DFFML: Support Archive Storage for Models

About Me:

Name	Saahil Ali
	Github — programmer290399 **Gitter** — @programmer290399 **LinkedIn** — Profile_Link
University	International Institute of Professional Studies, DAVV, University, Indore, Madhya Pradesh
	Program — M.Tech(IT) 5 yr **Year** — 3rd **Expected Graduation** — 2023
Contact Information	**Email** **Contact no.**
Time zone	IST (GMT +5:30)
Resume	Link

Code Contribution:

1. util : net : Download progress feature #1039 :

Related Issue	#563
Status	Merged

2. util: net: cached_download(): Changed logging mechanism to log only on 1% changes #1060 :

Related Issue	#1058
Status	Merged

3. service: dev: docs: build: Docs.sh port to dev.py cmd #1054:

Related Issue	#1017
Status	Merged

4. WIP : ci : lint : commits : Adding ci job to validate commit message format #1076:

Related Issue	#1040
Status	Received review, working on requested changes.

5. WIP: model : orion : Add Orion time-series anomaly detection model #1045:

Related Issue	#960
Status	Put on hold due to 2nd party plugin issue #1050

Project Information:

1. **Organisation:** Python Software Foundation

2. **Sub-org Name:** DFFML

3. **Project Abstract:**

Adding support for archive storage (i.e in the form of .zip or .tar.* family) of models for all pre-existing models in DFFML and update tests, documentation and fix any model specific bugs that come along the way while implementing this feature.
This will not only save the model state but also all the configuration of a model.
There would be two benefits of this implementation to the user:

1. Increased Reproducibility of DFFML models.
2. Better Portability of DFFML models.

4. **Detailed Description:**

DFFML provides a unified interface for training, testing, and deploying various machine learning models. It allows users to write their own plugins for fine-grained control on various aspects of a pipeline like reading data or training a model while providing a framework agnostic interface which helps put various parts together in a pipeline without affecting or breaking other pre-existing parts. This makes DFFML based projects easy to maintain, update and extend.

At the time of writing this proposal, issue #662 is open, which states that currently we have no support for saving Models as a single archived file which holds all the necessary information to restore the working state of the Model, which will enable a user to resume/reproduce work at a different point in time or on another machine/environment.

Following the preference of the project of avoiding the use of external dependencies I would use the Python's Data Compression and Archiving modules available in the standard library for this project. I would primarily focus on adding support for zip files and tarfiles as these two almost cover the types of archiving formats used. If time allows I would extend support for lesser known/used formats.

With that out of the way, I would like to break down this project into three sub-parts, all the steps are going to include documenting any relevant changes:

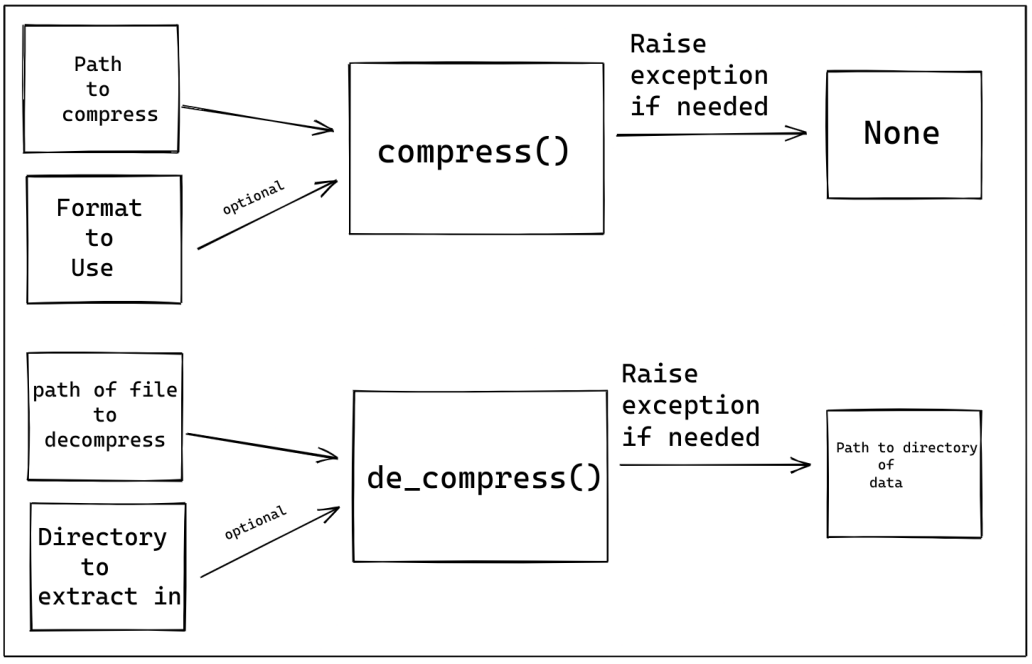
1. Implementing a complete utility module under *dffml/util* to to extract and create archived files, something like this but with more functions (as discussed later).

2. Using the archiving module implemented in step 1, edit the relevant base classes to use this module to automatically extract /archive model related data if the location attribute (which is currently directory) has an extension that we are planning to support in this project.

3. Implementing new test cases for all models and updating the pre-existing code to work with the changes made in step 2, also making sure that the existing tests pass and fixing any bugs that might be introduced in this process.

