COIT20277 Introduction to Artificial Intelligence

Week 2

- Manage Anaconda Environments
- Scikit-Learn Installation and Basics
- Python for Supervised Learning

Acknowledgement of Country

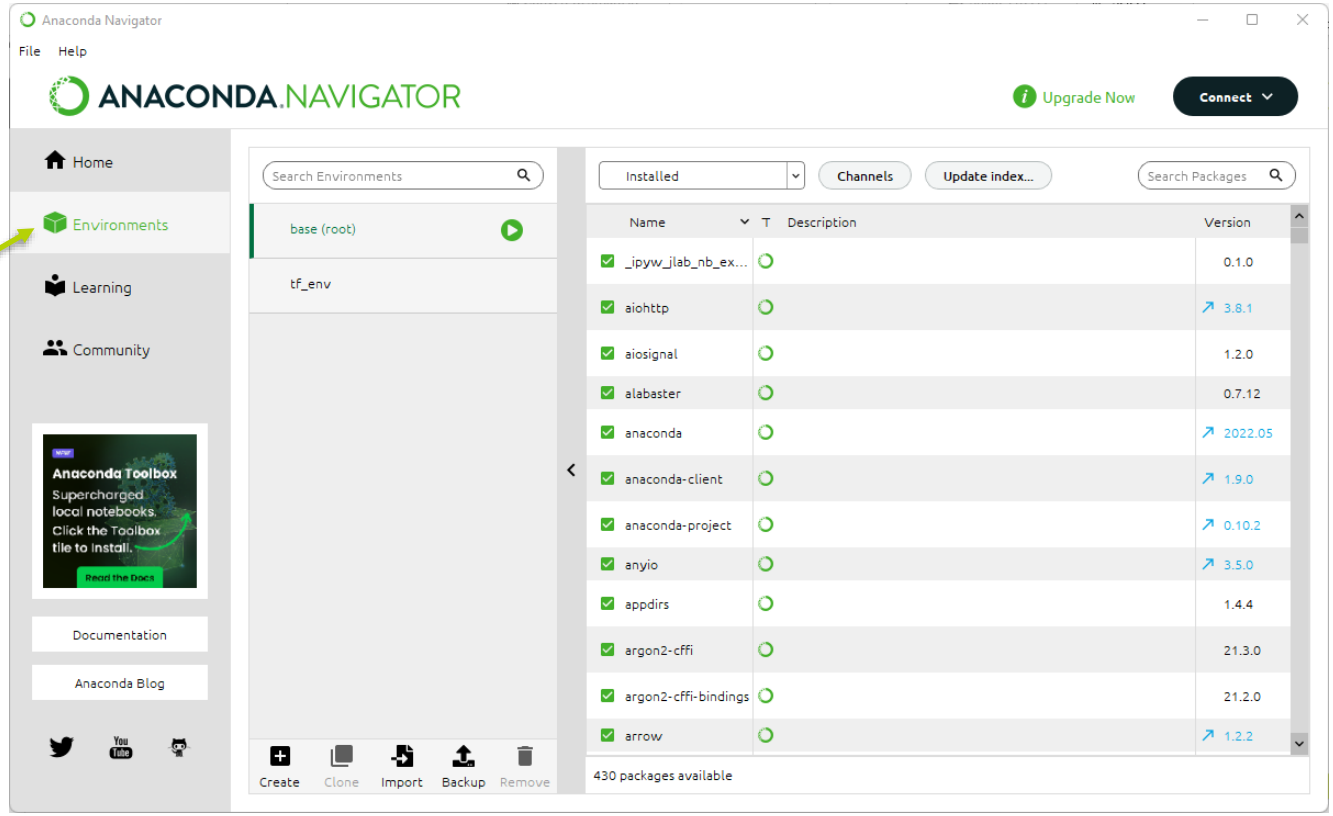
I respectfully acknowledge the Traditional Custodians of the land on which we live, work and learn. I pay my respects to the First Nations people and their Elders, past, present and future

CQUniversity AUSTRALIA

BE WHAT YOU WANT TO BE
cqu.edu.au

Managing environments

- On the **Environments** page, the left column displays your environments. At the bottom of the environments list are the Create, Clone, Import, Backup, and Remove buttons.

