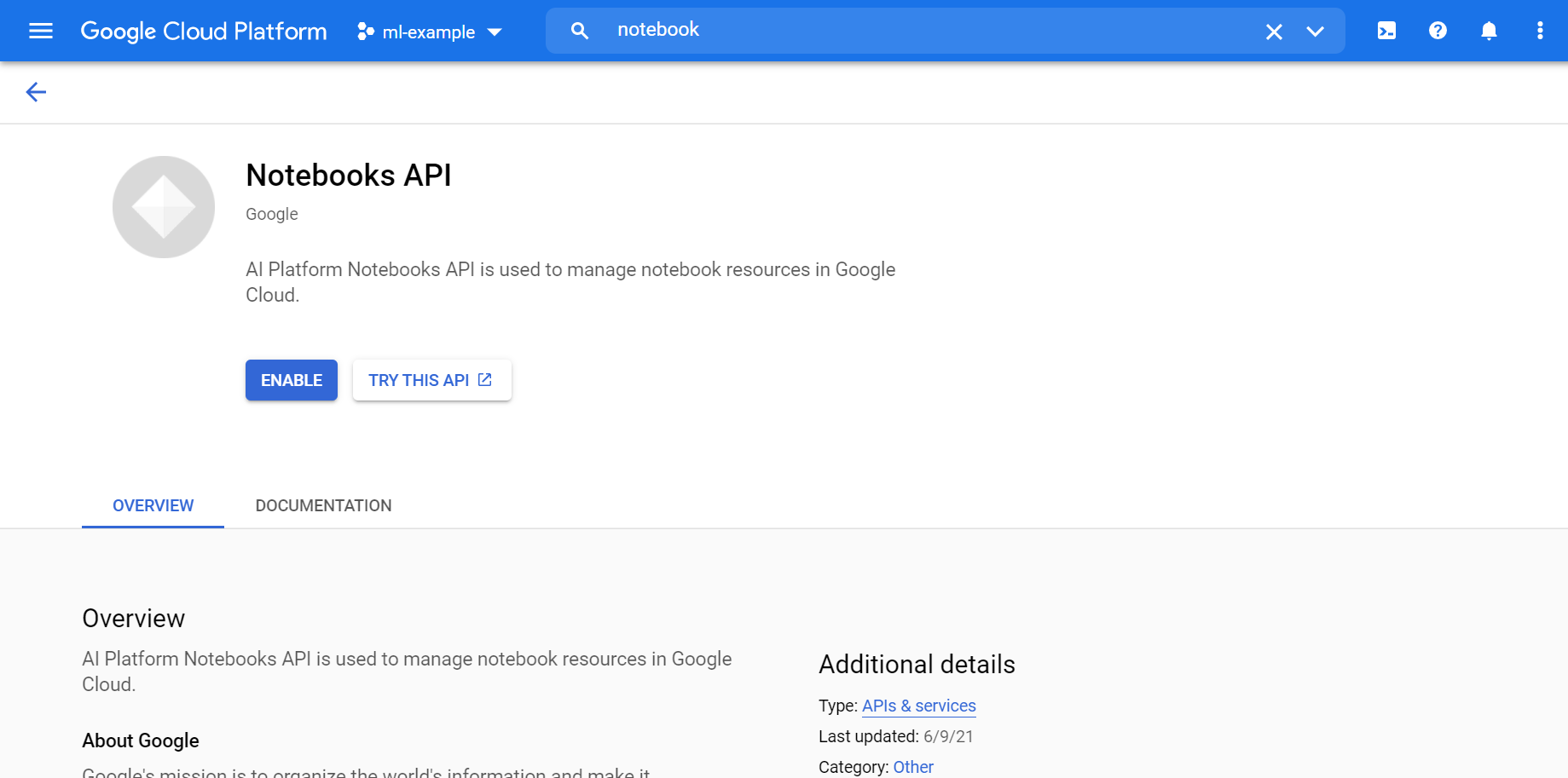
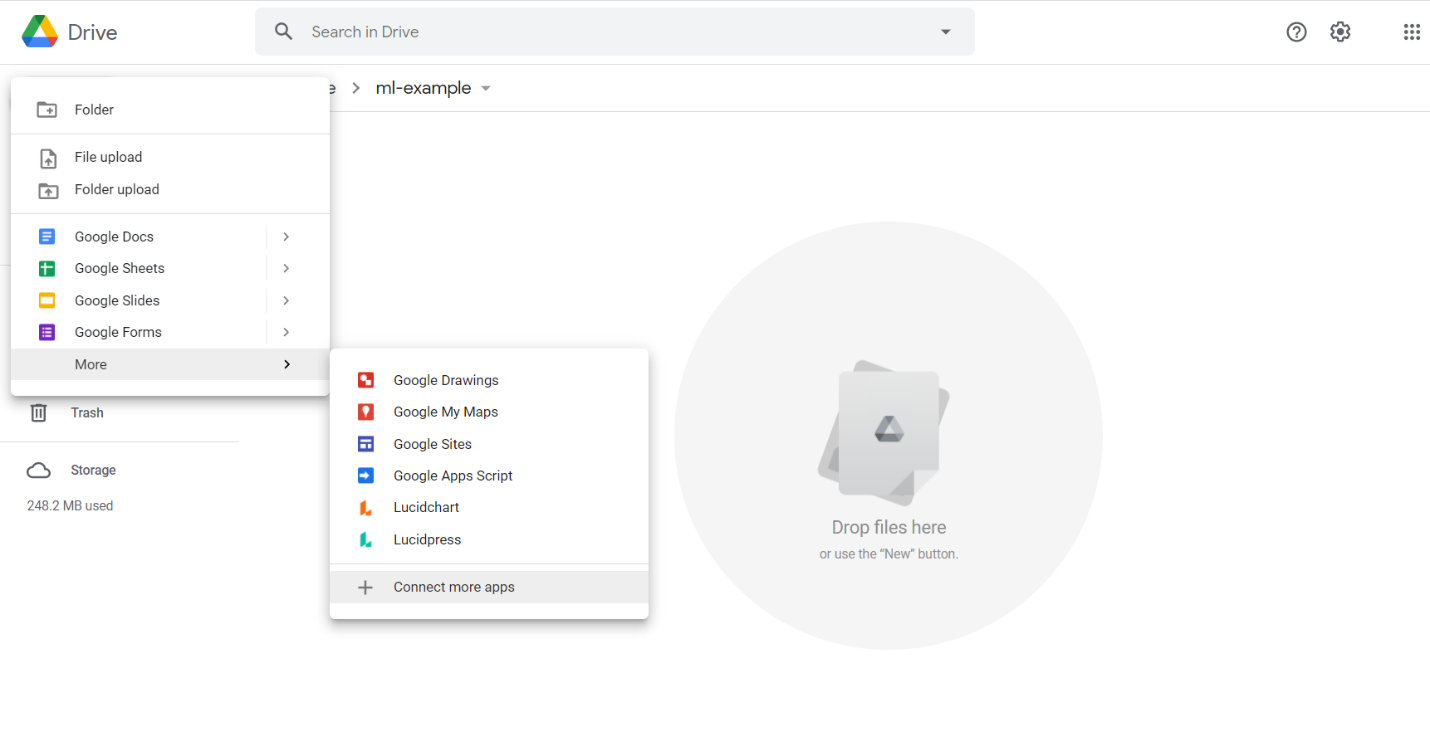
In order to develop Python programs, we essentially need three resources: storage, processing, and an IDE. When you SSH into a virtual instance, you’re using GCP storage and processing, but you’re limited to a command line IDE like VIM (unless you configure SSH on a graphical IDE like VSCode). So, is there an easy way to let GCP provide a graphical IDE as well as the processing and the storage? The short answer is yes. One possible (but not recommended) approach to achieve this end is to create a Notebooks instance for an existing GCP project. You can do this by selecting your desired project in the GCP console, then searching for notebooks. You should find a page that looks like this:

