How to Use Kaggle

Current Page Notebooks

Notebooks

Explore and run machine learning code with Kaggle Notebooks, a cloud computational environment that enables reproducible and collaborative analysis

Types of Notebooks

There are two different types of Notebooks on Kaggle.

Scripts

The first type is a script. Scripts are files that execute everything as code sequentially. To start a script, click on "Create Notebook" and select "Script". This will open the Scripts editing interface.

From here you may select what type of script you would like to execute. You may write scripts in R or in Python.

You can also execute selected lines of code by highlighting the code in the editor interface and clicking the "Run" button or hitting shift-enter. Any results will be printed to the console.

"Deep Learning Support [.9663]" from the TalkingData AdTracking Fraud Detection Challenge is a great example of a Script-type.

RMarkdown Scripts

RMarkdown scripts are a special type of script that executes not just R code, but RMarkdown code. This is a combination of R code and Markdown editing syntax that is prefered by most R authors in our community.

The RMarkdown editor is the same one used for basic R or Python scripts, except that it uses the special RMarkdown syntax. To start editing an RMarkdown script, click on "Create Notebook", navigate to the "Scripts" pane, and click on that. Then, in the language dropdown, click on "RMarkdown".

"Head Start for Data Science" is a great example of a RMarkdown Script-type.

Notebooks

programming language of your choice (for writing code). To start a notebook, click on "Create Notebook", and select "Notebook". This will open the Notebooks editing interface.

Notebooks may be written in either R or Python.

"Comprehensive data exploration with Python" is a great example of a Python Jupyter Notebook-type. "How to Become a Data Scientist" is a great example of an R Jupyter Notebook-type.

Searching for Notebooks

In addition to being an interactive editing platform, you can find and use code that others in the community have shared public. Kagglers working with data across both the Datasets and Competitions platforms are constantly building cool things. Exploring and reading other Kagglers' code is a great way to both learn new techniques and stay involved in the community.

There's no better place than Kaggle Notebooks to discover such a huge repository of public, open-sourced, and reproducible code for data science and machine learning.

The latest and greatest from Notebooks is surfaced on Kaggle in several different places.

Site Search

You can use the site search in the top bar of the website while on any page to look for not only Notebooks but Datasets, Competitions, Users, and more across Kaggle. Start typing a search query to get quick results and hit "Enter" to see a full page of results that you can drill down into. From the full page search results, you can filter just to "Notebooks" and add even more filter criteria using the filter options on the left hand side of the page.

Homepage

When you're logged into your Kaggle account, the Kaggle homepage provides a live newsfeed of what people are doing on the platform. While Discussion forum posts and new Datasets make up some of the contents of the home page, most of it is dedicated to hot new Notebooks activity. By browsing down the page you can check out all the latest updates from your fellow Kagglers.

You can tweak your newsfeed to your liking by following other Kagglers. To follow someone, go to their profile page and click on "Follow User". Content posted and upvotes made by users you have followed will show up more prominently.

The same is true of other users who choose to follow you. Post high-quality notebooks and datasets and you will soon find other users following along with what you are doing!

Notebook Listing

A more structured way of accessing Notebooks is the Notebook listing, accessible from the "Notebooks" tab in the main menu bar.

The Notebook listing is sorted by "Hotness" by default. "Hotness" is what it sounds like: a way of measuring the interestingness of Notebooks on the platform. Notebooks which score highly in Hotness, and thus appear highly in this list, are usually either recently written Notebooks that are scoring highly in things like upvotes and views, or "all-time" greats that have been consistently popular on the platform for a long time.

Other methods of sorting are by

- Most Votes: Surfaces the most popular notebooks of all time
- Most Comments: Returns the most discussed notebooks of all time
- Recently Created: A real-time stream of new Notebooks
- Recently Run: A real-time stream of activity
- Relevance: Sorts the results based on their relevance to the query

Other filtering options, available from the navigation bar, are Categories (Datasets or Competitions?), Outputs, Languages (R or Python?), and Types (Script or Notebook?).

You can also use the Notebook listing to sort through your own Notebooks ("Your Work"), find Notebooks that others have shared with you ("Shared With You"), or to look at Notebooks you have previously upvoted ("Favorites").

Finally, a Notebooks-specific search bar is available here. This is often the fastest way to find a specific Notebook that you are looking for.

Datasets and Competitions

Data on Kaggle is available through either Datasets or our Competitions. Both prominently feature the best community-created Notebooks on the "Notebooks" tab. Browsing Notebooks on Datasets and Competitions provides a way to quickly get acquainted with a specific dataset. You can fork any existing public Notebook to make a copy of the code and start experimenting with changes.