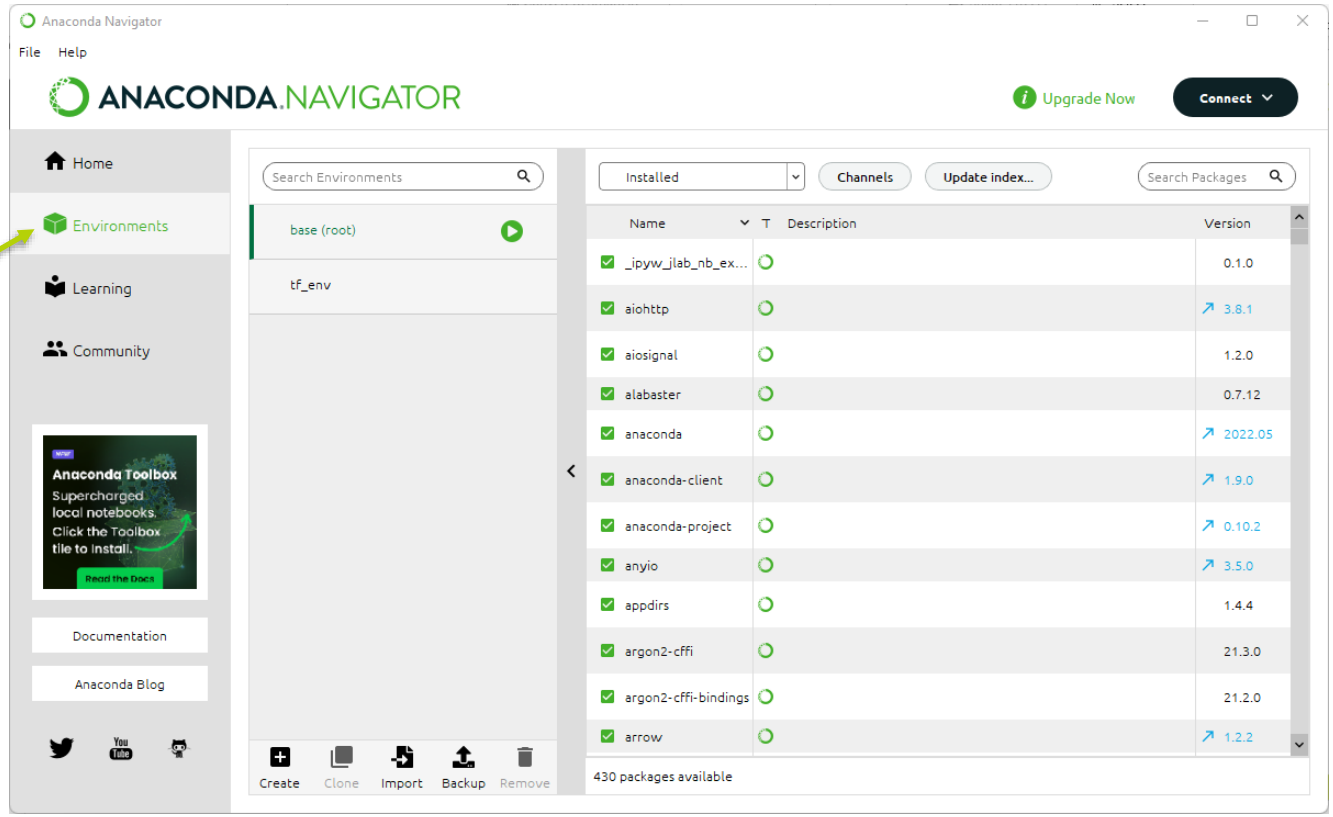


Click

Creating a new environment

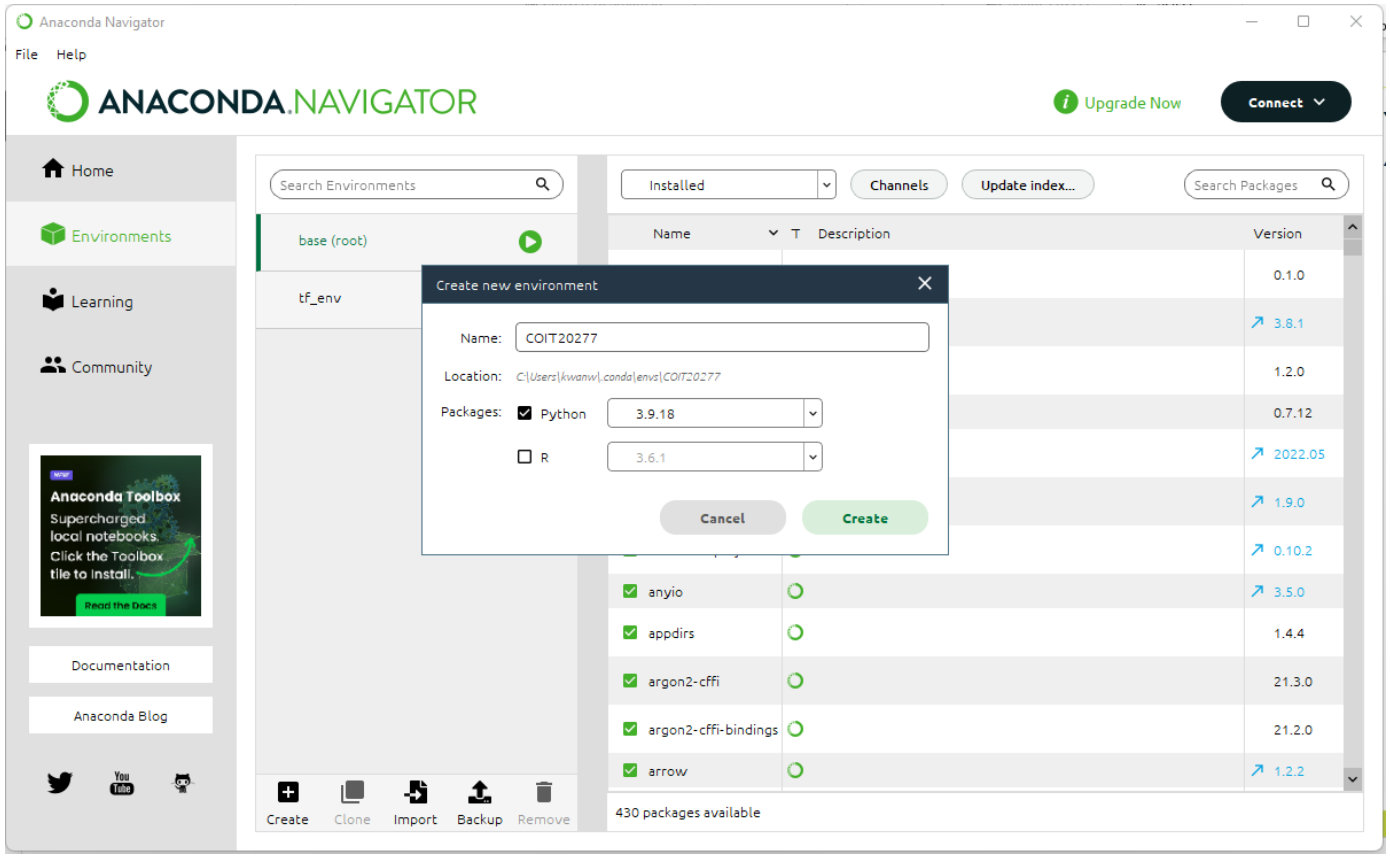
- At the bottom of the environments list, select **Create**.
- In the Create new environment dialog, enter **COIT20277** for the new environment.

