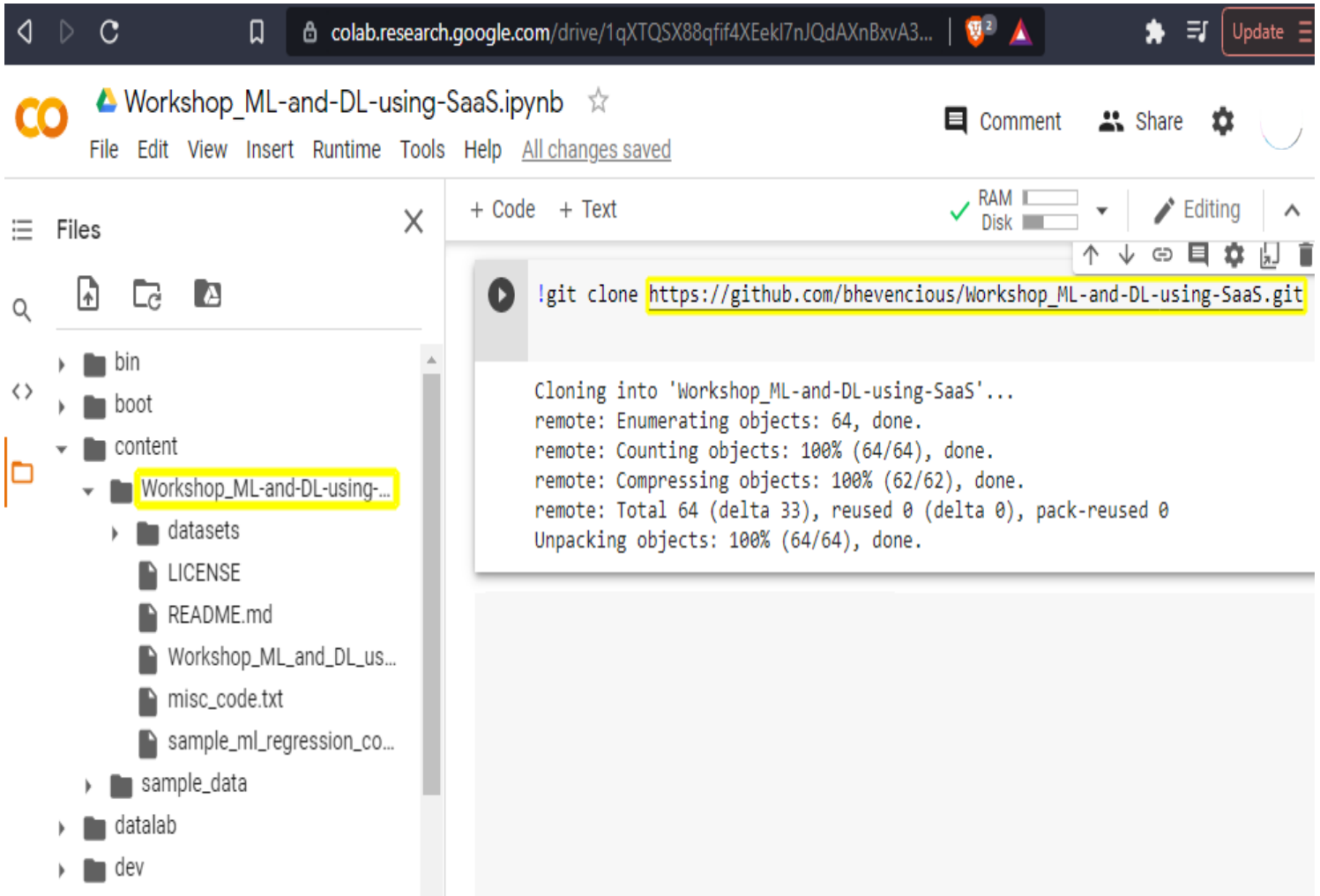


Google™ Colab

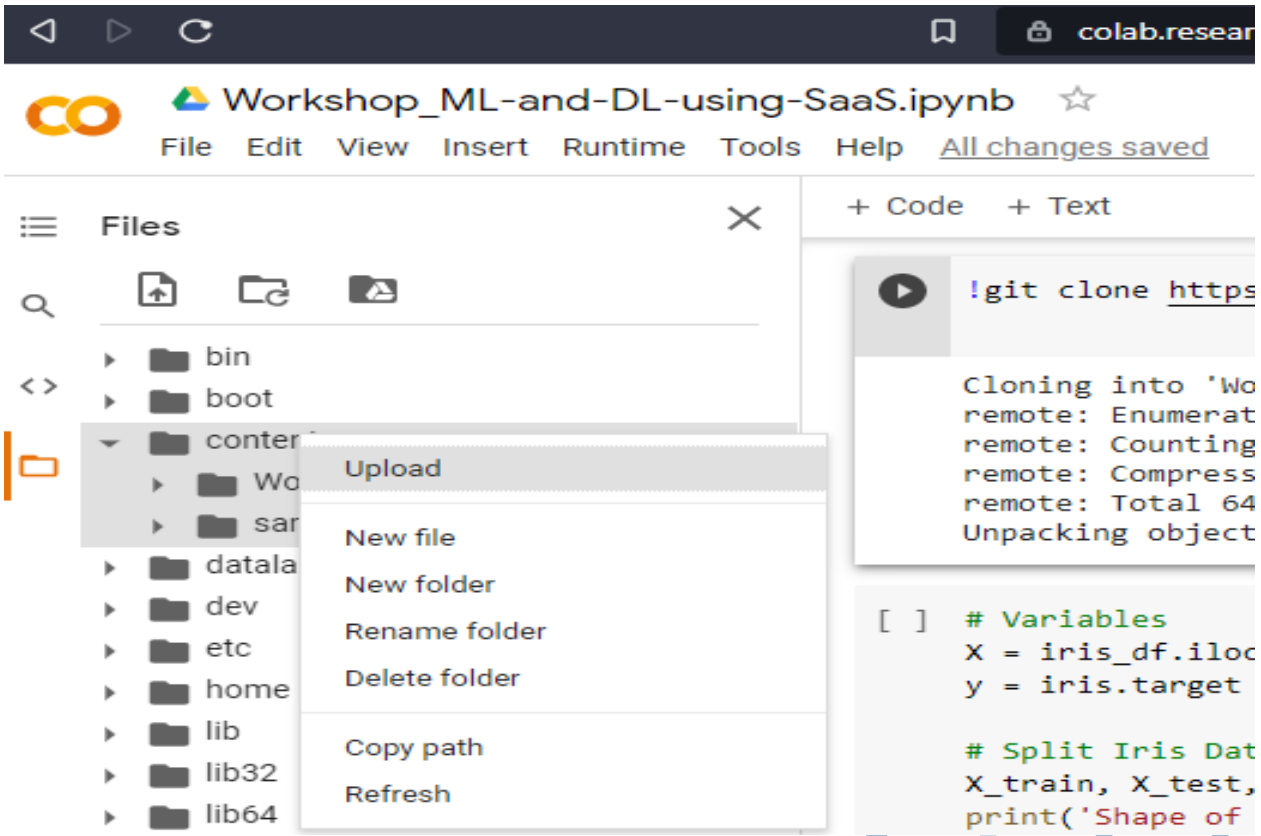
Colab/Colaboratory SaaS: Cloning a Remote Repository

University of Windsor

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



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.



