Lab and Q & A (B1M2L4T1)

Contents

- Setting up the development environment (60Min)
- Hands-on coding session with instructor guidance (60Min)
- Weekly wrap-up and Q&A (30Min)

Weekend Lab Work:

- A project on supervised learning using a provided dataset.

Setting up the Development Environment

- Setting up a development environment for machine learning (ML) involves installing the necessary software and configuring your system to effectively perform ML tasks.

- The specific steps may vary depending on your operating system and preferences, but the general approach involves the following:

1. **Choose an Operating System:**

Linux: Linux is a popular choice for ML due to its stability, performance, and extensive support for ML tools.

Windows: Windows is also a viable option, with various ML tools available, including the Anaconda distribution.

macOS: macOS offers a user-friendly environment and can be used for ML development, particularly with Python-based tools.

Setting up the Development Environment

2. Install Python:

Python is the primary programming language for ML, providing numerous libraries and frameworks.

Install the latest version of Python (preferably Python 3) from the official website.

3. Install a Virtual Environment Manager:

- Virtual environment managers like Anaconda or Pyenv help isolate ML projects and their dependencies from other applications.
- Consider Anaconda, a popular distribution that includes Python, virtual environment management, and various ML libraries.

Setting up the Development Environment

4. Create a Virtual Environment:

Use the virtual environment manager's commands to create and activate a virtual environment for your project.

5. Install Essential ML Libraries:

Install core ML librarieslike TensorFlow, scikit-learn, pandas, and NumPy using pip or the Anaconda Navigator.

Additional libraries may be needed based on your specific ML tasks, such as Matplotlib for data visualization.

Setting up the Development Environment

6. Set Up Code Editor or IDE:

Choose a code editor or IDE (Integrated Development Environment) for comfortable coding and project management.

Popular options include Visual Studio Code, PyCharm, and Jupyter Notebook, each with unique features and extensions.

7. Install GPU Drivers (Optional):

If you have a dedicated GPU, install the appropriate drivers for optimal performance during GPU-intensive ML tasks.

GPU drivers enable access to the GPU's parallel processing capabilities, significantly accelerating training and computations.

Setting up the Development Environment

9. Test and Verify:

Ensure that the installed libraries and tools are working correctly by running simple tests or tutorials.

10. Alternative is COLAB: (Google Cloud)

Google Colab, or Colaboratory, is a hosted Jupyter notebook service that allows you to run Python code in the cloud.

It is a free service that provides access to powerful computing resources, including GPUs and TPUs.

COLAB

To use Colab, you first need to create a Google (gmail) account.

Once you have an account, you can go to the Colab website and create a new notebook.

Free and easy to use: Colab is a free service that is easy to get started with.

Access to powerful computing resources: Colab provides access to powerful computing resources, including GPUs and TPUs.

Collaboration: Colab makes it easy to collaborate with others by sharing your notebooks.

If you are new to machine learning, Colab is a great option for getting started.

Using IRIS Dataset

here are pictures of the three flowers species:

Iris Versicolor | Iris Setosa | Iris Virginica

IRIS Dataset for Supervised Learning

```
subset(iris, Species == "setosa")[1:5,]
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
```

```
subset(iris, Species == "versicolor")[1:5,]
```

```
##    Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 51          7.0         3.2          4.7         1.4 versicolor
## 52          6.4         3.2          4.5         1.5 versicolor
## 53          6.9         3.1          4.9         1.5 versicolor
## 54          5.5         2.3          4.0         1.3 versicolor
## 55          6.5         2.8          4.6         1.5 versicolor
```

```
subset(iris, Species == "virginica")[1:5,]
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width   Species
## 101          6.3         3.3          6.0         2.5 virginica
## 102          5.8         2.7          5.1         1.9 virginica
## 103          7.1         3.0          5.9         2.1 virginica
## 104          6.3         2.9          5.6         1.8 virginica
## 105          6.5         3.0          5.8         2.2 virginica
```

Project on Supervised Learning

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import accuracy_score
from sklearn.datasets import load_iris
import matplotlib.pyplot as plt

# Load the Iris flower dataset
#iris = pd.read_csv('iris.csv')
#iris=load_iris()
iris = pd.read_csv('iris_dataset.csv', delimiter=',');
#print(iris)
# Preprocess the data
X = iris[['sepal_length', 'sepal_width', 'petal_length', 'petal_width']].values
y = iris['variety'].values
#X = iris['data']
print(X[0:5])
#y = iris['target']
print(y[0:5])
```

Project on Supervised Learning

```
# Split the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
# Train a logistic regression model
model = LogisticRegression()
model.fit(X_train, y_train)
# Make predictions on the test set
y_pred = model.predict(X_test)

#print(y_test[0:50])
#print(y_pred[0:50])