Click



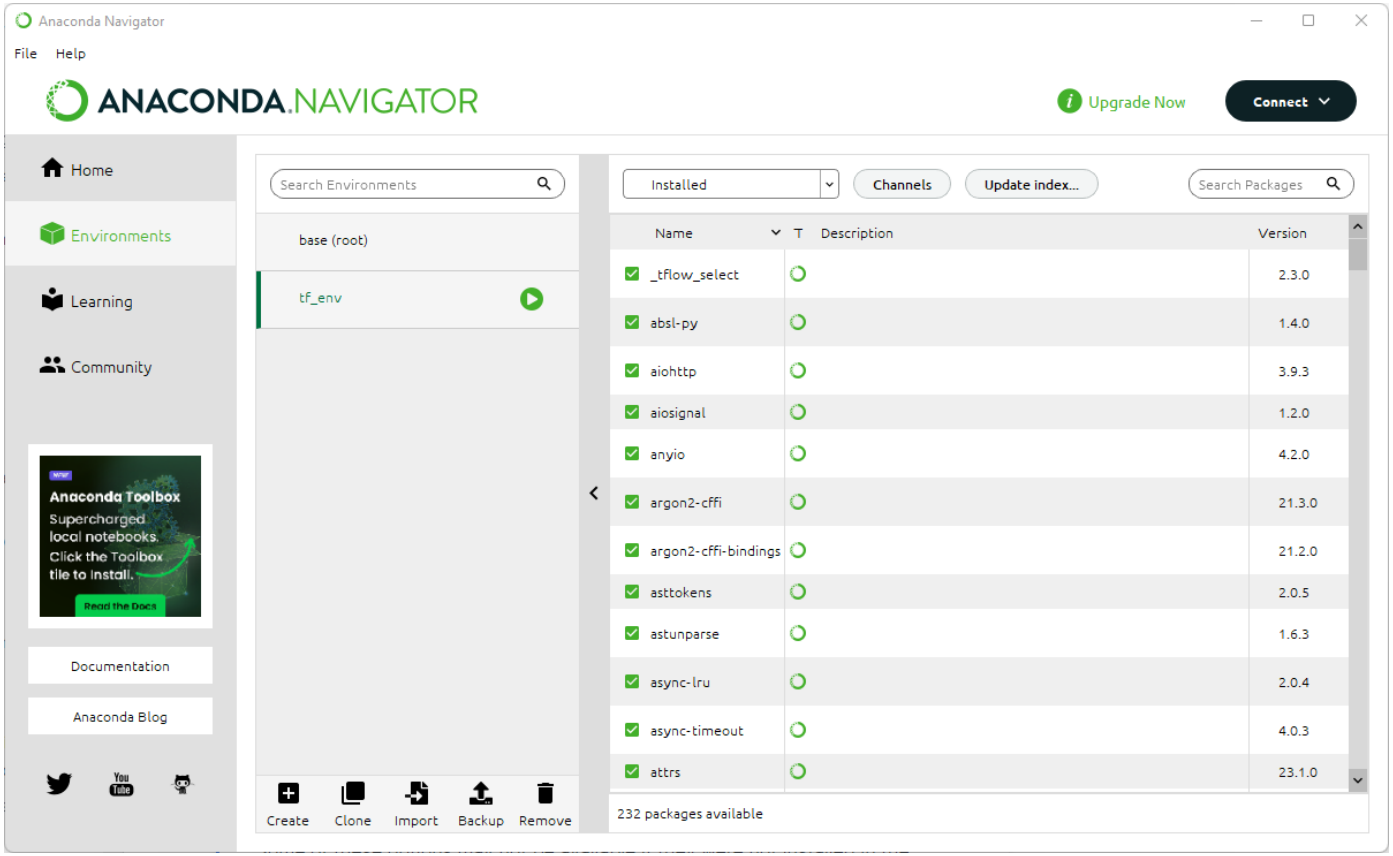
Creating a new environment (cont…)

- Select **Python (3.9.18)** to set the package type for the new environment.
- Click **Create**.



