DFFML: Enhancing User Experience with Notebook Examples for Machine Learning Use Cases.

About me

Chaudhry Mohammad Hashim

Github: mHash1m

Gitter: mHash1m

Education:

Institute: Iqra University - Islamabad.

Program: Bachelors in Computer Science.

Year: 4th.

Expected Graduation: Dec 2021

Contact Info:

Email: hashimchaudry23@gmail.com

Time Zone:

PKT (GMT+5)

LinkedIn:

https://www.linkedin.com/in/hashim-chaudry-770203191

I am a Data Science / Machine Learning enthusiast residing in Pakistan. Like most of the developing countries, state-of-the-art technologies don't quite bloom here so it is always a challenge to pursue cutting-edge careers. I do my best to brim over the gaps with online courses and self-learning alongside my University studies to keep up with the latest. I have worked on several data science projects and have experience working and collaborating in open-source communities. I hope to learn a lot more and make a significant impact through this amazing opportunity ie. GSoC. Ultimately, I would like to use the experience to help people around me and give back to the community.

Code Contribution

I started contributing to Dffml last year. I have several contributions from last year as well, so for the ease of mentors, I've filtered out a list of all my Pull Requests to date, right here.

To enlist a few:

I. record: Separate confidence from prediction **(OPEN)**

II. docs: tutorials: use-cases: Moving bw Models **(OPEN)**

III. model: wrap daal4py LR **(MERGED)**

IV. docs: model: tensorflow_hub: Add example for text classifier **(MERGED)**

V. model: make model plugins import third party libraries dynamically. **(MERGED)**

VI. tests: high_level: Added tests for load and save functions. **(MERGED)**

Project Information

Org Name:

Python Software Foundation (PSF)

Sub-org Name:

Data Flow Facilitator for Machine Learning (DFFML)

Project Abstract:

The project aims to develop Jupyter notebooks for use case examples that explain the Machine Learning workflow of DFFML API.

Detailed Description:

DFFML curates several models and provides its users with a simple way to use Machine Learning. However, it can be hectic or time-consuming especially for new users to discover all the use-cases of an API. For users to be able to fully understand how they can integrate DFFML into their routine machine learning tasks or the state-of-the-art projects, they need to see the potential DFFML has and how it can make their jobs a lot easier!

It is, in fact, one of the hardest jobs to understand another person's code, let alone figure out what that code could possibly help you achieve. However, DFFML's use-cases and functionality captured with runnable code examples as notebooks can really improve user experience.

The project will be implemented using **Jupyter notebooks.** DFFML hasn't integrated any notebooks in the codebase yet. Jupyter notebooks allow documentation right along with the code in a very presentable way. It also allows selective cell-level code execution which can help users understand the flow better. These notebooks will be designed to be beginner-friendly. The code will be written with new users, and their possible use cases, in mind to maximize the ctrl +c and ctrl +v prowess 😉. Moreover, the code will contain appropriate descriptions so that it minimizes the learning curve for new users and helps them be productive in a relatively shorter time. The project will also involve testing the Jupyter notebooks using the testing framework `nteract/testbook`.

A library's success is dependent upon its usability and I believe user experience is something that can always be improved. I would love to contribute to making DFFML better for its users.

"The next big thing is the one that makes the last big thing usable."

— Blake Ross, Co-creator of Mozilla Firefox

Project Scope:

This project will basically comprise two phases.

Phase 1. Adding General Machine Learning Use-Case Examples:

i. Comparison of model performance.

ii. Tuning models

iii. Ensemble models.

iv. Saving weights and using a pre-trained model.

v. Transfer Learning.

For code demonstration of Phase 1 (Adding General Machine Learning Use-Case Examples), I've opened up a PR to create a notebook demonstrating a similar use-case, which is awaiting other PR's to build on and expected to be complete soon.

Phase 2. Adding Multi-Output Models support and Use-cases:

i. Add support for Multi-Output models.

This task will be achieved by wrapping the sklearn.multioutput sub-modules.

More specifically, I'll be wrapping the `multioutput.MultiOutputClassifier` and `multioutput.MultiOutputRegressor`.

sklearn provides a range of different regression and classification models that can in turn be passed on as a parameter to these modules to build a multi-output model.

ii. Use-case example for multi-output regression models.

This part of phase 2 will be focused on building an example usage of the `MultiOutputRegressor` module we will have wrapped earlier. The following example from the documentation of sklearn shows how to build a `MultiOutputRegressor` model out of a simple Ridge Regression model to handle multi-output data.

```
>>> import numpy as np
>>> from sklearn.datasets import load_linnerud
>>> from sklearn.multioutput import MultiOutputRegressor
>>> from sklearn.linear_model import Ridge
>>> X, y = load_linnerud(return_X_y=True)
>>> clf = MultiOutputRegressor(Ridge(random_state=123)).fit(X, y)
>>> clf.predict(X[[0]])
array([[176..., 35..., 57...]])
```

iii. Use-case example for multi-output classification models.