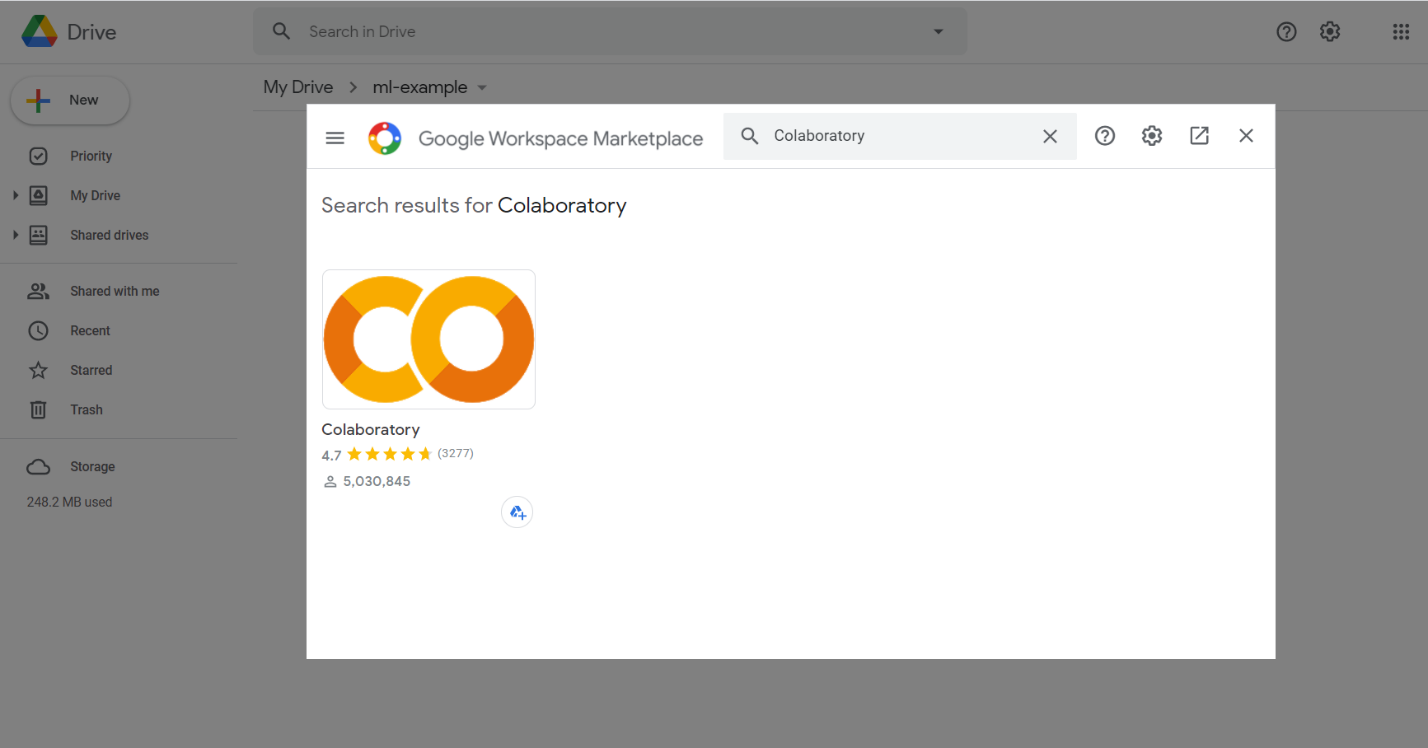


After clicking enable, you can navigate to the Notebooks page in the GCP console. At this point, you will be able to create and configure a new Notebooks instance. Once your instance has been created, an Open JupyterLab link will become active. You can follow this link and begin developing on a Jupyter Notebook. The code you write here will be run on your provisioned resources for your project, and you will have access to any storage resources from your project as well. However, I would **not recommend** that most readers use this approach to developing on Google Cloud. First, it’s very costly to maintain sufficient resources on your GCP project. Second, GCP Notebooks have a lot of capablities that *you will likely never use*. So, unless you have extremely high processing demands or need to interface with other GCP resources, you **should not use** GCP Notebooks.