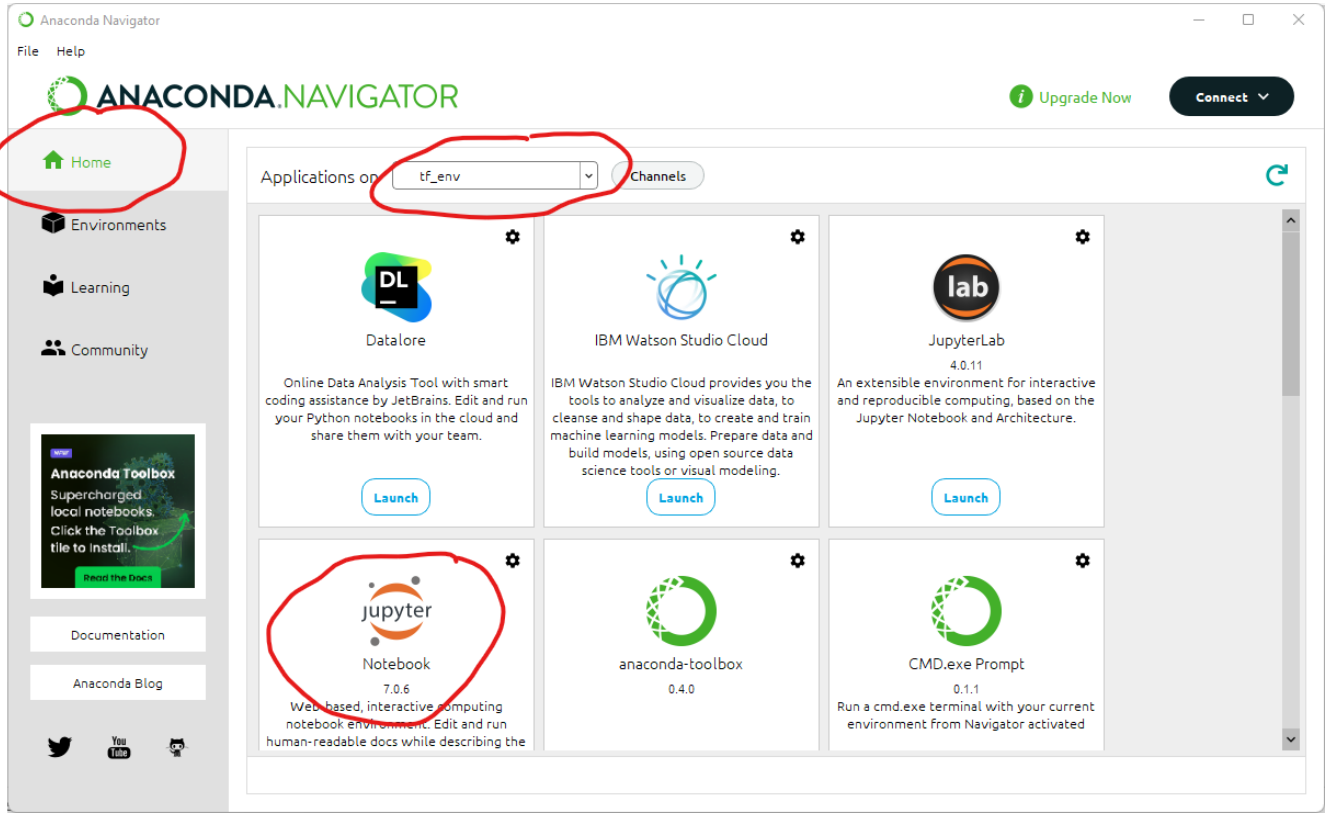
Using an environment

- In the environments list, select the environment name (in my case, tf_env) to activate it.
- Then, launch



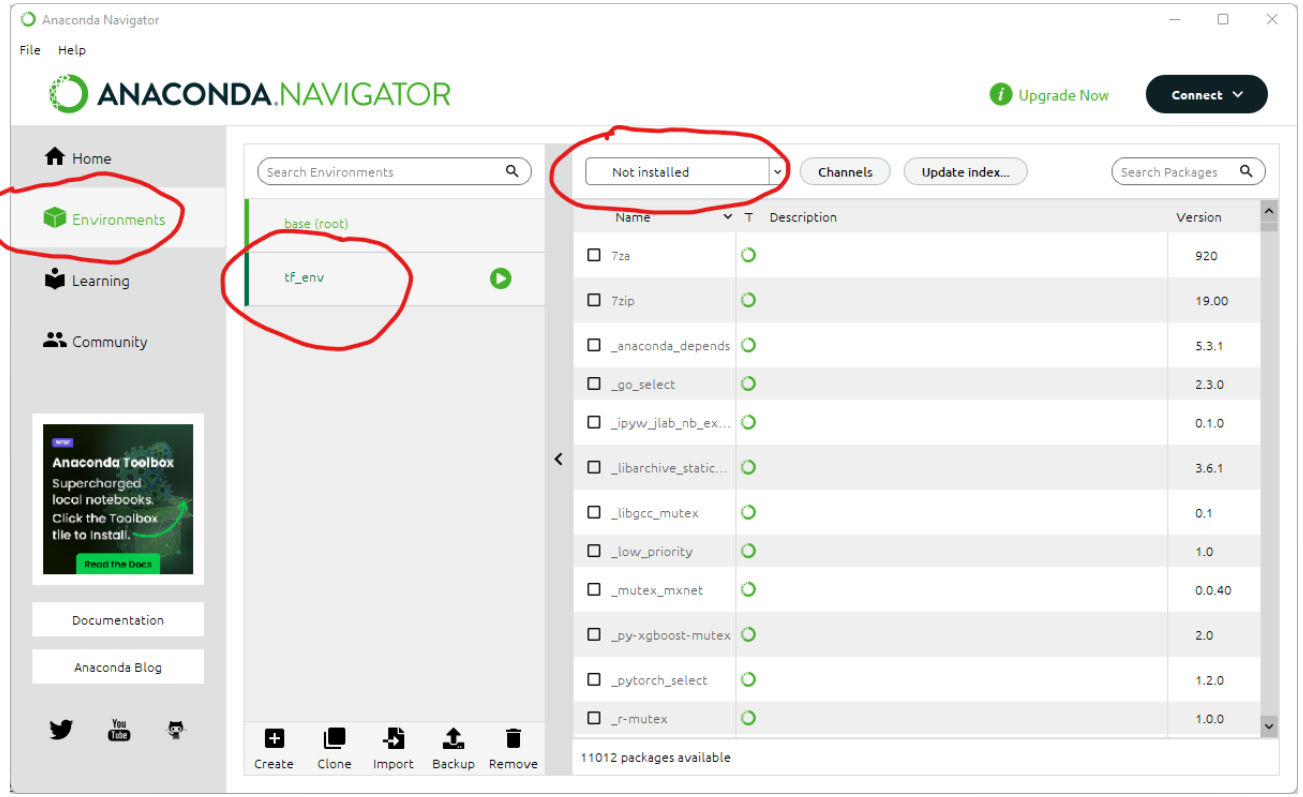
Using an environment (cont…)

- Select **Home** on the left column.
- When Jupyter Notebook is launched, it will use settings and packages installed in this environment.