# Evaluate the model's performance
accuracy = accuracy_score(y_test, y_pred)
print("Accuracy:", accuracy)
plt.scatter(y_test, y_pred, color='black')
plt.plot(y_test, y_pred, color='red', linewidth=1)
plt.xlabel('Actual')
plt.ylabel('Predicted')
```

Project output

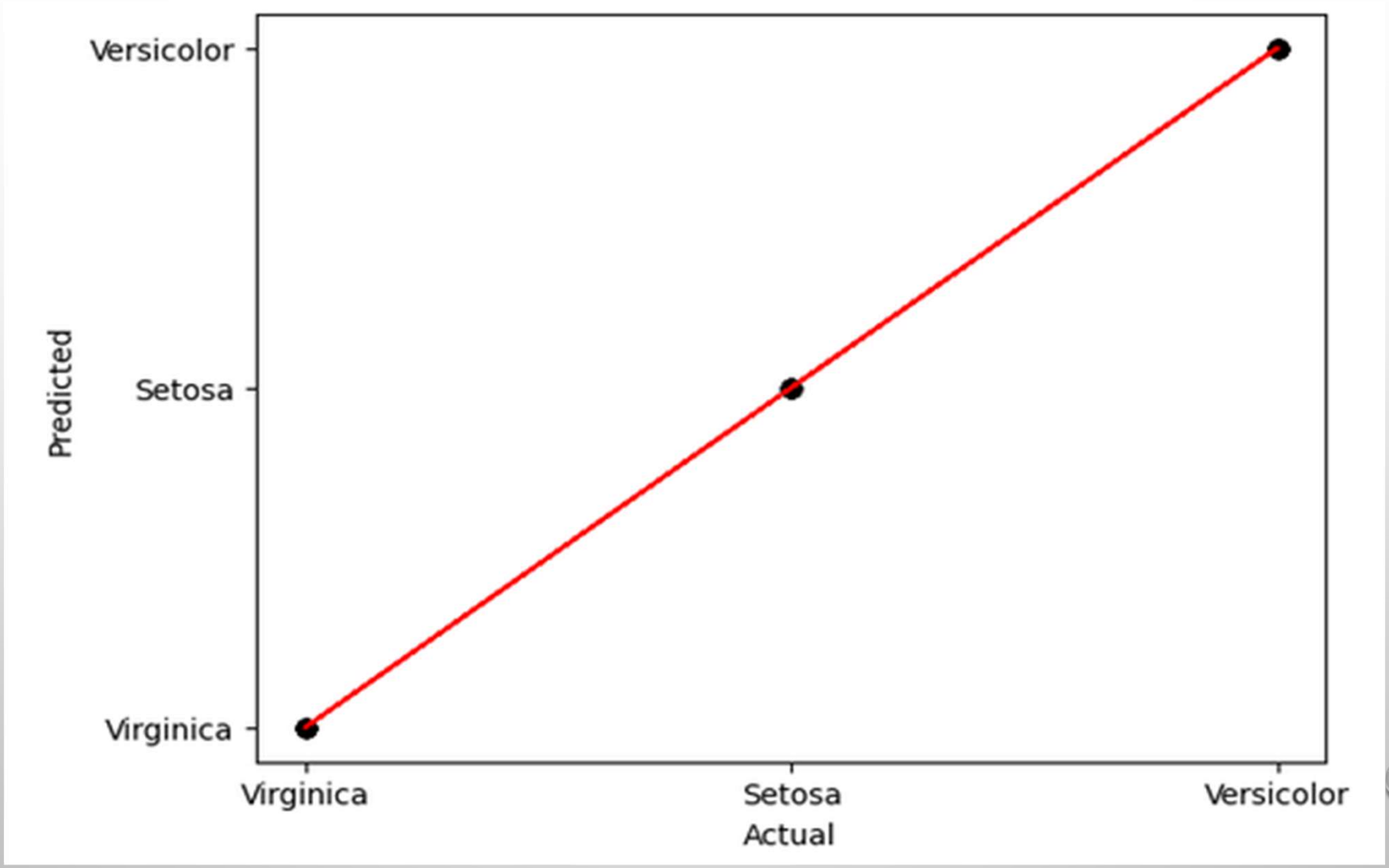
supervised_Learning.ipynb

File Edit View Insert Runtime Tools Help All changes saved

\+ Code + Text