This part of phase 2 will be focused on building an example usage of the `` `MultiOutputClassifier` `` module we wrapped earlier.
The following example from the documentation of sklearn shows how to build a `` `MultiOutputClassifier` `` model out of a simple KNeighbors Classifier model to handle multi-output data.

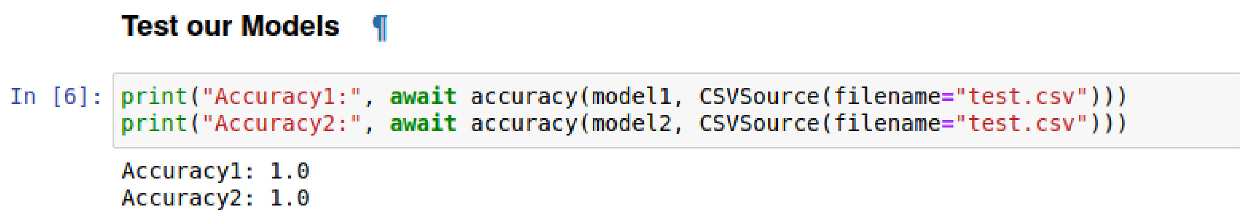
```
>>> import numpy as np
>>> from sklearn.datasets import make_multilabel_classification
>>> from sklearn.multioutput import MultiOutputClassifier
>>> from sklearn.neighbors import KNeighborsClassifier
>>> X, y = make_multilabel_classification(n_classes=3,
random_state=0)
>>> clf = MultiOutputClassifier(KNeighborsClassifier()).fit(X, y)
>>> clf.predict(X[-2:])
array([[1, 1, 0], [1, 1, 1]])
```

Testing:

Code testing will be carried out simultaneously during each phase.

Testing the notebooks will involve the usage of `nteract/testbook`, a unit testing framework for testing code in Jupyter Notebooks. Testbook makes it so that you don't have to extract any code from the notebook manually. You can simply assert `STDOUT` of a cell in the notebook you want to test. The following example demonstrates the usage:

The Notebook Cell to be tested:



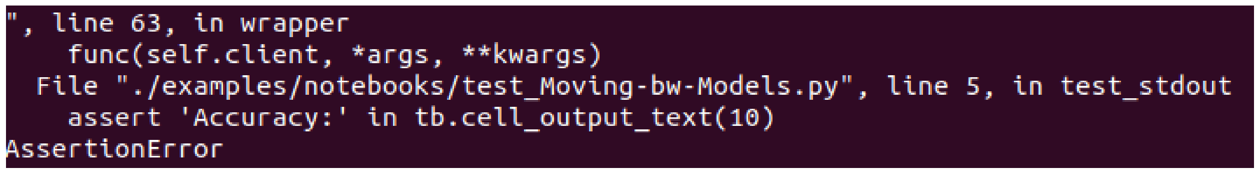
Test our Models ¶

```
In [6]: print("Accuracy1:", await accuracy(model1, CSVSource(filename="test.csv")))
        print("Accuracy2:", await accuracy(model2, CSVSource(filename="test.csv")))

        Accuracy1: 1.0
        Accuracy2: 1.0
```

A basic example of false assert:

```
from testbook import testbook


@testbook('examples/notebooks/Moving-bw-Models.ipynb', execute=True)

def test_stdout(tb):

   assert 'Accuracy:' in tb.cell_output_text(10)

test_stdout()
```

Test output of false assert:

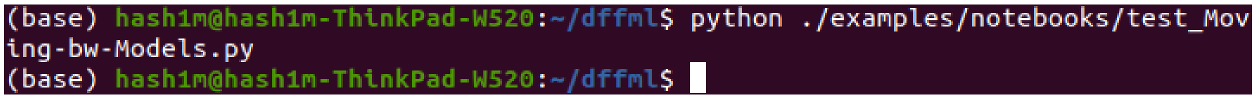


```
", line 63, in wrapper
    func(self.client, *args, **kwargs)
  File "./examples/notebooks/test_Moving-bw-Models.py", line 5, in test_stdout
    assert 'Accuracy:' in tb.cell_output_text(10)
AssertionError
```

A Basic example of true assert:

```
from testbook import testbook

@testbook('examples/notebooks/Moving-bw-Models.ipynb', execute=True)

def test_stdout(tb):

   assert 'Accuracy1:' in tb.cell_output_text(10)

   assert 'Accuracy2:' in tb.cell_output_text(10)

test_stdout()
```

Test output of true assert(no output):



```
(base) hash1m@hash1m-ThinkPad-W520:~/dffml$ python ./examples/notebooks/test_Mov
ing-bw-Models.py
(base) hash1m@hash1m-ThinkPad-W520:~/dffml$ 
```

As we can see, `testbook` makes it quite simple to test out any notebooks by enabling cell-level testing.