Scikit-Learn Installation

- Scikit-Learn is just one of Python packages that you can install in your current environment.
- Install Scikit-Learn by following the instructions at https://docs.anaconda.com/free/navigator/tutorials/manage-packages/



Additional Packages Needed

- Numpy: For dealing with *N*-dimensional array data
- Pandas: For reading data files and analyzing data
- Matplotlib: For plotting data
- SciPy: For scientific computing

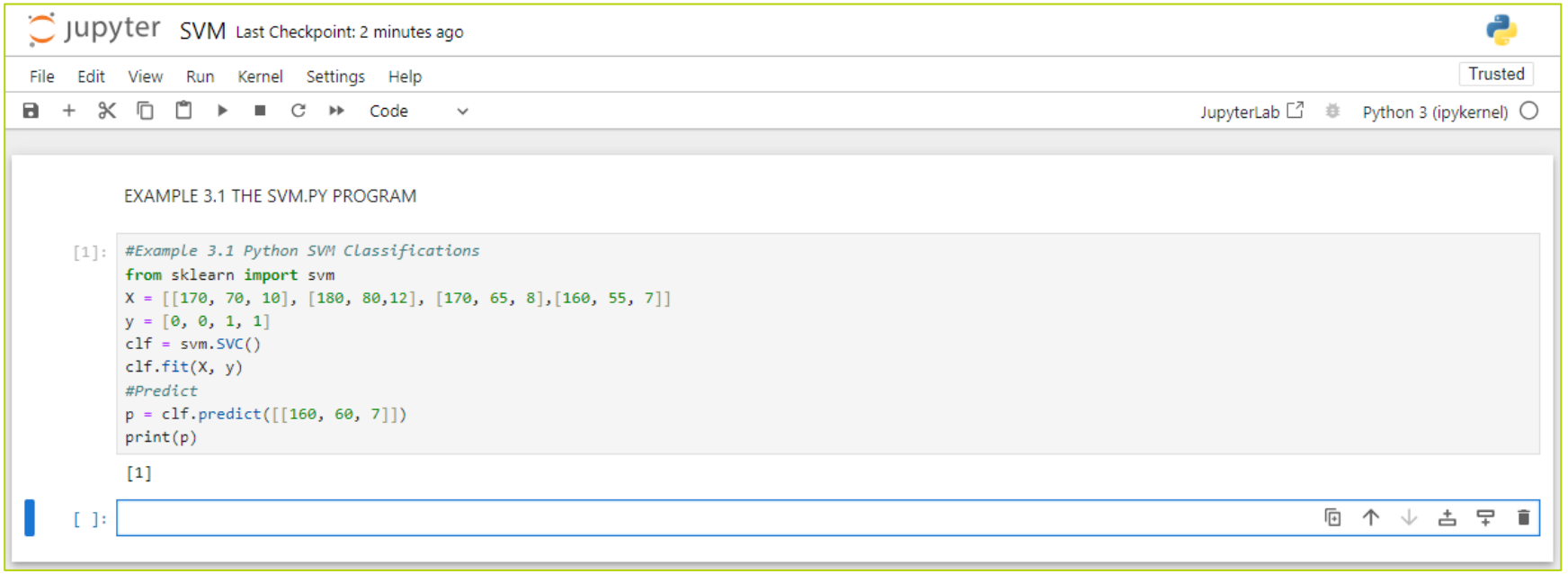
Supervised Learning: Classification

- The supervised learning algorithms explored in this tutorial are:
  - Support vector machines
  - Naïve Bayes
  - Decision trees and Random forest
  - K-nearest neighbor

- You can learn more about these algorithms implemented in scikit-learn by accessing https://scikit-learn.org/stable/supervised_learning.html

Support Vector Machine (Prac #1)

- Example 3.1 shows a simple SVM gender classification example based on height, weight, and shoe size.
- It first uses `from sklearn import svm` to import the SVM library. It uses an array called `X` to store four sets of values of height in centimeters, weight in kilos, and shoe size in UK size.
- It uses an array named `y` to store four sets of known genders, 0 for Male, 1 for Female.
- It then trains the SVM classifier and makes a prediction for a given height, weight, and shoe size `[[160, 60, 7]]`.