The Iris Species dataset and the Titanic competition are two classic examples of Datasets and Competitions, respectively, hosting great Notebooks on their content.

Tags and Tag Pages

Tags are the most advanced of the searching options available in the Notebook listing page. Tags are added by Notebook owners to indicate the topic of the Notebook, techniques you can use (e.g., "classification"), or the type of the data itself (e.g., "text data"). You can navigate to tag pages to browse more content sharing a tag either by clicking on a tag on a Notebook, or by searching by tag using the tag-specific search syntax: `tag:[TAG NAME]`.

Searching by tags allow you to search for Notebooks by topical area or technique. For example, if you are interested in learning new techniques for tackling classification problems you might try a search with the tag "classification" (`tag:classification`); if you are interested in an analysis of police records maybe a search with "crime" (`tag:crime`) would do the trick.

Alternatively, you can achieve the same thing by visiting the related tag pages. For example, the crime and classification tags live at https://www.kaggle.com/tags/crime and https://www.kaggle.com/tags/classification, respectively.

Tag pages include a section listing the most popular pages with the given tag, making them a great way of searching for Notebooks by content.

Using the Notebook Editor

Kaggle Notebooks may be created and edited via the Notebook editor. On larger screens, the Notebook editor consists of three parts:

- An editing window
- A console
- A settings window

The Notebook editor allows you to write and execute both traditional Scripts (for code-only files ideal for batch execution or Rmarkdown scripts) and Notebooks (for interactive code and markdown editor ideal for narrative analyses, visualizations, and sharing work).

The main difference between Scripts and Notebooks is the editing pane and how you experience editing and executing code.

Editing

Whether you use Scripts or Notebooks might depend on your choice of language and what your use case is. R users tend to prefer the Scripts, while Python users prefer the Notebooks. For more on why that is, refer to the "Types of Notebooks" section. Scripts are also favored for making competition submissions where the code is the focus, whereas Notebooks are popular for sharing EDAs (exploratory data analysis), tutorials, and other share-worthy insights.

Both editing interfaces are organized around the concept of "Versions". This is a collection consisting of a Notebook version, the output it generates, and the associated metadata about the environment.

In the Script editor, the code you write is executed all at once, whenever you generate a new version. For finer-grained control, it's also possible to specifically execute only a single line or selection of lines of code.

Notebooks are built on Jupyter notebooks. Notebook Notebooks consist of individual cells, each of which may be a Markdown (text) cell or a code cell. Code can be run (and the resulting variables saved) by running individual code cells, and cells can be added or deleted from the notebook at any time.

Console

The console tab provides an alternative interface to the same Python or R container running in the Notebook. Commands you input into the console will not change the content of your version. However, any variables you create in the console will persist throughout the session (unless you delete them). Additionally, any code that you execute in the editor will also execute in the console pane.

Settings

In the expanded editor, the settings pane takes up the right side of the screen. In the compact editor (where you hide the settings pane), it is folded into tabs above the Editor tab. In either case the settings pane contains the following tabs:

There's a tab called “Data” that provides a way of adding or removing data from the Notebook.

There's a tab called the Settings. The Settings tab has settings for toggling Language, toggling Docker image selection, toggling Internet (which is on by default), and toggling an Accelerator between CPU (default), GPU, and TPU.

Language is the programming language the Notebook is authored in. You can use it to switch between R and Python in the notebook flavor, and between R, RMarkdown, and Python in the script flavor. For more details on the differences, see the “Types of Notebooks” section.

The Docker image section can be used to pin the R or Python environment used for the Notebook against a certain Docker container version. More information can be found in "The Notebook Environment" section.

Adding Data Sources

One of the advantages to using Notebooks as your data science workbench is that you can easily add data sources from thousands of publicly available Datasets or even upload your own. You can also use output files from another Notebook as a data source. You can add multiple data sources to your Notebook’s environment, allowing you to join together interesting datasets.

Datasets

Kaggle Datasets provides a rich mix of interesting datasets for any kind of data science project.

There are two ways of loading a Dataset in a Notebook. The first is to navigate to a chosen dataset’s landing page, then click on the “New Notebook” button. This will launch a new Notebook session with the dataset in question spun up and ready to go.

Alternatively, you may wish to add datasets after creating your Notebook. To do that, navigate to the “Data” pane in a Notebook editor and click the “Add Data” button. This will open a modal that lets you select Datasets to add to your Notebook.

Competitions

You can also add Competition data sources to your Notebook environment using the same steps as above.