```
[[5.1 3.5 1.4 0.2]
 [4.9 3.  1.4 0.2]
 [4.7 3.2 1.3 0.2]
 [4.6 3.1 1.5 0.2]
 [5.  3.6 1.4 0.2]]
[0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 2 2 2 2 2 2 2 2 2 2 2
 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2 2
 2 2]
[1 0 1 0 1 1 2 2 1 1 0 0 1 0 1 0 2 0 1 1 0 0 1 1 0 1 1 1 2 0]
[1 0 1 0 1 1 2 2 1 1 0 0 1 0 1 0 2 0 1 2 0 0 1 1 0 1 1 1 2 0]
Accuracy: 0.9666666666666667
```

Project output



Weekly wrap-up and Q&A

Day-1: Q&A

1. AI Definition.
2. What is “Turing Test”?
3. What is Domain Knowledge?
4. Important topics (applied areas) of AI?
5. Future of AI?
6. Real-world case studies of AI?
7. Important Tools of AI?

Weekly wrap-up and Q&A

Day-2: Q&A

1. Definition of Machine Learning (ML).
2. Ex. of Tasks that can be solved by ML?
3. Difference between Supervised and unsupervised Learning techniques?
4. Types of ML techniques?
5. Over-fitting and Under-fitting Definitions?
6. Clustering?

Weekly wrap-up and Q&A

DAY-2: Q&A

7. Definition of Machine Learning (ML).

8. List of Classification Techniques.

9. Decision-Trees.

10. Evaluation of classification techniques?

11. Difference between Training/Testing/Validation sets.

12. SVM and KNN Techniques.

13. Ensemble methods: Bagging versus Boosting.

Weekly wrap-up and Q&A

DAY-3: Q&A

1. Unsupervised Learning Algorithms.