Documentation:

Notebooks are quite presentable on their own, but the idea of presenting a link to a notebook is not so much. To overcome this, I'll be integrating the notebooks into DFFML documentation which will ultimately improve the discoverability of the notebooks.

To carry out the task, we will be using `nbsphinx`, a Sphinx extension that provides a source parser for *.ipynb files. The process is as follows:

We edit the `conf.py` file and add 'nbsphinx' to the extensions for Sphinx.
After that, we can simply go ahead and edit the `.rst` file we want the notebook(s) to be displayed in and add the name(s) of the *.ipynb file(s) to the `toctree`.

Following is a demonstration of adding the `.ipynb` files to the toctree:

```
.. toctree::
    :maxdepth: 1
    dffml/examples/notebooks/example.ipynb
```

Stretch Goals:

The stretch goals of the project aim to add more code demos to the DFFML documentation. The following are the possible demos I'll start working on after the main project goals:

Demos:

i. Demo Code for the MNIST Handwritten Digits DFFML CLI example.

ii. Demo Code for the Flower17 Species Classification DFFML CLI example.

iii. Demo Code Example around the Heart Disease UCI dataset.

iv. Demo Code Example around the Wine Quality Dataset.

v. Demo Code Example around the Perfume Dataset.

Project Timeline(Tentative):

Note: This is a tentative timeline as the deliverables each week have different time requirements. It only highlights and doesn't restrict the pace of the project. Each week of the timeline also accounts for any sort of study required to carry out the tasks.

- **Community Bonding (May 17 - June 6):**
    - Further exploring DFFML and its possible use-cases.
    - Studying the code-base and its architecture more thoroughly to get a better understanding of how to approach the project, and how to achieve the project goals.
    - Working on DFFML issues open in Github repository.

Phase 1: Adding General Machine Learning Use-Case Examples

- **Week 1 (June 7 - 11):**
    - Develop a use-case example for "Comparison of model performance with visualizations".
    - Test the example notebook.

- **Week 2 (June 14):**
    - Develop the use-case example for “Tuning models”.
    - Test the example notebook.

- **Week 3 (June 21):**
    - Develop the use-case example for “Ensemble models”.
    - Test the example notebook.

- **Week 4 (June 28):**
    - Develop the use-case example for “Saving weights and using a pre-trained model”.
    - Test the example notebook.

- **Week 5 - 6 (July 5 - 16):**
    - Develop the use-case example for “Transfer Learning”.
    - Test the example notebook.
    - Documentation of Phase 1.
    - Wrap up Phase 1.

Phase 2: Adding Multi-Output Models support and Use-cases

- **Week 7 (July 19):**
    - Wrap Multi-Output Classifier.
    - Test Multi-Output Classifier.

- **Week 8 (July 26):**
    - Wrap Multi-Output Regressor.
    - Test Multi-Output Regressor.

- **Week 9 (August 2):**
  - Develop the use-case example for a multi-output classification model.
  - Test the example notebook.

- **Week 10 (August 9):**
  - Develop the use-case example for a multi-output regression model.
  - Test the example notebook.

- **The final week (August 16):**
  - Wrap up the whole project.
  - Submit the project.

Other Commitments

I don’t currently have any commitments that might hinder my work schedule. Although, due to the COVID-19 pandemic, it's hard to say anything about the University schedule and when the final exams will be held. Even so, I’ll let the mentors know in advance whenever my Final Exams are scheduled. In case we have a conflict with examinations, I’ll do my best to adapt accordingly and commit at least 12 hours a week during my exams. I’ll cover up for lost time around the exams to achieve the goals associated with the project.

Are you applying for other projects in GSoC?

Yes, I'll be applying to another organization outside the PSF umbrella. However, this doesn't affect my commitment to DFFML in any way as it is still number one on my preference list. As a matter of fact, looking into other projects helped me come up with use-cases for this one 😁.

Future Enhancements

As a future enhancement to this project, perhaps sklearn's Chain algorithms could also be wrapped to have a variety in multi-output usage. I've also curated an extensive list of stretch goals that can be worked on later in succession to this project in case all of them are not covered under the project.