The main difference is that you need to accept the rules for any Competition data sources you add to your Notebook. Whether you start a new Notebook from the “Notebooks” tab of a Competition or add a Competition data source from an existing Notebook editor, you’ll be prompted to read and accept the rules first.

You can mix Competitions and Datasets data sources in the same Notebook, but please be sure to abide by the rules of the specific Competition with respect to using external data sources. If you don’t, you risk consequences for rule-breaking in the Competition.

Notebooks

You will notice that there is a third option in the “Add Data” modal: Notebook Output Files.

Up to 20 GBs of output from a Notebook may be saved to disk in /kaggle/working. This data is saved automatically and you can then reuse that data in any future Notebook: just navigate to the “Data” pane in a Notebook editor, click on “Add Data”, click on the "Notebook Output Files" tab, find a Notebook of interest, and then click to add it to your current Notebook.

By chaining Notebooks as data sources in this way, it’s possible to build pipelines and generate more and better content than you could in a single notebook alone.

“Minimal LSTM + NB-SVM baseline ensemble”, written by Jeremy Howard, is one example of a great Notebook using this feature. Click on the “Data” tab to view the data sources he uses.

Collaborating on Notebooks

Notebooks collaboration is a powerful feature. It allows multiple users to co-own and edit a Notebook. For example, you can work with Competition teammates to iterate on a model or collaborate with classmates on a data science project.

Inviting Collaborators

From your Notebook editor or viewer, public or private, you may navigate to the 'Share' or 'Sharing' button in the Notebook’s menu to expose, among other settings, the Collaborators options. There, use the search box to find and add other users as Notebook collaborators.

If your Notebook is private, you may choose between giving Collaborators either viewing privileges (“Can view”) or editing privileges (“Can edit”). If your Notebook is public, Collaborators can only be added with editing privileges (“Can edit”), as anyone can view it already.

When you add a collaborator, they will receive a notification via email.

“Creating, Reading & Writing Data”, a Notebook from the Advanced Pandas Kaggle Learn track, is one example of great collaborative Notebook.

Collaborating on Datasets

Using Notebooks is a powerful way to work with your collaborators on Datasets, too.

Datasets created on Kaggle also have privacy settings, and these settings are distinct from the sharing settings on your Notebook meaning each can be shared with a different group of users. That is, your Notebook collaborators won’t automatically have the same access to any private Datasets as you unless they are explicitly invited to collaborate on the Dataset. Anyone has access to Datasets shared publicly.

To learn more about how to use Datasets collaboratively, read more here.

The Notebook Environment

Notebooks is more than just a code editor. It’s a versioned computational environment designed to make it easy to reproduce data science work. In the Notebooks IDE, you have access to an interactive session running in a Docker container with pre-installed packages, the ability to mount versioned data sources, customizable compute resources like GPUs, and more.

Notebook Versions and Containers

When you create a Notebook version using 'Save & Run All', you execute the Notebook from top to bottom in a separate session from your interactive session. Once it finishes, you will have generated a new Notebook version. A Notebook version is a snapshot of your work including your compiled code, log files, output files, data sources, and more. The latest Notebook version of your Notebook is what is shown to users in the Notebook viewer.

Every Notebook version you create is associated with a specific Docker image version as well. Docker is a containerization technology which provides an isolated environment in which to do your work. Docker specifies the contents of this environment including installed Python and R packages using what is known as an image. Every Notebook version you create is associated with a Docker image.

By default for new notebooks, this will be the latest version of the default Python or R images that we maintain at Kaggle. The contents of this image is publicly available on GitHub. You may view it at https://github.com/Kaggle/docker-rstats for the R container, or https://github.com/Kaggle/docker-python for the Python container.

Dockerfiles and Notebook Versions

Even if you are using one of the default Kaggle containers, the number, names, and versions of the packages that you're using are still a moving target as our team continually updates them to ensure the latest and greatest packages are available. We update the images about every two weeks, mainly to upgrade to the latest versions of the packages we provide but also occasionally to add or remove certain packages. You can subscribe to notifications when we release a new Docker image on GitHub.

It is also possible to pin a specific Docker image for use in a Notebook if there are multiple custom images available. This can be done by accessing the "Settings" tab in the Notebook editor. Next to "Environment", there is an option to select "Preferences." This opens a modal where you can select what your environment preference is between pinning to a specific image (the image version when your notebook was created) or always using the latest image. You can read more about these options here.

In order to ensure that your Notebooks remain reproducible, we publicly expose the Dockerfile defining the environment the Notebook version was created in. You may download the contents of that Dockerfile by visiting the "Execution Info" section on your Notebook and navigating to the "Container image" field.