2. K-Means Algorithms?

3. KNN Algorithms

4. What is PCA?

5. Reinforcement Learning?

Weekly wrap-up and Q&A

DAY-4: Q&A

1. Applying ML to Real-world Problems?

2. **Project:** Customer Segmentation using K-Means Clustering?

Weekend Lab Work

1. Anaconda Installation?

2. **Project:**

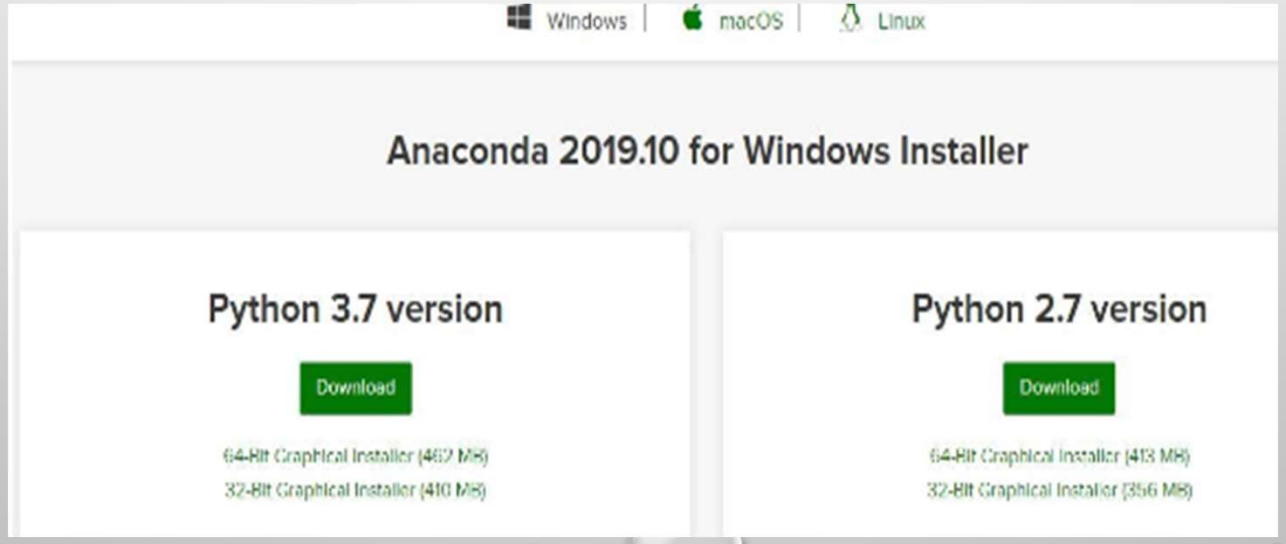
Build Your First Machine Learning Project [Full Beginner Walkthrough]

Olympics Medals Prediction Problem

https://www.youtube.com/watch?v=Hr06nSA-qww

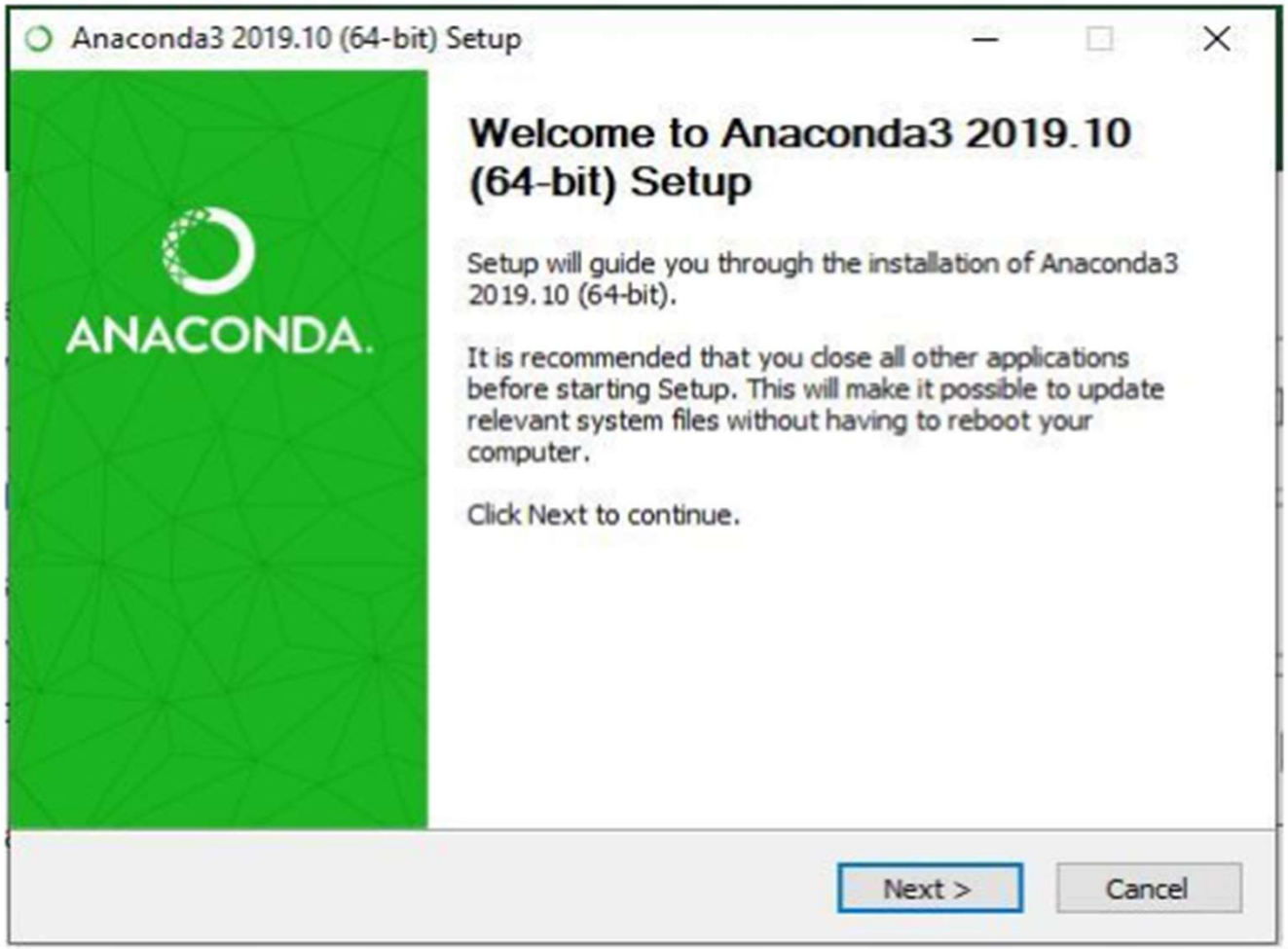
Anaconda Installation for Windows