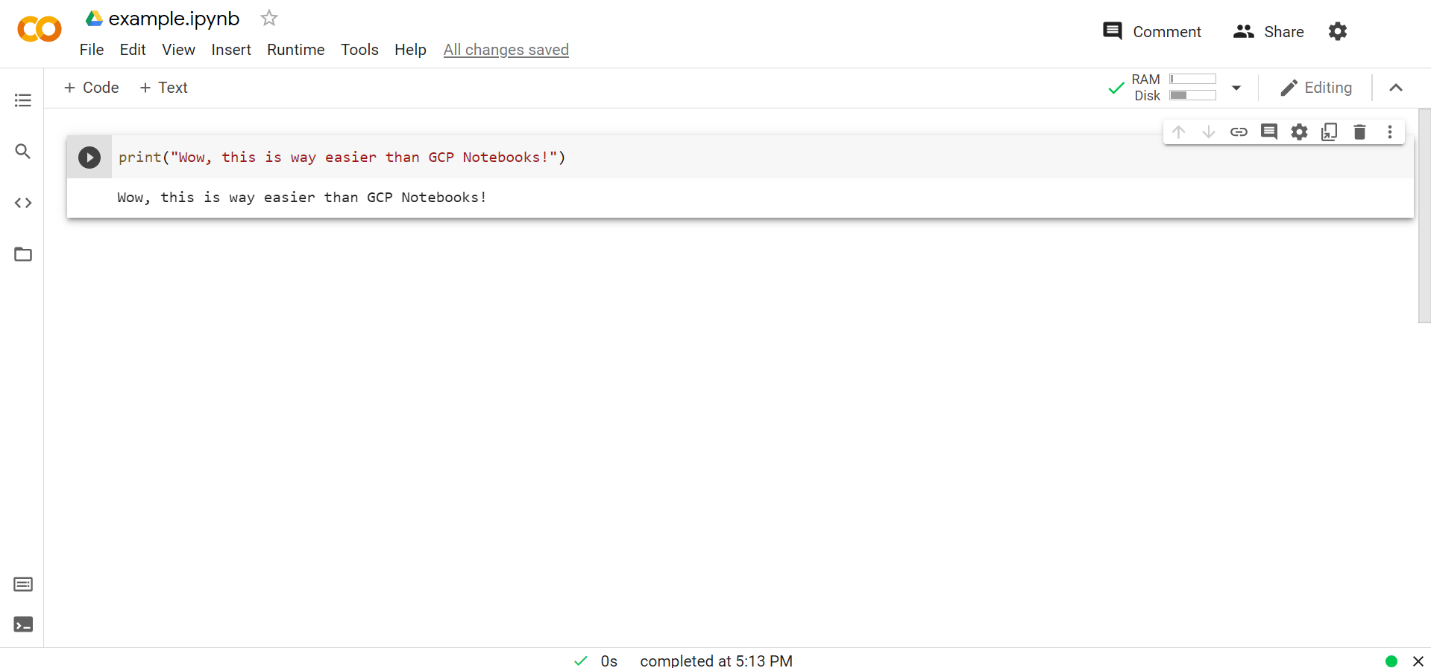
Instead, I suggest that you use a **completely** **free** alternative from Google: Google Colaboratory (Colab). When you develop in Colab, your code will be run on resources owned by Google, but you do not need to associate any Colab notebook with a GCP project. In fact, you don’t even need an active GCP account to start developing on Colab—all you need is an active Google account.

To get started with Colab, first navigate to your Google Drive. To see if you have Colab installed, select “New” then hover over “More.”

If Google Colaboratory does not appear as an option, you’ll need to install the app by selecting “connect more apps” and searching for Colaboratory, which will appear as below.

