Public API

Create Datasets, Notebooks, and connect with Kaggle

Getting Started: Installation & Authentication

The easiest way to interact with Kaggle's public API is via our command-line tool (CLI) implemented in Python. This section covers installation of the kaggle package and authentication.

Installation

Ensure you have Python and the package manager pip installed. Run the following command to access the Kaggle API using the command line: `pip install kaggle` (You may need to do pip install --user kaggle on Mac/Linux. This is recommended if problems come up during the installation process.) Follow the authentication steps below and you'll be able to use the `kaggle` CLI tool.

If you run into a kaggle: command not found error, ensure that your python binaries are on your path. You can see where kaggle is installed by doing pip uninstall kaggle and seeing where the binary is. For a local user install

on Linux, the default location is ~/.local/bin. On Windows, the default location is $PYTHON_HOME/Scripts.

Authentication

In order to use the Kaggle's public API, you must first authenticate using an API token. Go to the 'Account' tab of your user profile and select 'Create New Token'. This will trigger the download of kaggle.json, a file containing your API credentials.

If you are using the Kaggle CLI tool, the tool will look for this token at ~/.kaggle/kaggle.json on Linux, OSX, and other UNIX-based operating systems, and at C:\Users\<Windows-username>\.kaggle\kaggle.json on Windows. If the token is not there, an error will be raised. Hence, once you've downloaded the token, you should move it from your Downloads folder to this folder.

If you are using the Kaggle API directly, where you keep the token doesn't matter, so long as you are able to provide your credentials at runtime.

Interacting with Competitions

The Kaggle API and CLI tool provide easy ways to interact with Competitions on Kaggle. The commands available can make participating in competitions a seamless part of your model building workflow.

If you haven't installed the package needed to use the command line tool or generated an API token, check out the getting started steps first.

Just like participating in a Competition normally through the user interface, you must read and accept the rules in order to download data or make submissions. You cannot accept Competition rules via the API. You must do this by visiting the Kaggle website and accepting the rules there.

Some of the commands for interacting with Competitions via CLI include:

- `kaggle competitions list`: list the currently active competitions
- `kaggle competitions download -c [COMPETITION]`: download files associated with a competition
- `kaggle competitions submit -c [COMPETITION] -f [FILE] -m [MESSAGE]`: make a competition submission

View all available commands on the official documentation on GitHub and keep up-to-date with the latest features and bug fixes in the changelog.

To explore additional CLI arguments, remember that you can always append `-h` after any call to see the help menu for that command.

Submitting to a Competition

Assuming that you have already accepted the terms of a Competition (this can only be done through the website, and not through the CLI), you may use the Kaggle CLI to submit predictions to the Competition and have them scored. To do so, run the command `kaggle competitions submit -c [COMPETITION NAME] -f [FILE PATH]`.

You can list all previous submission to a Competition you have entered using the command `kaggle competitions submissions -c [COMPETITION NAME]`.

To explore some further CLI arguments, remember that you can always append `-h` after any call to see the help menu for that command.

Interacting with Datasets

The Kaggle API and CLI tool provide easy ways to interact with Datasets on Kaggle. The commands available can make searching for and downloading Kaggle Datasets a seamless part of your data science workflow.

If you haven't installed the Kaggle Python package needed to use the command line tool or generated an API token, check out the getting started steps first.

Some of the commands for interacting with Datasets via CLI include:

- `kaggle datasets list -s [KEYWORD]`: list datasets matching a search term
- `kaggle datasets download -d [DATASET]`: download files associated with a dataset

If you are creating or updating a dataset on Kaggle, you can also use the API to make maintenance convenient or even programmatic. View all available commands on the official documentation on GitHub and keep up-to-date with the latest features and bug fixes in the changelog.

To explore additional CLI arguments, remember that you can always append `-h` after any call to see the help menu for that command.

Other than the Kaggle API, there is also a Kaggle connector on DataStudio! You can select Kaggle Datasets as a data source to import directly into DataStudio. Work in DataStudio to easily create beautiful and effective dashboards on Kaggle Datasets!

Creating and Maintaining Datasets

The Kaggle API can be used to to create new Datasets and Dataset versions on Kaggle from the comfort of the command-line. This can make sharing data and projects on Kaggle a simple part of your workflow. You can even use the API plus a tool like crontab to schedule programmatic updates of your Datasets to keep them well maintained.

If you haven't installed the Kaggle Python package needed to use the command line tool or generated an API token, check out the getting started steps first.

Create a New Dataset

Here are the steps you can follow to create a new dataset on Kaggle:

- Create a folder containing the files you want to upload
- Run `kaggle datasets init -p /path/to/dataset` to generate a metadata file
- Add your dataset's metadata to the generated file, `datapackage.json`

- Run `kaggle datasets create -p /path/to/dataset` to create the dataset

  Your dataset will be private by default. You can also add a `-u` flag to make it public when you create it, or navigate to “Settings” > “Sharing” from your dataset’s page to make it public or share with collaborators.