- Anaconda is an open-source software that contains Jupyter, spyder, etc that are used for large data processing, data analytics, heavy scientific computing.
- Anaconda works for R and python programming language.
- Head over to anaconda.com and install the latest version of Anaconda.
- Make sure to download the “Python 3.7 Version” for the appropriate architecture.



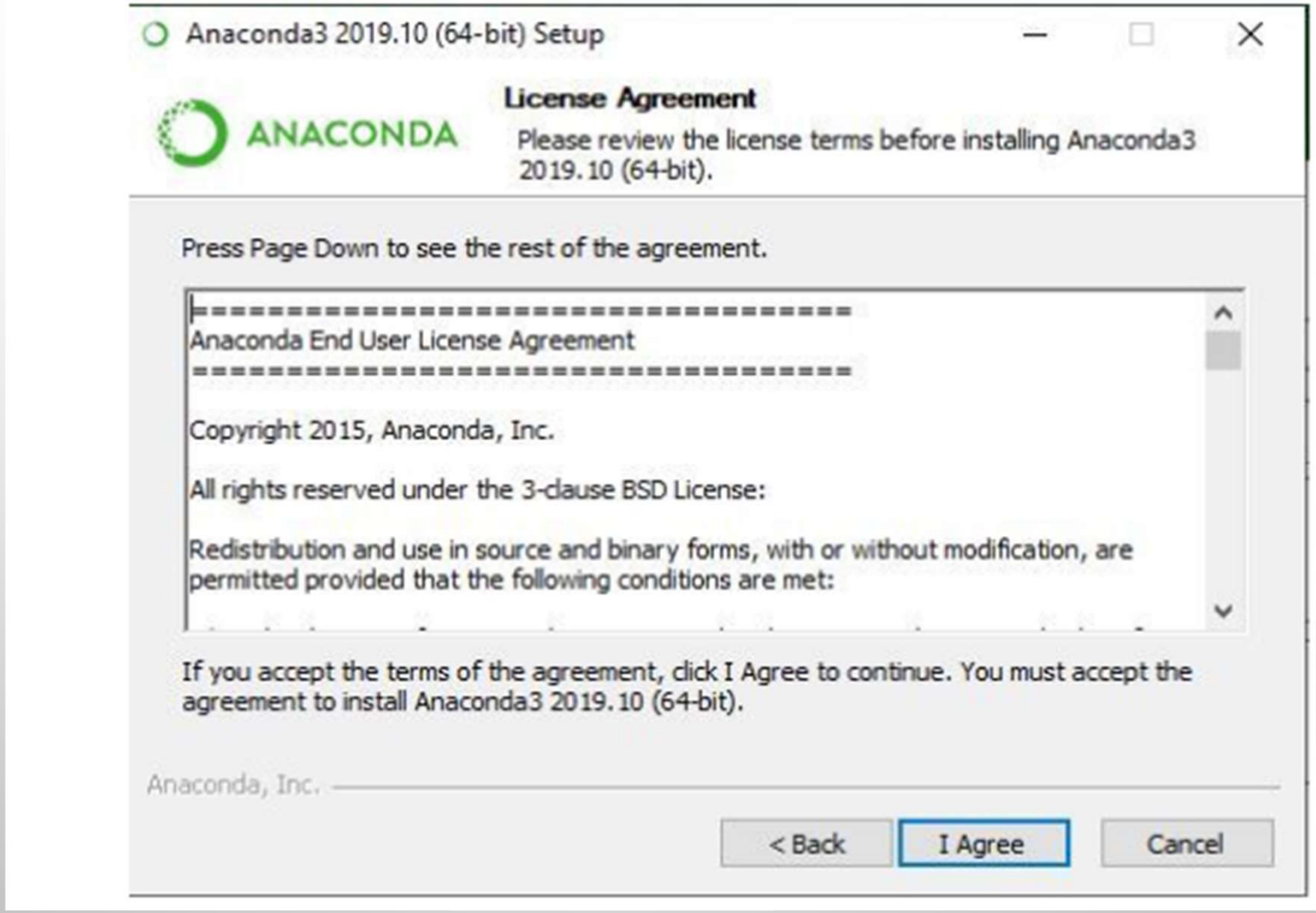
Anaconda Installation for Windows

Getting Started:



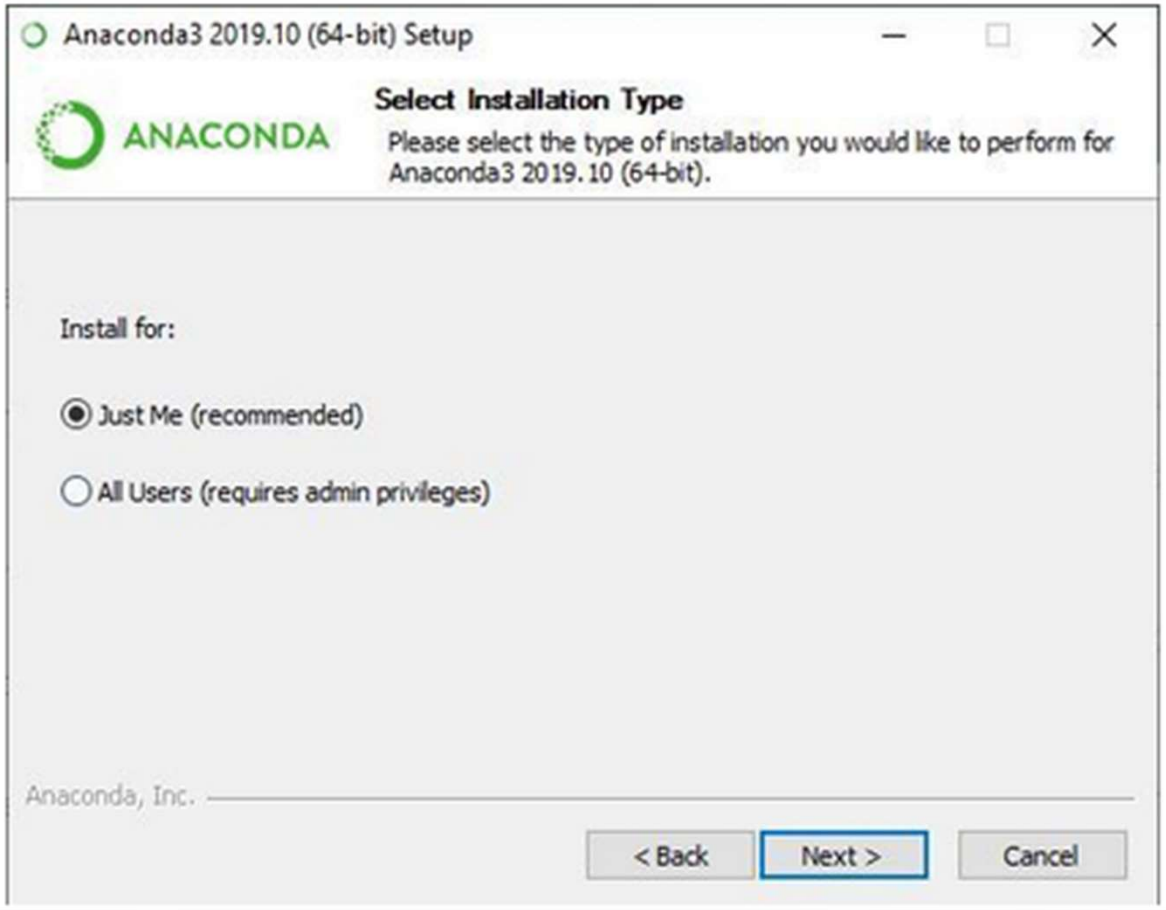
Anaconda Installation for Windows

Getting through the License Agreement:



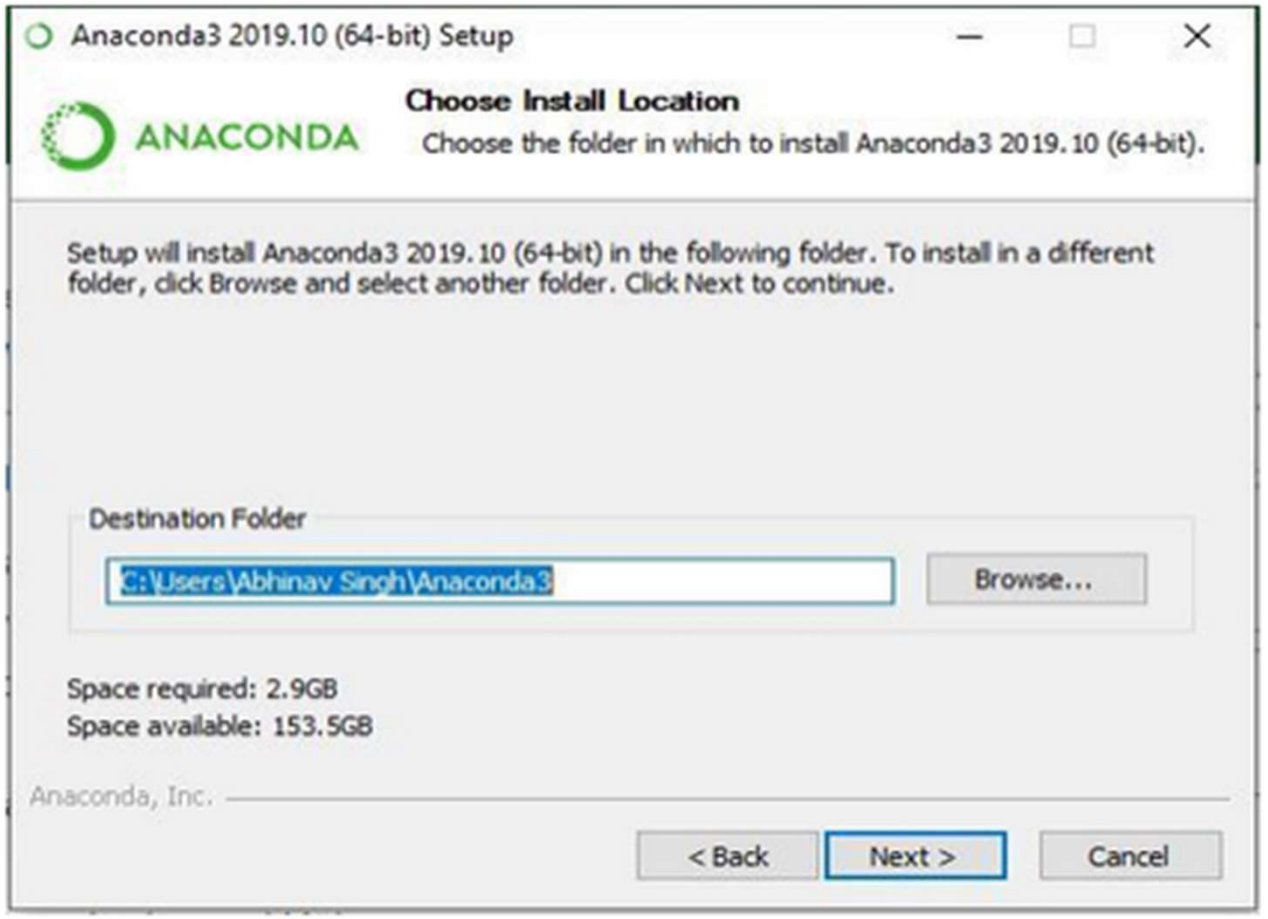
Anaconda Installation for Windows

Select Installation Type: Select **Just Me** if you want the software to be used by a single User