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

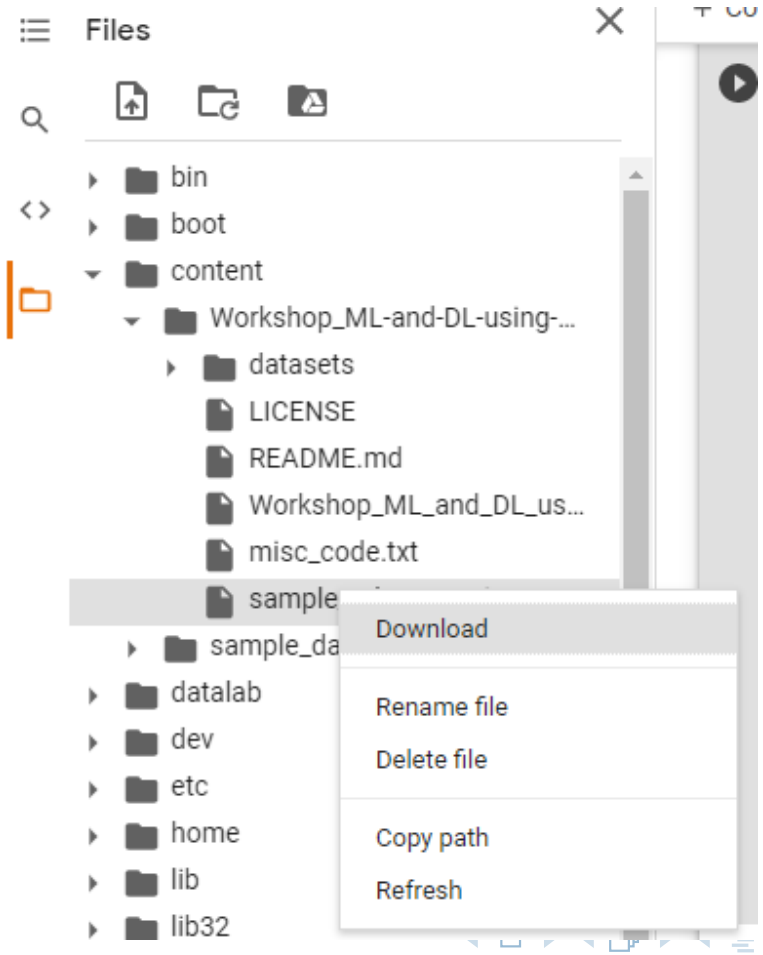
```
'''
Read a local dataset/file (in .csv format) present in your working directory
USE: pd.read_csv("<LOCAL_URL>")
'''
a = pd.read_csv("sample_data/california_housing_test.csv")
a.head()
```

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()
```

University of Windsor

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



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