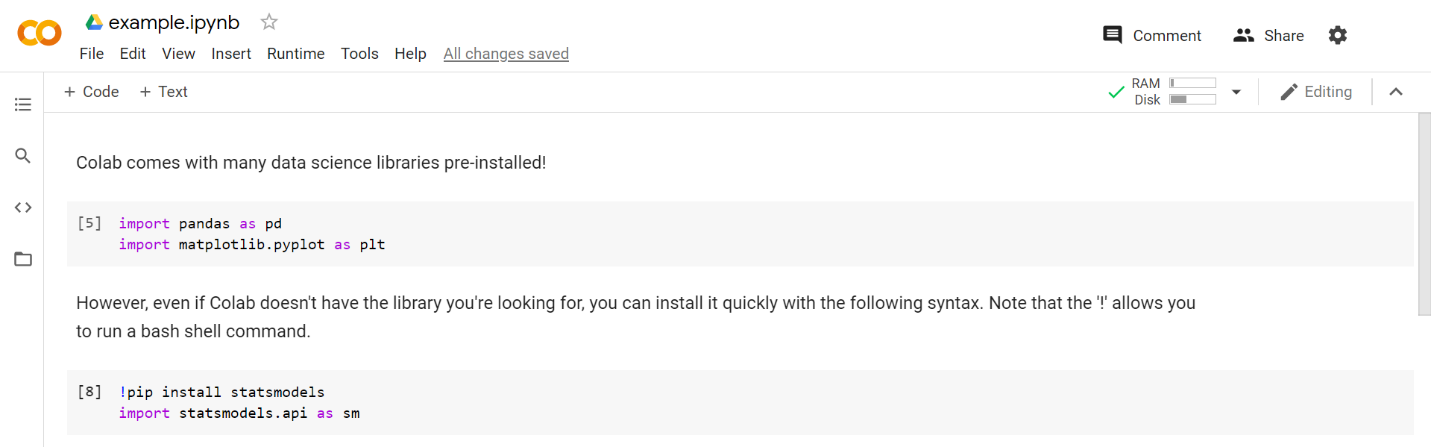


After installing Colab, you will be able to create a new Colaboratory notebook from the “New” menu. Congratulations, you can now start writing Python3 code in this notebook! In Colab (and in any Jupyter environment), you can split your Python script into cells and then run each cell individually by clicking the “run” button on the top left corner of each cell. This makes it easy to test components of your script as you go: just press “run,” see if it works as expected, and if it doesn’t, make changes. You can also delete an unwanted cell using the “trash” symbol in the upper-right corner.