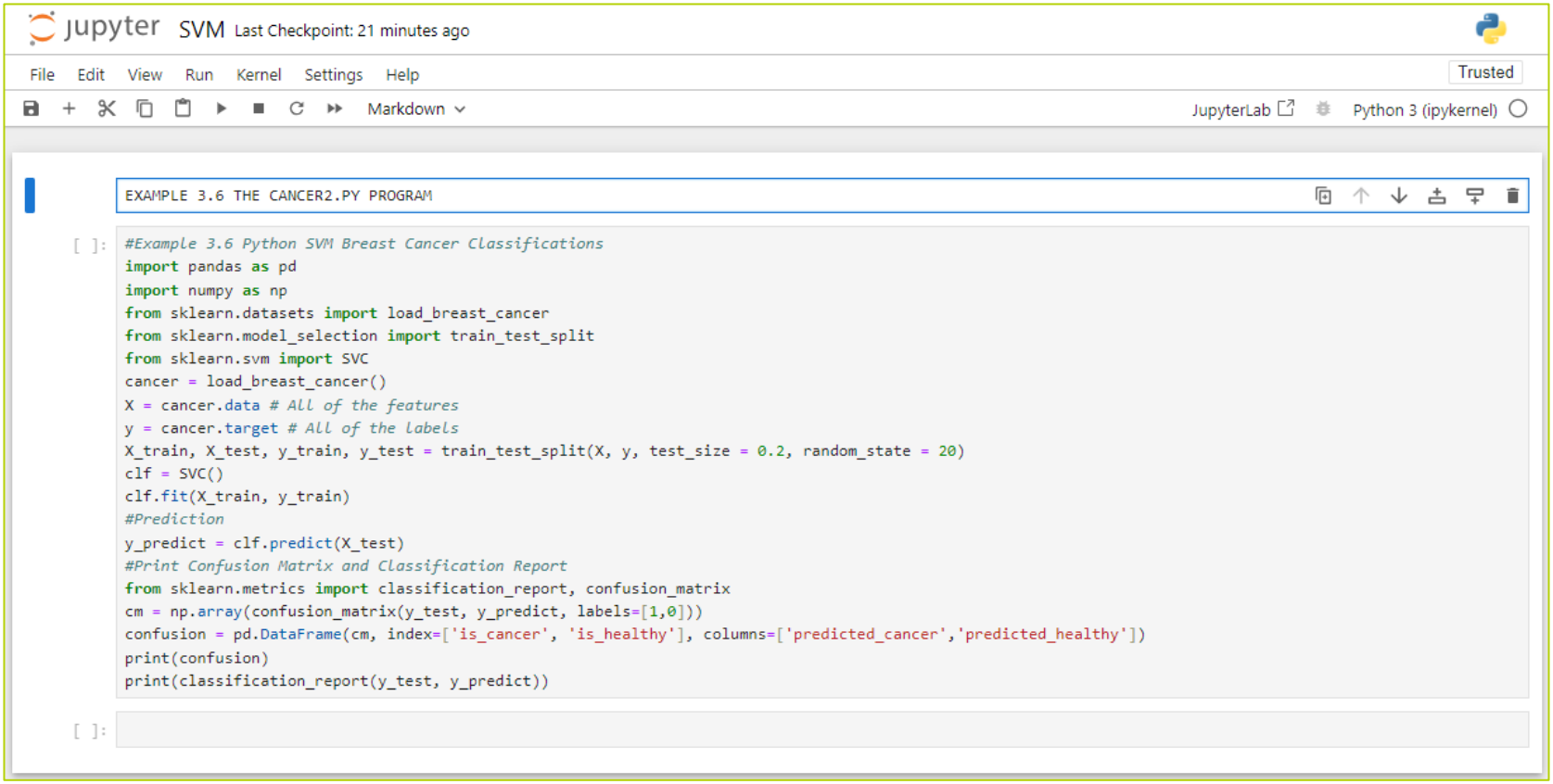


Prac #1

- Modify the Python program from the last slide to use twice the number of samples of `X` and `y`.
- Submit your Python program at the end of the tutorial.

Support Vector Machine (Prac #2)

- It first loads the breast cancer data's `load_breast_cancer()` function and then puts the data into a dataframe format, uses features data as `X` and target as `y`.
- It uses `train_test_split()` to split `X` and `y` into training and testing sets.
- Specifically, 80 percent of data is for training, and 20 percent is for testing.
- It uses the training data X_train and y_train to train an SVM and uses testing data X_test to make predictions. w

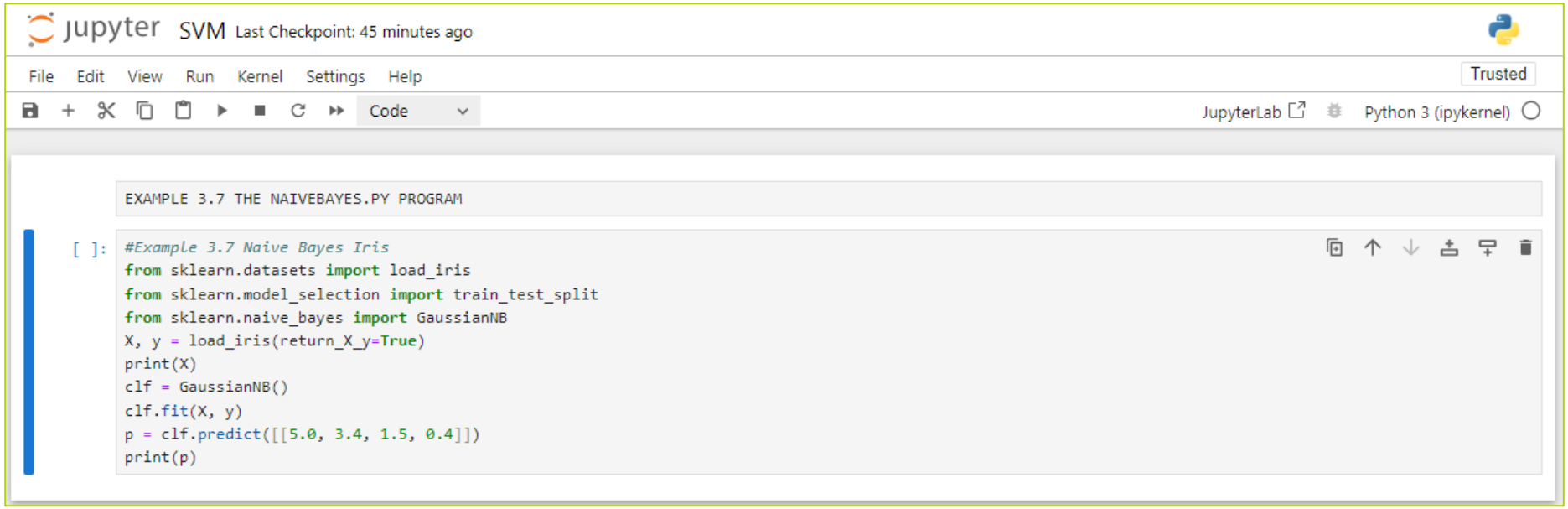


Prac #2

- Enter the Python program shown in the last slide in your Jupyter Notebook.
- Run the code to explore and make sure you understand the output.
- Experiment by varying the sizes of the training and the test sets. Plot the classification accuracy versus the size of the training set.
- Submit your Python program at the end of the tutorial.