23.1 Create and use free MySQL databases via:
`https://www.freesqldatabase.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`

23.3 Install/Import MySQL database query dependencies.

(a) Python package for MySQL - pymysql

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

Google™ Colab

Colab/Colaboratory SaaS: Working with Databases

University of Windsor

```
# 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)
```

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

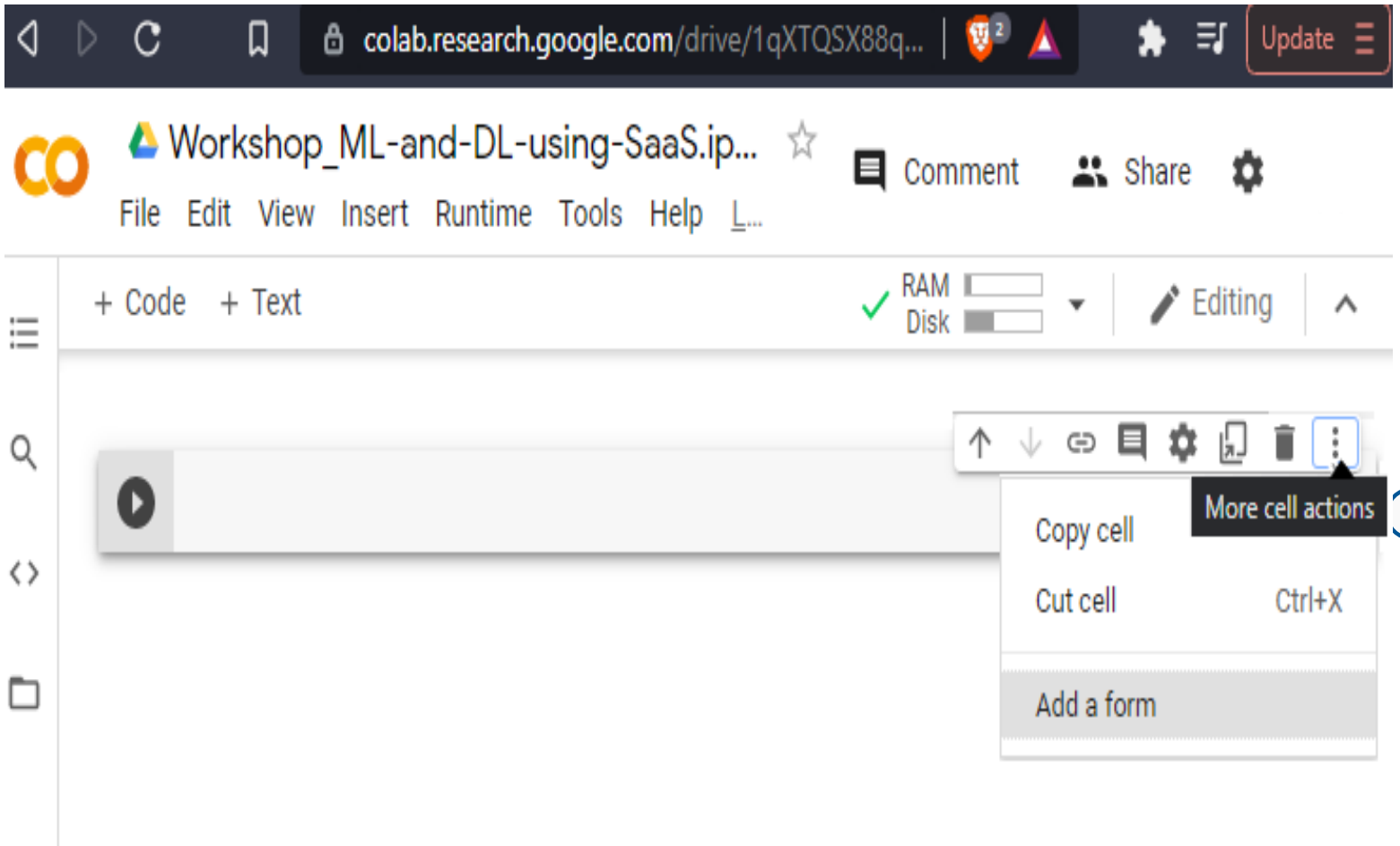
Google™ Colab

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

Google^{TM} Colab

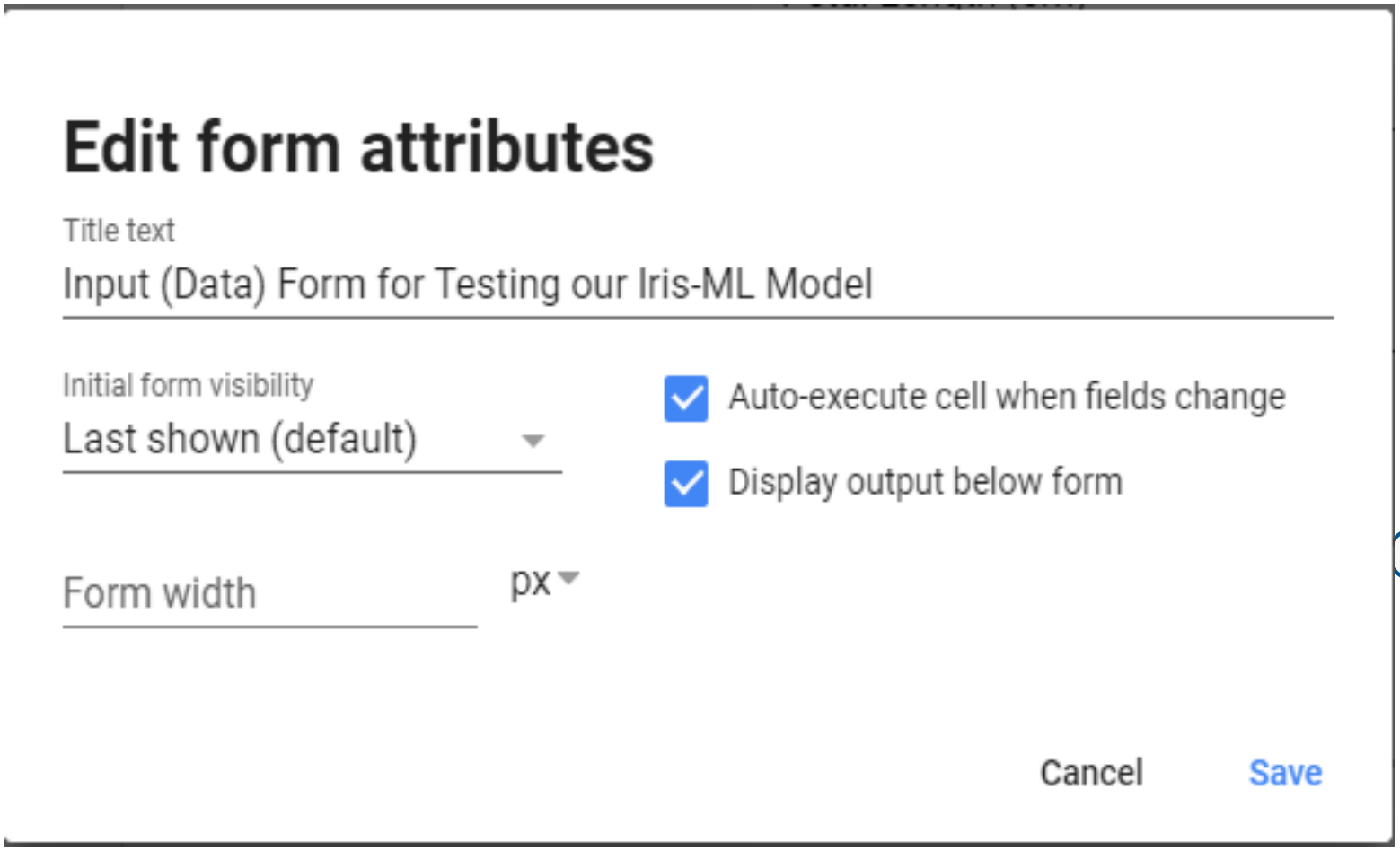
Colab/Colaboratory SaaS: CPU vs. GPU

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



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

26. Data-Form properties dialog/window:



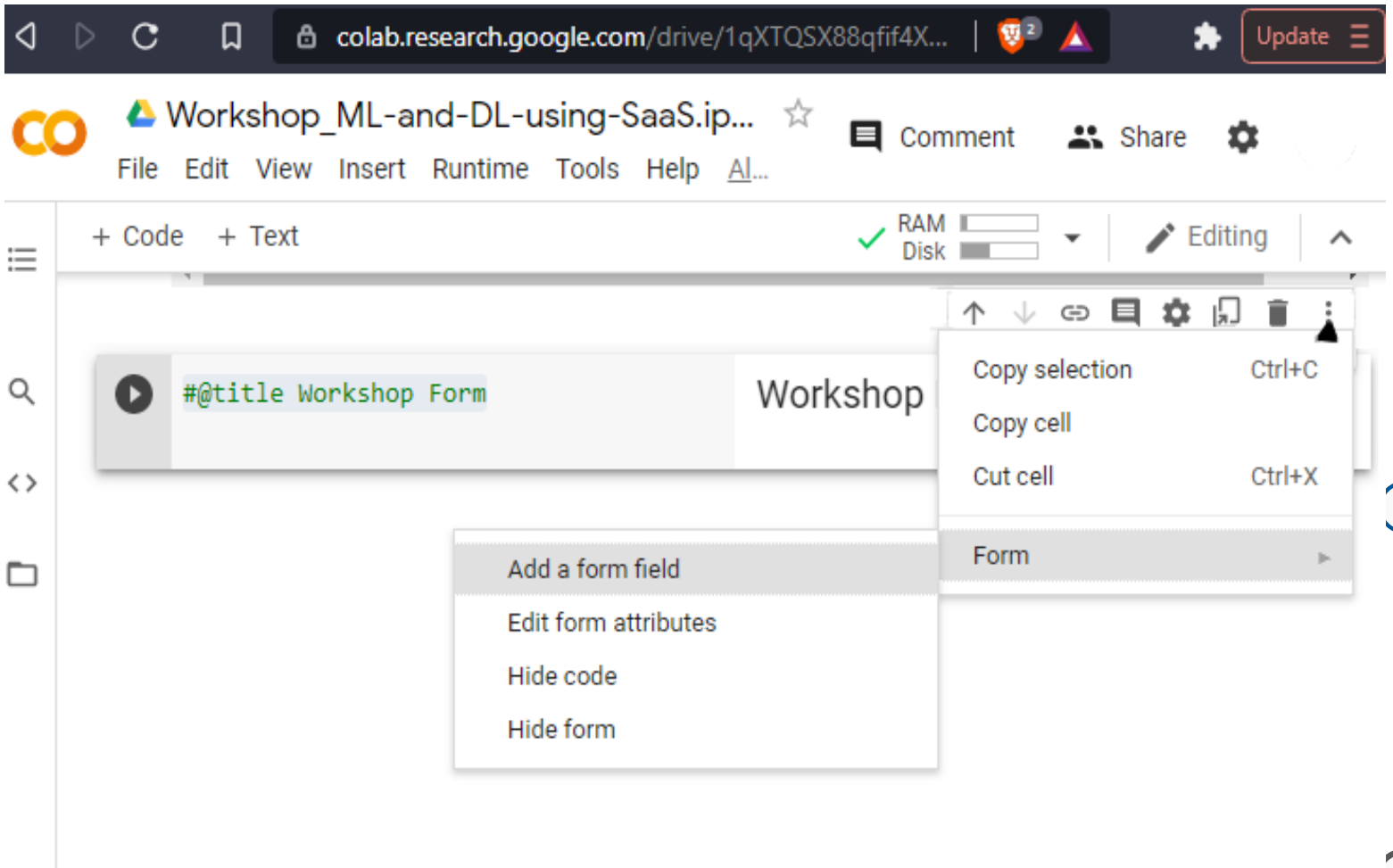
Google™ Colab

Colab/Colaboratory SaaS: Add Fields to Data Forms

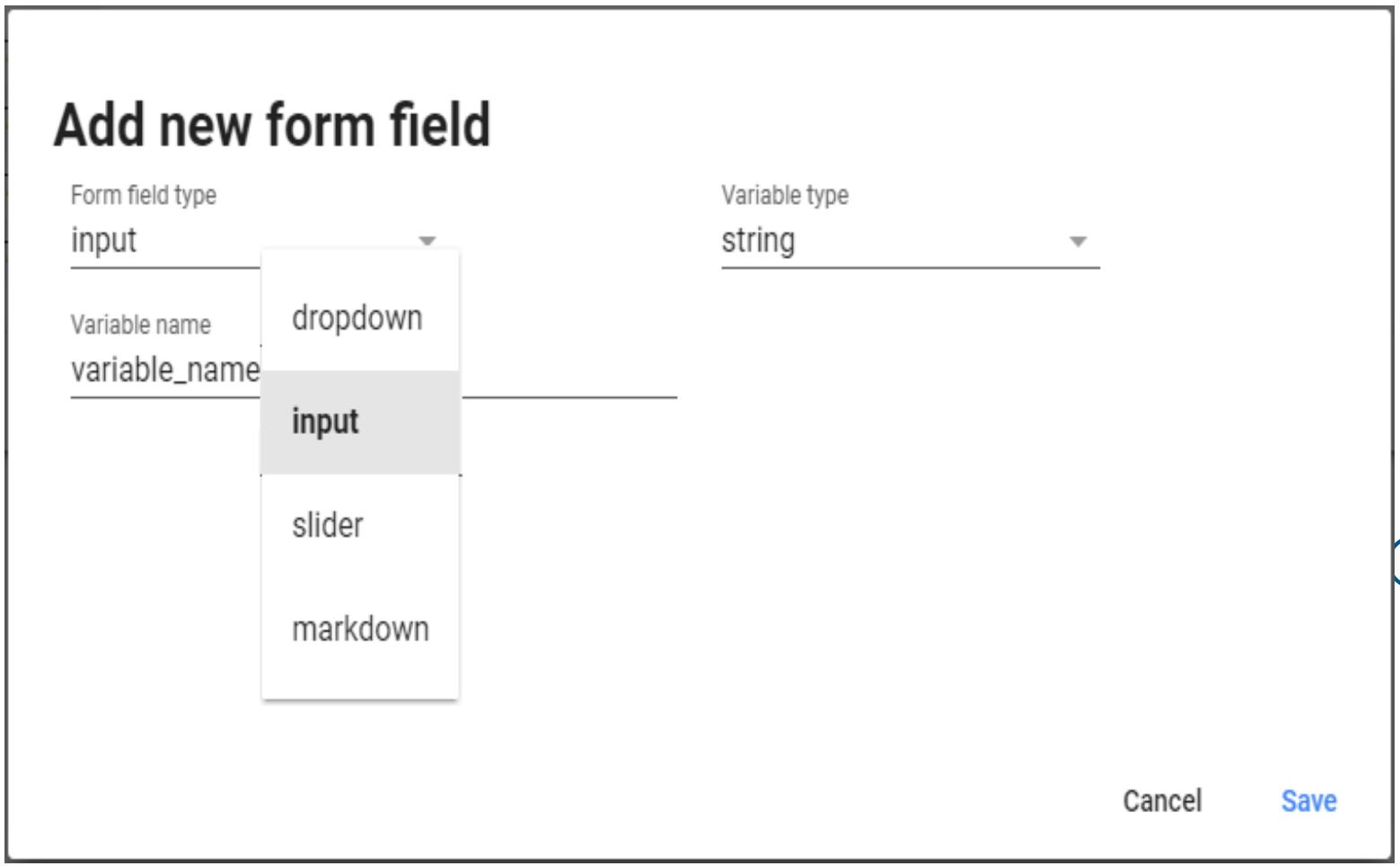