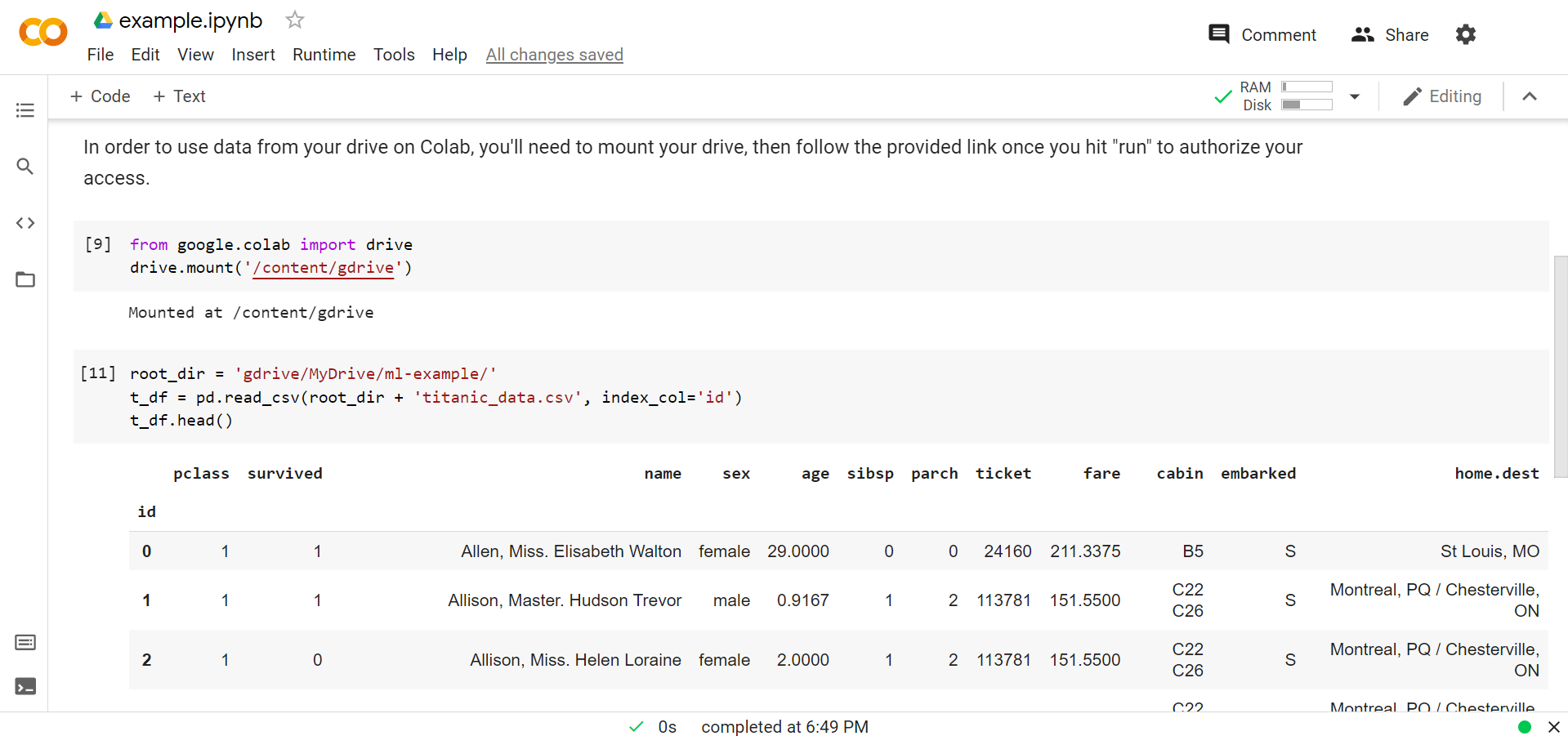


Colab also comes with many data science libraries like pandas, numpy, and scikit-learn already installed. Moreover, it’s easy to use data from your Google drive in your Colab scripts. As an example, let’s pretend that you want to follow along with a logistic regression example from chapter 5.

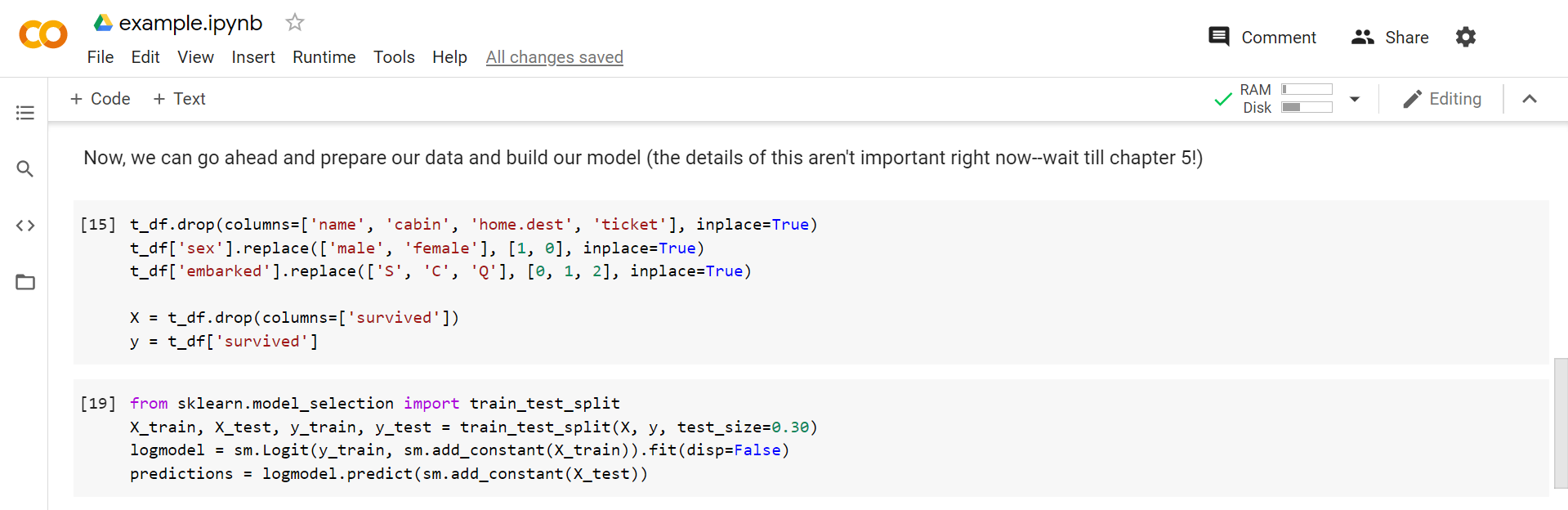
First, you import and download any necessary libraries.



Next, you download the necessary data (assuming you have it stored in the same folder you are currently working in).



Now, you can go ahead and build your model! Don’t worry if you don’t understand the Python code for this step—you’ll learn about it in chapter 5.



Finally, we can visualize our results.



As you can see, Google Colab is a simple and powerful tool for developing Python programs, especially when machine learning is involved. For the majority of readers, I’d recommend using Google Colab over any alternative options if you plan on developing on the cloud.