Naïve Bayes

- In the Python code, it uses `X, y = load_iris(return_X_y=True)` to load the iris data and returns the data as `X` and `y`, where `X` includes all four features of that data.
- It then trains the naive Bayes classifier model and makes a prediction for a given sample with the sepal's length and width and the petal's length and width.

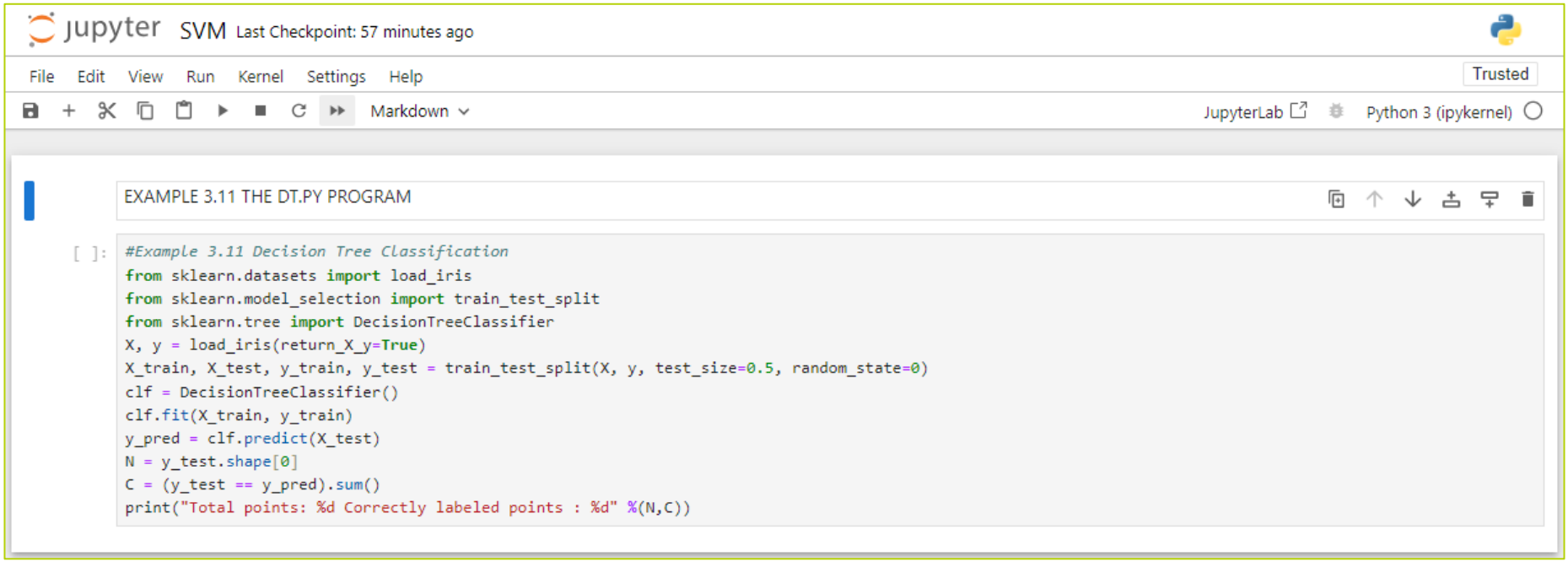


Prac

- What does `GaussianNB` refer to in the Python code? Report on this in your submission.
- Modify the code so that it uses different training and test splits. Run the code and make sure you understand the output.
- Experiment by varying the sizes of the training and the test sets. Plot the classification accuracy versus the size of the training set.
- Submit your Python program at the end of the tutorial.

Decision Trees

- In the Python code, it uses `X, y = load_iris(return_X_y=True)` to load the Iris data and returns the data as `X` and `y`, using all four features of the data.
- It then splits the data into the training set and the testing set.
- It then trains the decision tree classifier model and makes predictions on the testing set.
- It also calculates and displays the total number of points, as well as the number of points that are correctly predicted.