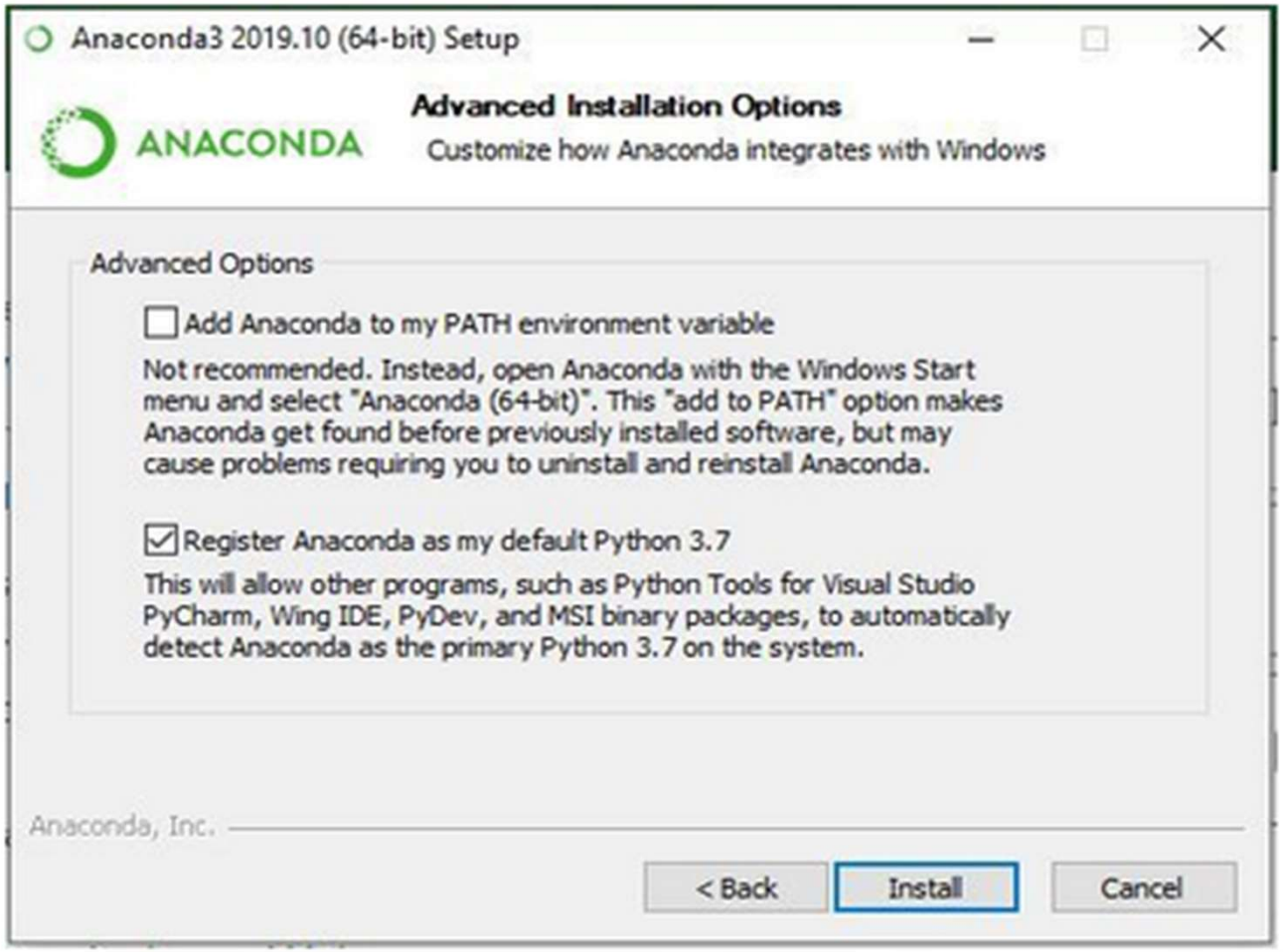
Anaconda Installation for Windows

Choose Installation Location:



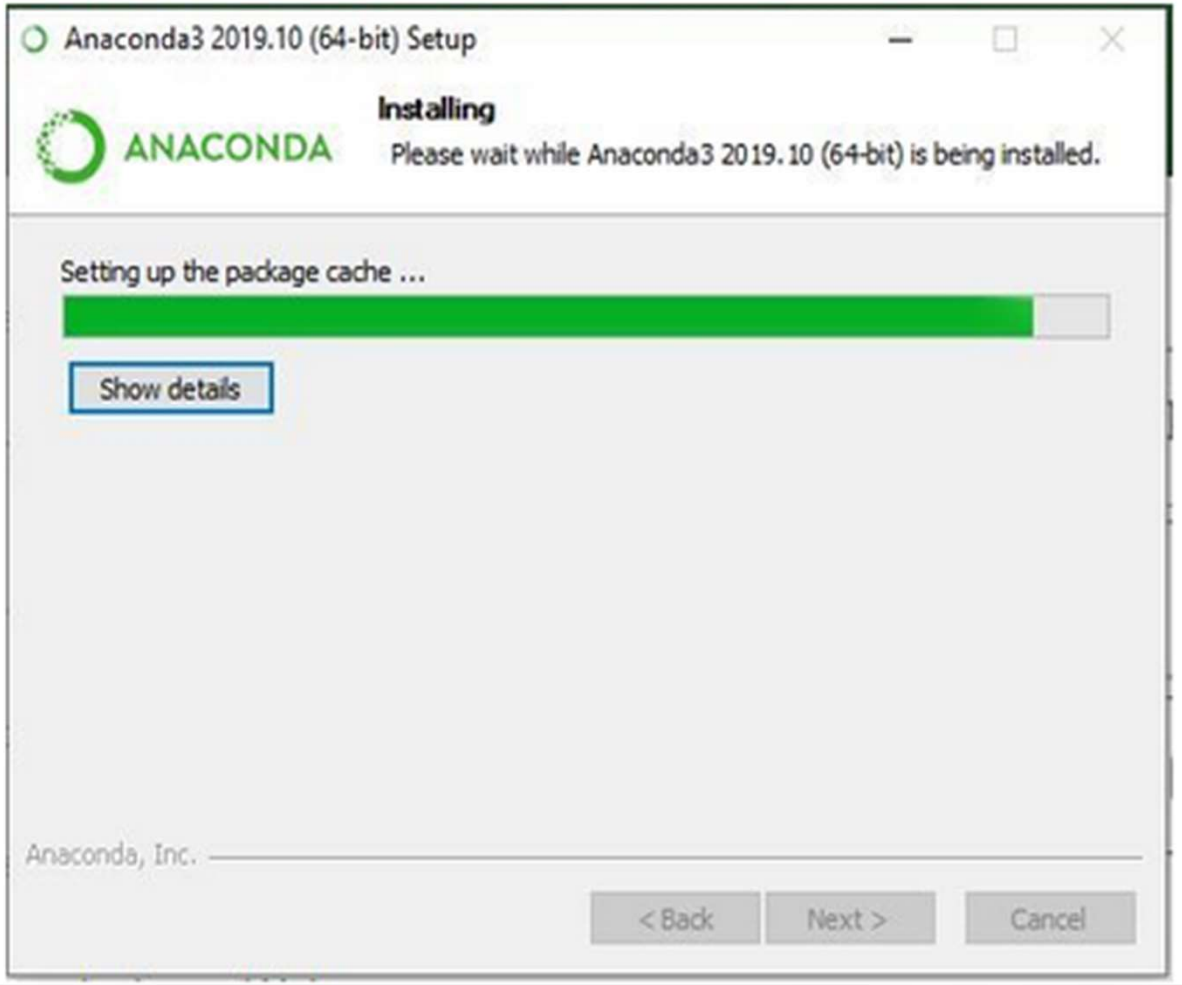
Anaconda Installation for Windows

Advanced Installation Option:



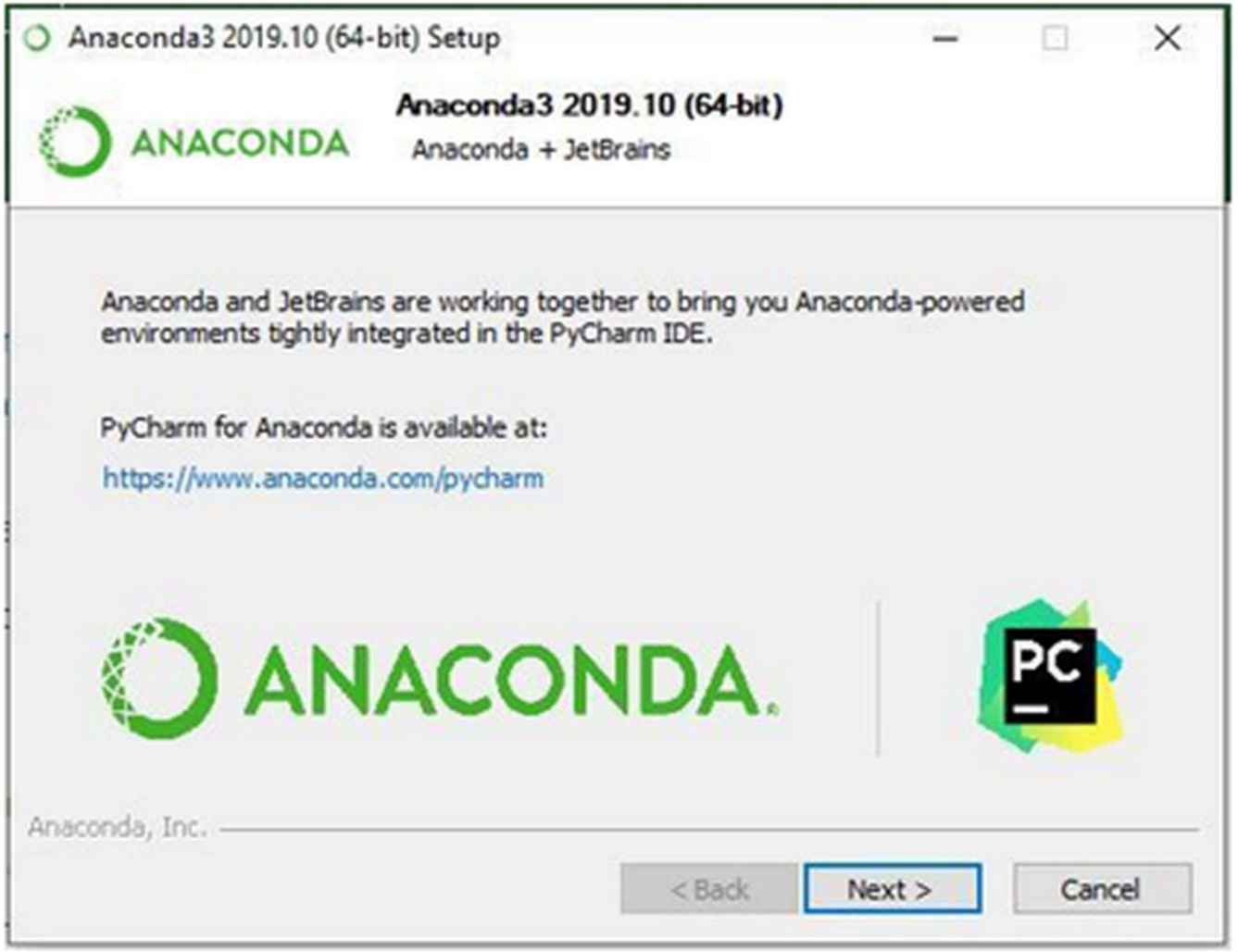
Anaconda Installation for Windows

Getting through the Installation Process:



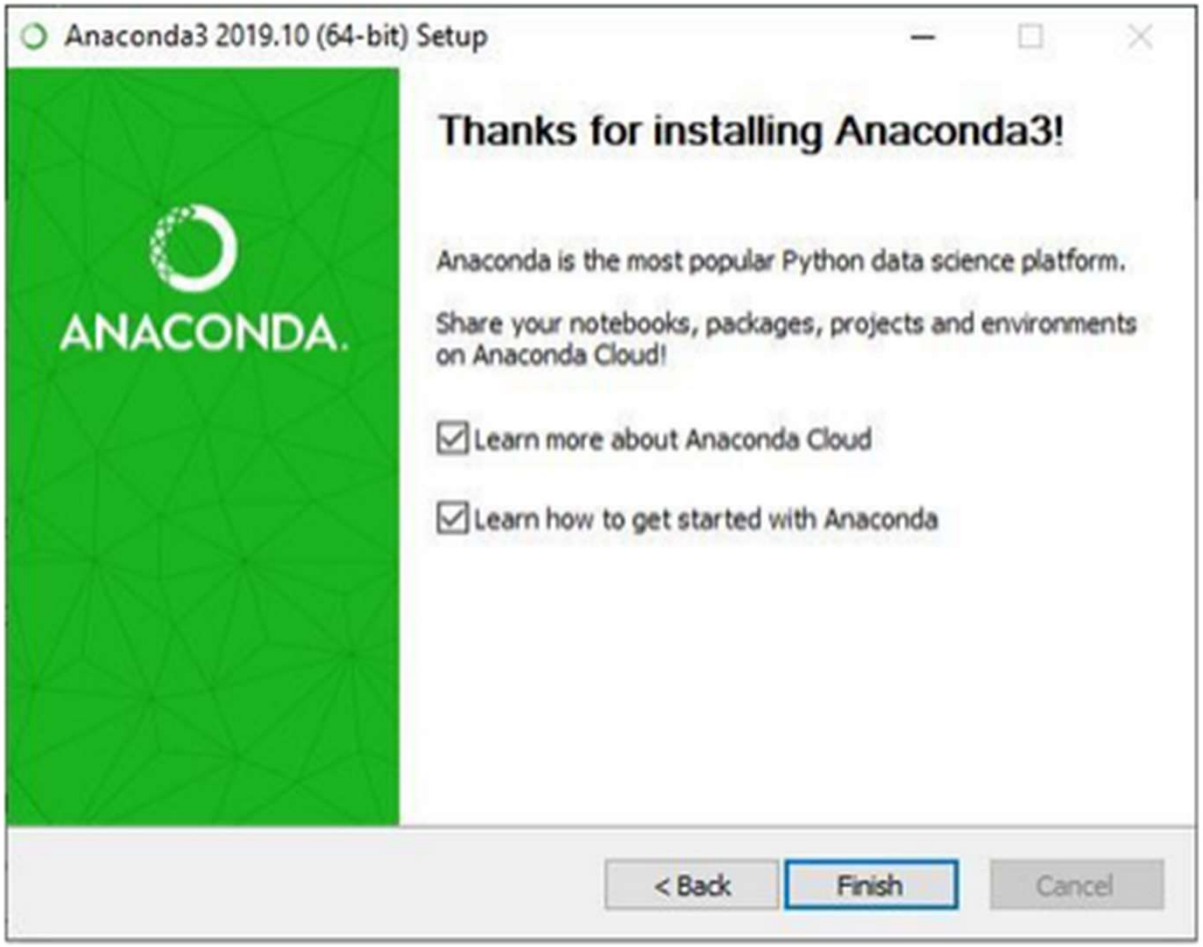
Anaconda Installation for Windows

Recommendation to Install Pycharm:



Anaconda Installation for Windows

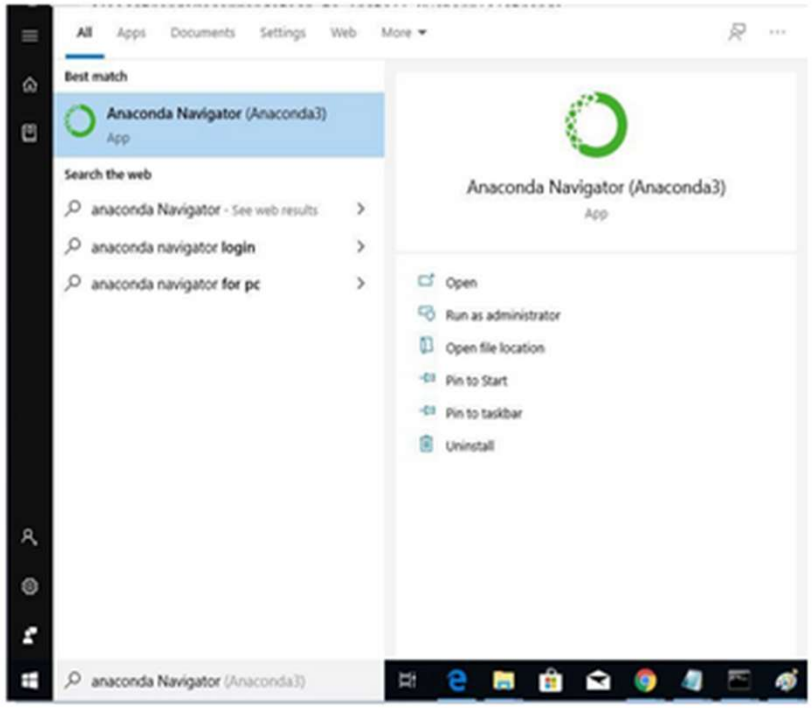
Finishing up the Installation:



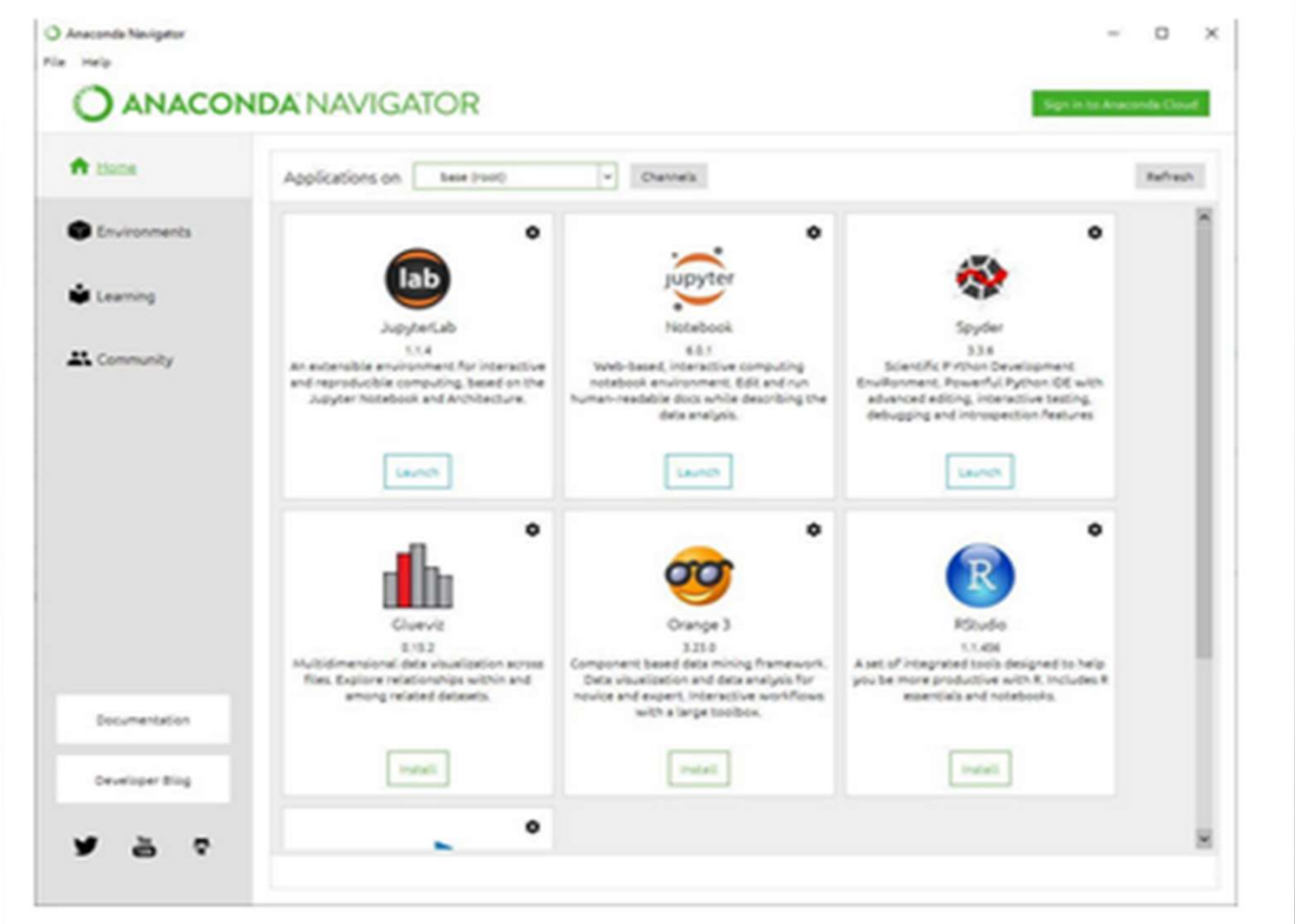
Anaconda Installation for Windows

Working with Anaconda:

Once the installation process is done, Anaconda can be used to perform multiple operations. To begin using Anaconda, search for Anaconda Navigator from the Start Menu in Windows



Anaconda Installation for Windows



References

[1] http://www.lac.inpe.br/~rafael.santos/Docs/CAP394/WholeStory-Iris.html
[2] github.com/swapnilthomare/Data-Science
[3] github.com/saedhussain/gcp_serverless_ml
[4] github.com/gaiyaobed/student_predict
[5] https://www.geeksforgeeks.org/how-to-install-anaconda-on-windows/
[6] https://www.datacamp.com/tutorial/installing-anaconda-windows

Weekly wrap-up and Q&A

THANK YOU