Modifying the Default Environment

You can request a modification to the default environment by submitting a pull request or an issue to the R or Python container on GitHub. Be sure to explain why you think a package should be added to the default environment. We welcome pull requests and engagement with our public images if users believe there are new packages that will be helpful and used by a significant majority of our users.

More rarely, if you notice that something in our default environments broke, you may notify us of it using the same mechanism.

Note that, even if approved, it can take several days for requested packages to be added to the live container image on the website.

Modifying a Notebook-specific Environment

It is also possible to modify the Docker container associated with the current Notebook image.

Using a standard package installer

In the Notebook Editor, make sure "Internet" is enabled in the Settings pane (it will be by default if it's a new notebook).

For Python, you can run arbitrary shell commands by prepending ! to a code cell. For instance, to install a new package using pip, run `!pip install my-new-package`. You can also upgrade or downgrade an existing package by running `!pip install my-existing-package==X.Y.Z`.

To install packages from GitHub in R, load the devtools package by running `library(devtools)`. Then, you can run commands such as `install_github("some_user/some_package")` to install a new package from GitHub.

Adding a free GPU

You can add a single NVIDIA Tesla P100 to your Notebook for free. GPU environments have lower CPU and main memory, but are a great way to achieve significant speed-ups for certain types of work like training neural networks on image data. One of the major benefits to using Notebooks as opposed to a local machine or your own VM is that the Notebook environment is already pre-configured with GPU-ready software and packages which can be time consuming and frustrating to set-up. Free GPU availability is limited: in busy times, you might be placed in a queue.

To add a GPU, navigate to the “Settings” pane from the Notebook editor and click the “Accelerator" > GPU option. Your session will restart which may take a few moments to several minutes if you don’t need to wait in a queue to access a GPU-enabled machine.

To learn more about getting the most out of using a GPU in Notebooks, check out this tutorial Notebook by Dan Becker.

Adding a free TPU

You can add a TPU v3-8 to your Notebook for free. TPUs are hardware accelerators specialized in deep learning tasks. They are supported in Tensorflow 2.1 both through the Keras high-level API and, at a lower level, in models using a custom training loop. Free TPU availability is limited: in busy times, you might be placed in a queue. To learn more about getting the most out of using a TPU in Notebooks, check out this in depth guide.

To add a TPU, navigate to the “Settings” pane from the Notebook editor and click the “Accelerator" > TPU v3-8 option. Your session will restart which may take a few moments to several minutes if you don’t need to wait in a queue to access a TPU-enabled machine.

Connecting Kaggle Notebooks to Google Cloud Services

Some of these services incur charges to attached GCP accounts. Please review pricing for each of the following products before you begin to use them in your notebook.

Kaggle currently has integrations with the Google Cloud Storage, BigQuery, and AutoML products. To enable these integrations, click on the “Add-ons” menu in the notebook editor and select “Google Cloud Services”. Once on the “Google Cloud Services” page you will need to attach your account to your notebook and you will need to select which of the integrations you want to enable. After enabling these integrations, you will be provided with a code snippet that can be copied and pasted into your notebook.

Each line of this code snippet corresponds to a different Google Cloud Services Integration where `PROJECT_ID` should be an existing Google Cloud Project. Per AutoML docs (linked below), AutoML currently requires that the location (`COMPUTE_REGION`) must be `us-central1` for your GCS Bucket.

For more information on how to use these services, please refer to Google Cloud Documentation or any of the specific product documentation.

BigQuery

- **BQ Documentation, BQML Documentation**

Google BigQuery is a fully managed, petabyte scale, low cost analytics data warehouse. There is no management required for users—instead, users can focus solely on analyzing data through queries and BigQuery ML to find meaningful insights in a pay-as-you-go billing model.

Google BigQuery can be accessed using Kaggle’s free-tier account to query public data but requires a billing-enabled GCP account to query any data that isn’t publicly released by BigQuery. You should carefully review the prices of BigQuery before trying the integration in Kaggle Notebooks, as it can be easy to incur charges.

```
# Set your own project id here
PROJECT_ID = 'your-google-cloud-project'
from google.cloud import bigquery
bigquery_client = bigquery.Client(project=PROJECT_ID)
```
For a more in-depth walkthrough of using the integration, please refer to the following notebooks:

- BigQuery in Kaggle Notebooks
- BigQuery Machine Learning Tutorial

Google Cloud Storage (GCS)

- **GCS Documentation**

Google Cloud Storage allows for storage and retrieval of data at any time across the globe. Users are able to use the storage space for any type of data and only pay for used storage space (per GB per month).