Prac

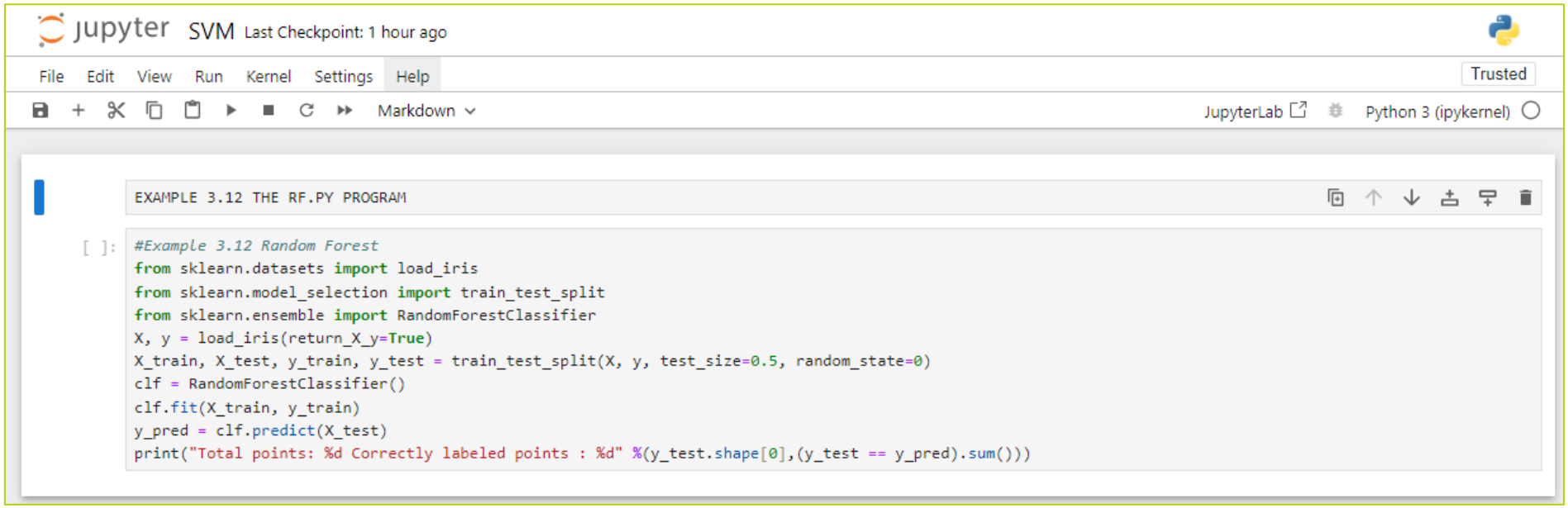
- Modify the Python program from the last slide so that it performs decision tree classification on Scikit-Learn's wine data.
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html

- Experiment by varying the sizes of the training and the test sets. Plot the classification accuracy versus the size of the training set.

- Submit your Python program at the end of the tutorial.

Random Forest

- In the Python code, it uses `X, y = load_iris(return_X_y=True)` to load the Iris data and returns the data as `X` and `y`, using all four features of the data.
- It then splits the data into the training set and the testing set.
- It then trains the random forest classifier model and makes predictions on the testing set.
- It calculates and displays the number of points that are correctly predicted.



Prac

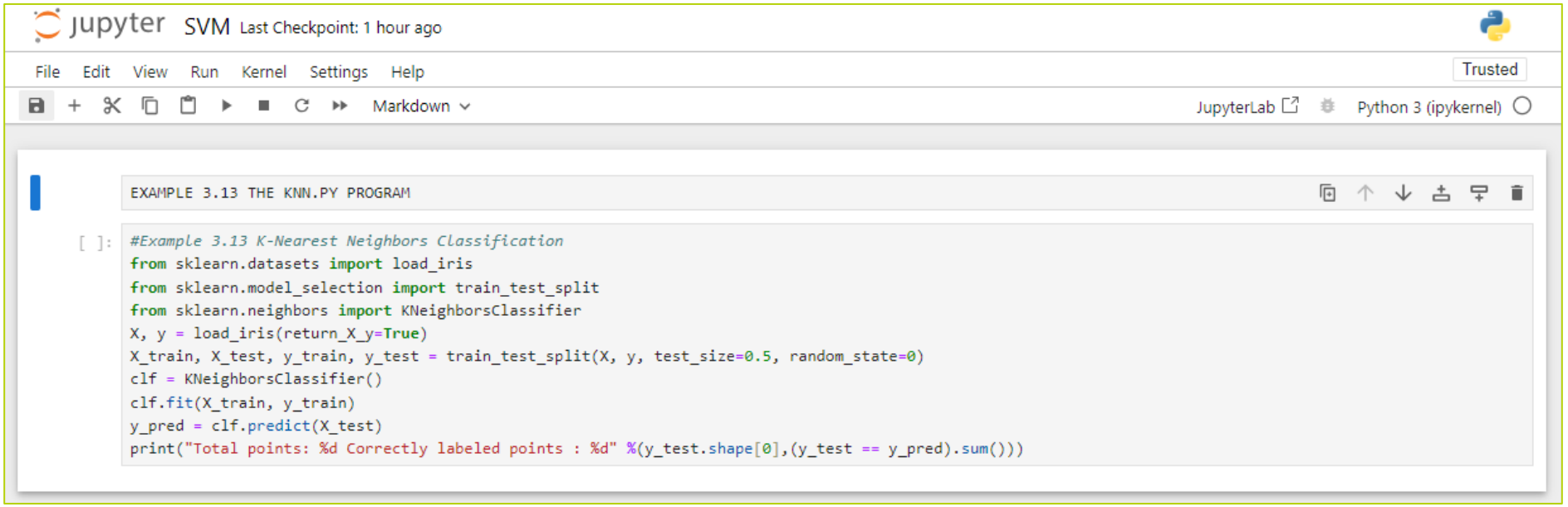
- Modify the Python program from the last slide so that it performs random forest classification on Scikit-Learn's diabetes data.
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes

- Plot the classification accuracy versus the size of the training set.

- Submit your Python program at the end of the tutorial.

K-Nearest Neighbors

- In the Python code, it uses `X, y = load_iris(return_X_y=True)` to load the Iris data and returns the data as `X` and `y`, using all four features of the data.
- It then splits the data into the training set and the testing set.
- It then trains the K-NN classifier model and makes predictions on the testing set.
- It calculates and displays the number of points that are correctly predicted.



Prac

- Modify the Python program from the last slide so that it performs K-Nearest Neighbors classification on Scikit-Learn's diabetes data.
- https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_diabetes.html#sklearn.datasets.load_diabetes

- Plot the classification accuracy versus the size of the training set.

- Submit your Python program at the end of the tutorial.

THANK YOU

TIME FOR DISCUSSION & QUESTIONS