University of Windsor

27.1 Edit attributes → Form → Add a form field **OR**

27.2 Insert → Add a form field:



28. Form fields:

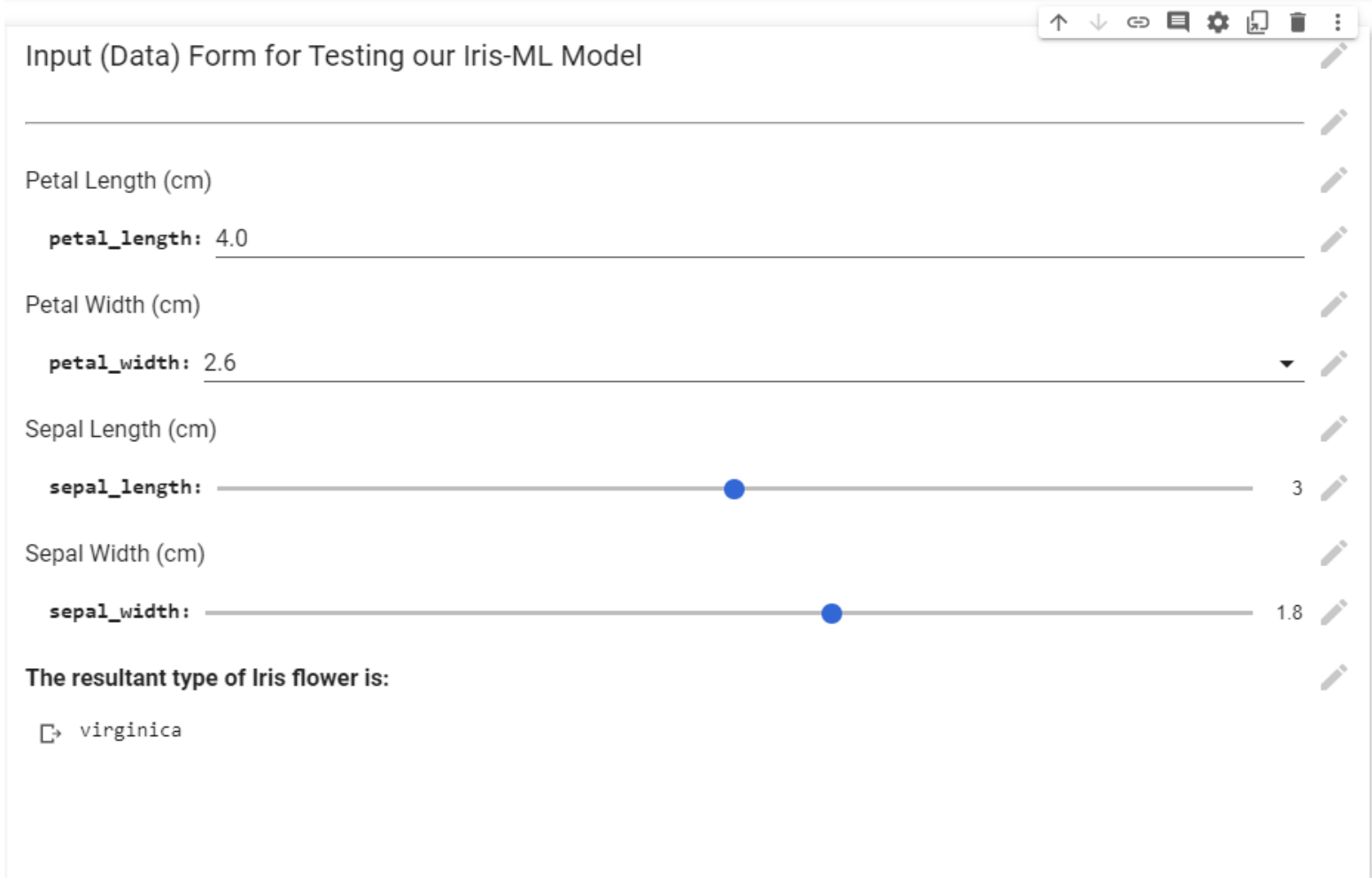


University of Windsor

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

```
#@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",

#@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)
```

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



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

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

```
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 Lengt... 0.2
Petal Width... 2.4
Sepal Leng...  1.3
Sepal Widt...  0.7
```

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

University of Windsor

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** (*!ls*) - 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

University of Windsor

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.

Basic Commands
Basic Unix-based (Linux) Commands

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.

Basic Commands
Basic Unix-based (Linux) Commands

University of Windsor

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

University of Windsor

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_alphabetic.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://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
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. McLay, 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