Create a New Dataset Version

If you’d like to upload a new version of an existing dataset, follow these steps:

- Run `kaggle datasets init -p /path/to/dataset` to generate a metadata file (if you don’t already have one)
- Make sure the `id` field in `dataset-metadata.json` (or `datapackage.json`) points to your dataset
- Run `kaggle datasets version -p /path/to/dataset -m "Your message here"`

  These instructions are the basic commands required to get started with creating and updating Datasets on Kaggle. You can find out more details from the official documentation on GitHub:

- Initializing metadata
- Create a Dataset
- Update a Dataset

Working with Dataset Metadata

If you want a faster way to complete the required `dataset-metadata.json` file (for example, if you want to add column-level descriptions for many tabular data files), we recommend using Frictionless Data's Data Package Creator. Simply upload the `dataset-metadata.json` file that you've initialized for your dataset, fill out metadata in the user interface, and download the result.

To explore some further CLI arguments, remember that you can always append `-h` after any call to see the help menu for that command.

Interacting with Notebooks

The Kaggle API and CLI tool provide easy ways to interact with Notebooks on Kaggle. The commands available enable both searching for and downloading published Notebooks and their metadata as well as workflows for creating and running Notebooks using computational resources on Kaggle.

If you haven't installed the Kaggle Python package needed to use the command line tool or generated an API token, check out the getting started steps first.

Some of the commands for interacting with Notebooks via CLI include:

- `kaggle kernels list -s [KEYWORD]`: list Notebooks matching a search term
- `kaggle kernels push -k [KERNEL] -p /path/to/folder` : create and run a Notebook on Kaggle
- `kaggle kernels pull [KERNEL] -p /path/to/download -m`: download code files and metadata associated with a Notebook

  If you are creating a new Notebook or running a new version of an existing Notebook on Kaggle, you can also use the API to make this workflow convenient or even programmatic. View all available commands on the official documentation on GitHub and keep up-to-date with the latest features and bug fixes in the changelog.

  To explore additional CLI arguments, remember that you can always append `-h` after any call to see the help menu for that command.

Creating and Running a New Notebook

The Kaggle API can be used to to create new Notebooks and Notebook versions on Kaggle from the comfort of the command-line. This can make executing and sharing code on Kaggle a simple part of your workflow.

If you haven't installed the Kaggle Python package needed to use the command line tool or generated an API token, check out the getting started steps first.

Here are the steps you can follow to create and run a new Notebook on Kaggle:

- Create a local folder containing the code files you want to upload (e.g., your Python or R notebooks, scripts, or RMarkdown files)
- Run `kaggle kernels init -p /path/to/folder` to generate a metadata file
- Add your Notebook's metadata to the generated file, `kernel-metadata.json`; As you add your title and slug, please be aware that Notebook titles and slugs are linked to each other. A Notebook slug is always the title lowercased with dashes (-) replacing spaces and removing special characters.
- Run `kaggle kernels push -p /path/to/folder` to create and run the Notebook on Kaggle

  Your Notebook will be private by default unless you set it to public in the metadata file. You can also navigate to "Options" > “Sharing” from your published Notebook's page to make it public or share with collaborators.

Creating and Running a New Notebook Version

If you’d like to create and run a new version of an existing Notebook, follow these steps:

- Run `kaggle kernels pull [KERNEL] -p /path/to/download -m` to download your Notebook's most recent code and metadata files (if you your local copies aren't current)
- Make sure the `id` field in `kernel-metadata.json` points to your Notebook; you no longer need to include the `title` field which is optional for Notebook versions unless you want to rename your Notebook (make sure to update the `id` field in your next push AFTER the rename is complete)

- Run `kaggle kernels push -p /path/to/folder`

  These instructions are the basic commands required to get started with creating, running, and updating Notebooks on Kaggle. You can find out more details from the official documentation on GitHub:

- Initializing metadata
- Push a Notebook
- Pull a Notebook
- Retrieve a Notebook's output

Using Models in Notebooks

Models can be downloaded via notebooks using the following code:

```
import kagglehub

# Authenticate
kagglehub.login() # This will prompt you for your credentials.
# We also offer other ways to authenticate (credential file & env variables): https://github.com/Kaggle/kagglehub?tab=readme-ov-file#authenticate

# Download latest version
path = kagglehub.model_download("google/gemma/pyTorch/2b")
```

```
    # Download specific version (here version 1)
    path =
kagglehub.model_download("google/gemma/pyTorch/2b/1")

    print("Path to model files:", path)
```

Models can be uploaded via notebooks using the following code:

```
   import kagglehub
    from kagglehub.config import
get_kaggle_credentials

    # Other ways to authenticate also available:
https://github.com/Kaggle/kagglehub?tab=readme-ov-
file#authenticate
    kagglehub.login()

    username, _ = get_kaggle_credentials()

    # For PyTorch framework & `2b` variation.
    # Replace the framework with "jax", "other" based
on which framework you are uploading to.

kagglehub.model_upload(f'{username}/my_model/pyTorch/
2b', 'path/to/local/model/files', 'Apache 2.0')

    # Run the same command again to upload a new
```

```
version for an existing variation.
```

See kagglehub documentation. for more information.