Google Cloud Storage is a paid service and requires a billing-enabled GCP account. You should carefully review the prices of GCS before trying the integration in Kaggle Notebooks, as it can be easy to incur charges.

```
# Set your own project id here
PROJECT_ID = 'your-google-cloud-project'
from google.cloud import storage
storage_client = storage.Client(project=PROJECT_ID)
```
For a more in-depth walkthrough of using the integration, please refer to the following notebooks:

- Moving Data to/from GCS

AutoML

- **AutoML Documentation**

Google AutoML is a suite of products that enables users to train custom machine learning models for tasks on structured data, vision and language. It is currently in Beta, so you may encounter usability frictions or known issues. We welcome all feedback from the community. User feedback will help us improve documentation and be shared directly with the AutoML team to help improve the product.

Google AutoML is a paid service and requires a billing-enabled GCP account. You should carefully review the prices of AutoML before trying the integration in Kaggle Notebooks, as it can be easy to incur charges. You can see the pricing for each of the offerings in beta here:

- AutoML Tables Pricing
- AutoML Vision Pricing
- AutoML Natural Language Pricing

```
# Set your own project id and compute region here
PROJECT_ID = 'your-google-cloud-project'
COMPUTE_REGION = 'us-central1' # must be `us-central1` to use AutoML (see
docs)
from google.cloud import automl_v1beta1 as automl
automl_client = automl.AutoMlClient()
project_location = automl_client.location_path(PROJECT_ID, COMPUTE_REGION)
```
For a more in-depth walkthrough of using the integration, please refer to the following notebooks:

- AutoML Tables Tutorial

Google Cloud AI Notebooks

If you run into compute constraints while using notebooks on Kaggle, you can consider upgrading to Google Cloud AI Notebooks. These notebooks run under your project in Google Cloud, and can be configured to use your choice of virtual machine, accelerators and run without limits

To export your notebook to Google Cloud, you can go to the **File** menu and select "Upgrade to Google Cloud AI Notebooks" from within the Notebooks Editor. You can also upgrade a notebook from the Viewer by clicking on the three-dot menu on the top right.

For a more detailed description of how to export your Kaggle Notebooks to Google Cloud AI Notebooks, check out the announcement post here:

- [Feature Launch] Upgrade to Notebooks on Google Cloud for more compute!

Technical Specifications

Kaggle Notebooks run in a remote computational environment. We provide the hardware—you need only worry about the code.

At time of writing, each Notebook editing session is provided with the following resources:

- 12 hours execution time for CPU and GPU notebook sessions and 9 hours for TPU notebook sessions
- 20 Gigabytes of auto-saved disk space (/kaggle/working)
- Additional scratchpad disk space (outside /kaggle/working) that will not be saved outside of the current session

CPU Specifications

- 4 CPU cores
- 30 Gigabytes of RAM

P100 GPU Specifications

- 1 Nvidia Tesla P100 GPU
- 4 CPU cores
- 29 Gigabytes of RAM

T4 ×2 GPU Specifications

- 2 Nvidia Tesla T4 GPUs
- 4 CPU cores
- 29 Gigabytes of RAM

TPU 1VM Specifications

- 96 CPU cores
- 330 Gigabytes of RAM

NOTE: CPU Platforms (ex. Intel Skylake, Broadwell, AMD) may be variable during regular notebook runs, however submissions runs (for code competitions or when submissions are rerun in bulk) are always run on Intel Skylake CPUs.

CPU Specifications

While editing a Notebook, you are provided with 20 minutes of idle time for your interactive session. If the code is not modified or executed in that time the current interactive session will end. If this happens, you will need to click the Edit button again to continue editing. If you want to run a computation that takes longer, you can Save a Version of your Notebook from top to bottom by selecting the "Save & Run All" option in the "Save Version" menu (see below).

Once you are satisfied with the contents of the Notebook you can click "Save Version" to save your changes. From there you will have two options for creating a new version:

- **Quick Save** skips the top-to-bottom notebook execution and just takes a snapshot of your notebook exactly as it’s displayed in the editor. This is a great option for taking a bunch of versions while you’re still actively experimenting. Quick Save is a brand new way of saving work on Kaggle.

- **Save & Run All** creates a new session with a completely clean state and runs your notebook from top to bottom. This is perfect for major milestones or when you want to share your work, as it gives you (and anyone else who reads your notebook) the confidence that your notebook can be run reproducibly. In order to save successfully, the entire Notebook must execute within 12 hours (9 hours for TPU notebooks). Save & Run All is identical to the “Commit” behavior you may have used previously on Kaggle.