NewburyAITheInstallations

July 15, 2019

Compiled by: Rob Wallace, updated 15 July 2019

0.1 How to Install a Machine Learning Set Up on Your Computer

0.1.1 Installing the Initial Packages

Python is one of the most popular programming languages to use for Machine Learning. The recommended way to install Python and other necessary Python-related ML packages is to install Anaconda, which includes a Python distribution, and through its Conda package and environment management tool, provides access to many other installable packages.

"Anaconda is focused toward data-science and machine learning and scientific computing. It installs cleanly on your system in a single directory so it doesn't make a mess in your systems application and library directories. It is also performance optimised for important numerical packages like numpy, scipy etc.." Dr Donald Kinghorn, https:pugetsystems.com

0.1.2 Downloading Anaconda:

- The Anaconda downloads page is https://www.anaconda.com/distribution
- Click on the Windows, Linux or macOS Installer
- Get the 64-bit, and the **newer** version of Python, or the 32-bit, depending on your machine

0.1.3 Installing it:

- Double-click on the downloaded file to run it (Windows and macOS) For Linux and macOS, use Bash in a Terminal session.
- Answer to acceptance of the licence agreement
- Select "Just Me" for Install Type
- Accept the Install location default
- Tick "Register Anaconda as my default Python 3.7"

0.1.4 Checking the installation:

- In your Start Menu(Windows), find "Anaconda Prompt", or start a Terminal.
- In Terminal or Anaconda prompt, type: `python`
- The response should give the version and will activate a python coding prompt »>.
- To exit the python editor, hold down the ctrl key and type `d` (Windows)

0.1.5 Checking conda's channels

- Now check whether the channel (downloading sources) "conda-forge" is in conda's channel configuration. Type `conda info`
- If "conda-forge" is included in the channels list, all is OK. Otherwise, issue the following command:
- `conda config --add channels conda-forge`. Then check using `conda info` again.
- Then to exit terminal, type `exit`

0.1.6 Updating the Anaconda Packages:

- In a Terminal or Anaconda prompt, use conda, a package and environmental management tool to perform the following updates:

```
conda update conda
conda update anaconda
conda update python
conda update --all
```

0.1.7 Installing Some Machine Learning Packages

You will need several Python packages to use in your Machine Learning projects. You can first install them in your (base) environment, which will download them in order to use in your various environments later, as needed. It is better to use conda as your package and environment manager, rather than using "pip", a python installation manager.

- Some popular packages can be initially installed using the Anaconda or Terminal prompt by entering the following code line:

```
conda install pandas numpy matplotlib seaborn scikit-learn tensorflow keras
lightgbm hyperopt umap-learn
```

0.1.8 Creating a Separate Environment in which to run ML projects

It is general good practice to keep separate environments for projects especially when they have special package dependencies. Think of it as a separate "name-space" for your project.

- From the Terminal or prompt create an environment by using:

```
conda create --name myprojectname
```
(Beware of using spaces for names)
Now close this terminal and reopen another one before activating it:

- Look at your environment list by coding:

```
conda info --envs
```

- The * denotes the current active environment. Then code:

```
conda activate myprojectname
```
You will now see the prompt changes to reference the name of your environment.

0.1.9 Installing the Support Packages for your Projects

- From your environment prompt, enter the same code used to install the packages in the (base) environment, but include jupyter, which is not intalled by default in the environment you created:

```
conda install jupyter pandas numpy matplotlib seaborn scikit-learn tensorflow
keras lightgbm hyperopt umap-learn
```

You will see that the installation will not repeat the downloads, but merely "install" them from conda (base) into your particular environment.

- You can confirm the package installation in your environment using:

```
conda list
```

- It is also prudent to update the packages in your environment using:

```
conda update --all
```

Because the packages are now installed in your environment, they can be imported as needed using Python code. Otherwise when you "Run" your project code, the Python compiler will report that it is not be able to access the packages.

A useful resource is the **conda cheatsheet**, which shows how to perform various environment operations, including creating, activating, deactivating and removing environments, plus installing, listing and uninstalling packages. You can obtain a copy at: https://docs.conda.io/projects/conda/en/latest/user-guide/cheatsheet.html

0.1.10 Starting the Jupyter Notebook IDE

Starting a notebook in the Jupyter IDE is done by opening a Terminal, or Anaconda prompt. Ensure the directory signature shown is a top directory, not an empty branch, by typing `cd` plus a "space" after "cd", then "enter". Otherwise, navigate to the directory where you want to store your projects, but that can be done from the opening **dashboard** top directory structure anyway.

- From the chosen place in your directory hierarchy, type the following:

```
jupyter notebook
```

- After preliminary information in the terminal, the **dashboard** opens in your browser. From this interactive directory structure, navigate to your project directory and click on "New" on the top menu to open a new notebook. "New" will present you with a list of choices. Click on "Python 3" (the kernel to be started) for now.

As will be expected, there are many functions and operations possible in a Jupyter Notebook. Its editor structure is based on rows of cells, each of which can contain either python "code" or notebook "markdown", images, graphs, or even video. This makes it a powerful comunication medium.

- After you code and "Run" cells, a new empty cell is added below your last cell. After some work has been entered before running, it is advisable to save your work progressively by choosing "Save and Checkpoint" from the file dropdown menu.

- To exit a notebook, choose “Close and Halt” in the “File” menu.
- Then, close the **dashboard** like a webpage,
- then type `ctrl-c` to shut down the internal server, then type `exit` to shut down the terminal.

It is recommended to look at the following Jupyter Notebook user notes to help you: https://jupyter-notebook.readthedocs.io/en/stable/notebook.html

There is also a user manual in development, but substantially completed at: https://buildmedia.readthedocs.org/media/pdf/jupyter-notebook/latest/jupyter-notebook.pdf