Artificial Intelligence for Cybersecurity Experiment 1

Configuring artificial intelligence toolkits for cybersecurity

Artificial intelligence (AI) toolkits are software packages that provide advanced algorithms, machine learning models, and other AI-based capabilities to solve problems using artificial intelligence. AI toolkits for cyber security help detect, prevent, and respond to cyber threats. Using these toolkits, we can typically use advanced analytics, such as data mining and statistical analysis, to identify patterns and anomalies in network traffic, system logs, and other types of data that may indicate a security breach.

Programming language and the development environment

Python

- a high-level, interpreted programming language
- simple syntax, code readability, and versatility
- one of the most widely used programming languages in the world
- used for a wide range of applications, including web development, scientific computing, data analysis, artificial intelligence, and more
- its large and active community of developers has created a vast array of libraries and tools that make it easy to perform complex tasks.

IPython

- an interactive command-line shell (shell: a command-line interface (CLI) that allows users to interact with an operating system (OS) by entering commands as text) for Python that provides a rich environment for interactive computing
- provides a more user-friendly and interactive environment for executing Python code
- popular tool for data science and scientific computing due to its ability to combine code execution with rich output and documentation.

Jupyter Notebook

- a web-based interactive computational environment for creating and sharing documents that contain live code, equations, visualizations, and narrative text
- based on the IPython shell
- allows users to mix code, output, images, videos, and rich text in a single document, making it an ideal tool for data science, scientific computing, and teaching
- provides a wide range of features for data visualization, including built-in support for popular data visualization libraries such as Matplotlib, Seaborn, and Plotly.

Google Colab

- Google Colab is a free, cloud-based Jupyter notebook environment provided by Google.
- It allows users to write, run, and share Jupyter notebooks with others, without the need for installing any software on their own computer.
- Colab provides free access to GPUs and TPUs, making it an attractive option for running computationally intensive tasks, such as machine learning and data analysis.
- It integrates with Google Drive, allowing users to easily save and share their notebooks with others.
- It includes pre-installed libraries and tools such as TensorFlow, PyTorch, and OpenCV, making it easy to get started with machine learning and computer vision tasks.
- Colab is accessible from any device with an internet connection, and its collaboration features allow multiple users to work on the same notebook simultaneously.
- Colab supports multiple programming languages including Python, R, and Swift.

Python (a general-purpose programming language) ➔ IPython (an enhanced interactive shell for Python) ➔ Jupyter Notebook (a web-based interface for IPython with additional features) ➔ Google Colab (a free cloud-based version of Jupyter Notebook that provides access to additional hardware supports)

How to run Google Colab

1. To run Google Colab, you need a Google account. If you don't have one, you can create one for free.
2. Go to https://colab.research.google.com and sign in with your Google credentials.
3. From the Google Colab home page, click the "New Notebook" button to create a new notebook.
4. You can also upload an existing Jupyter notebook to Colab by clicking the "Upload" button and selecting the notebook file.
5. In the notebook, you can write and run code cells, add markdown cells for text and explanations, and insert images, videos, and other rich media.
6. You can execute code cells by clicking the "Run" button or by pressing Shift + Enter. The output will be displayed below the code cell.
7. To run code on a GPU or TPU, go to the "Runtime" menu and select "Change runtime type". From there, you can select the type of hardware accelerator you want to use.
8. You can save your notebook by going to the "File" menu and selecting "Save". Your notebook will be saved to your Google Drive, and you can access it from any device with an internet connection.
9. To share your notebook with others, go to the "Share" button and enter the email addresses of the people you want to share it with. You can also make your notebook public so anyone can access it by selecting "Get shareable link" in the "Share" menu.

Running non-Python command in the Colab

Shell command

The "!" symbol is used to run shell commands. The exclamation point is used to indicate that the following command should be executed in the shell, rather than as Python code.

For example, you can use `!ls` to list the files in the current directory, or `!pip install` to install a Python package. By running shell commands in Colab, you can perform various tasks such as installing packages, downloading files, and accessing external repositories.

In this way, the "!" symbol in Google Colab provides an interface to the underlying operating system and enables you to perform tasks that are not possible with pure Python code.

Magic command

The % symbol is used to run IPython magic commands. IPython magic commands are a set of special commands that provide additional functionality in IPython (and Jupyter) notebooks. They are used to perform tasks that are specific to the IPython environment, such as timing code execution, accessing the shell, and more. Magic commands start with the % symbol and can be used in a code cell to perform a variety of tasks.

For example, `%timeit` is a magic command in IPython that measures the execution time of a single-line expression. When you run `%timeit` in a code cell in Google Colab, it will display the execution time of the code in milliseconds.

Google Colab file structure

Google Colab provides a virtual file system that is accessible from the Jupyter notebook interface. The underlying virtual machine in which Google Colab runs is a Linux distribution. The files and directories in this virtual file system can be manipulated using the Jupyter notebook interface or using shell commands.

The file system in Google Colab includes several directories, including:

- /content: The default working directory for Google Colab notebooks. This is where you will store and access your Jupyter notebooks, datasets, and other files.

You can access and manipulate the files in the virtual file system using the Jupyter notebook interface, or by using shell commands. This includes creating and deleting files, uploading and downloading files, and managing the file system as you would on your local machine.

Installing additional packages on the Colab

You can install additional packages in Google Colab using shell commands or IPython magic commands.

To install a package using shell commands, you can use the `!pip install` command. For example, to install the numpy package, you would run the following code:

```
!pip install numpy
```

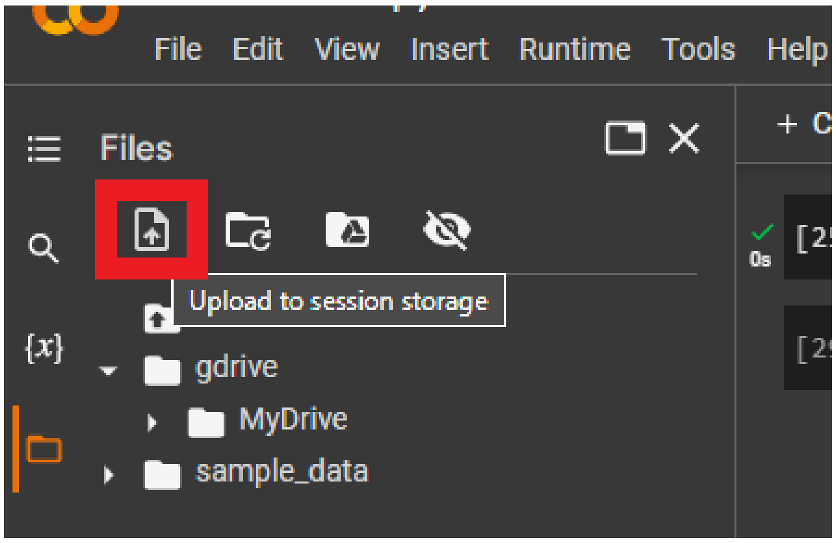
You will need to run the command in a code cell. Once the installation is complete, you can import the package and start using it in your code.

Working with Google Drive and other external repositories within the Colab

Google Colab allows you to work with files stored in your Google Drive as well as other external repositories. Here are some ways to work with Google Drive and other external repositories in Colab:

Upload to session storage

You can upload files from your local machine:



Mounting Google Drive

You can mount your Google Drive to Colab and access its files in the same way as if they were stored on your local machine. To mount your Google Drive, you need to run the following code in a code cell in Colab:

```
from google.colab import drive
drive.mount('/content/gdrive')
```

After running the code, you will be prompted to authorize Colab to access your Google Drive. Once you have authorized access, your Google Drive will be mounted at the `/content/gdrive` path.

Accessing files in Google Drive

To access files in your Google Drive, you can use the standard file operations such as `open`, `read`, and `write`. For example:

```
with open('/content/gdrive/MyDrive/file.txt', 'r') as f:
  data = f.read()
```

Accessing external repositories

You can also access external repositories such as GitHub, GitLab, or Bitbucket by cloning the repository using `git clone`. For example:

```
!git clone https://github.com/mrh-rakib/mrh-rakib.github.io
```

Downloading files from URLs

You can also download files from URLs using the `wget` or `curl` command. For example:

```
!wget https://www.example.com/file.zip
```

Note that all these operations can be performed in code cells, so you can automate the process of downloading and accessing files in your Google Drive or external repositories

Practice tasks

1. Configure and get yourself familiarised with your own AI development environment for cyber security.