STEP 1 : Implement Archive Handler:

Archive handler module:



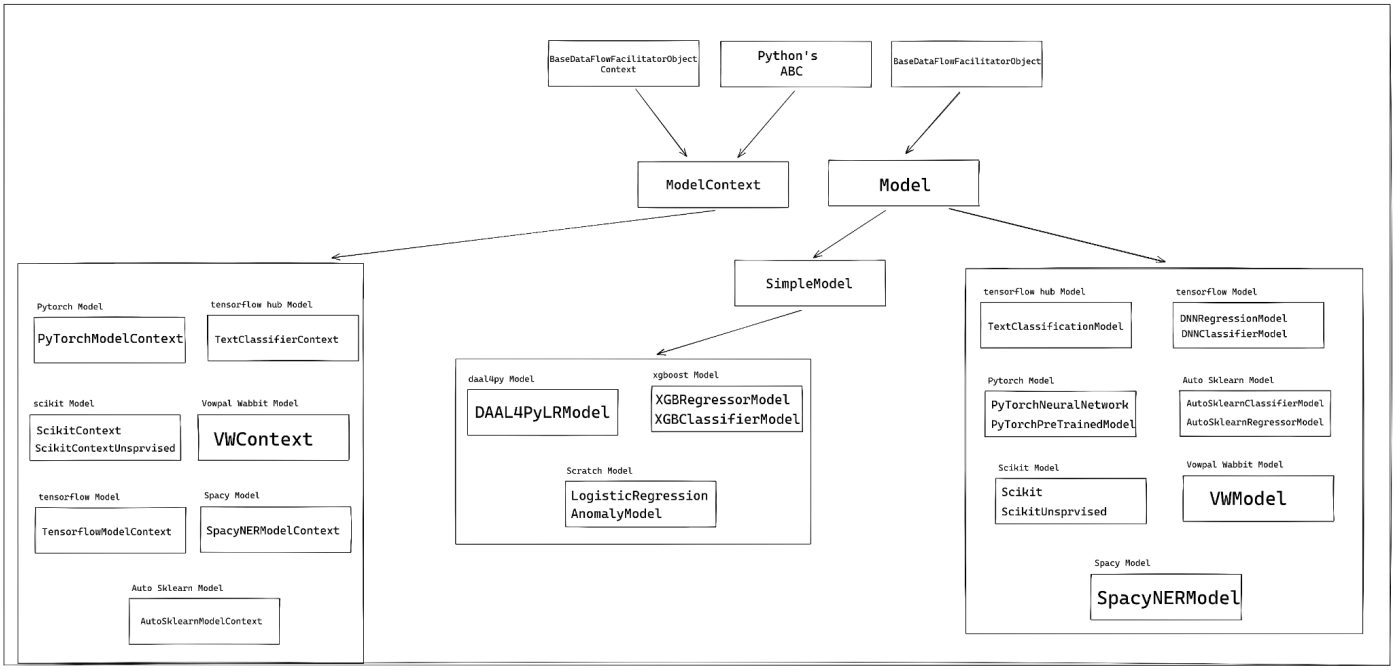
- The archive handler module would mainly consist of two functions one to compress data and the other to decompress data.
- Other helper functions might be needed in order to implement these functions, and we would cross that bridge when we'd come to it.
- This is just to give a rough idea of the implementations and these are subject to change if needed.

STEP 2 :Updating Model Base Classes:

- All models either inherit from *Model* Class or *SimpleModel* class
- *SimpleModel* class inherits from *Model* class ,thus implementation of the archive support should be ideally done in the *Model* class
- And this should be done as stated in Issue #662

STEP 3 : Updating all the Models and their Test Cases:

Inheritance Tree



- The image above shows the inheritance tree of models in DFFML, (it may not be clear so you can download/view the image here)
- The following models and their respective tests and documentation would be updated:

No.	Model Name		
1	Slr model	10	Vowpal Wabbit model
2	Auto Sklearn model	11	Xgboost model
3	Daal4py model		
4	Pytorch model		
5	Scikit model		
6	Scratch model		
7	Spacy model		
8	Tensorflow model		

- **NOTE:** This step would be affected by status of issue #1050

5. Weekly Timeline:

- Community Bonding [May 17 - June 07]:
  - Discuss my plan in detail with the mentors and get feedback to improve my current plan.
  - Dive deeper into the code of all the models to see if there can be any roadblocks and how they can be avoided.
  - Finish the issue I am working on currently i.e #1040, so that I can focus on this plan better during the coding period.

- Week 1 [June 07]:
  - Start working on Step 1 and come up with a quick implementation.
  - Improve the implementation with input from the mentors.
  - Write tests and document the code as per need.

- Week 2 [June 14]:
  - Implement the extraction/compression features in the base class.
  - At this point a lot of things might start to break.
  - Update and implement new tests for the slr model.
  - Update the related documentation.

- Week 3 [June 21]:
  - Make sure Daal4py and Xgboost model are working with new changes.
  - Fix any bugs or make any required changes.
  - Update and implement new tests for both the models.
  - Update the related documentation.

- Week 4 [June 28]:
  - Make sure Scratch model and Vowpal Wabbit model are working with new changes.
  - Fix any bugs or make any required changes.
  - Update and implement new tests for both the models.
  - Update the related documentation.

- Week 5 [July 05]:
  - Make sure various PyTorch models are working with new changes.
  - Fix any bugs or make any required changes.
  - Update and implement new tests for all the models.
  - Update the related documentation.

- Week 6 [July 12]:
    - Make sure various Scikit models are working with new changes.
    - Fix any bugs or make any required changes.
    - Update and implement new tests for all the models.
    - Update the related documentation.

- Week 7 [July 19]:
    - Make sure various Tensorflow hub models are working with new changes.
    - Fix any bugs or make any required changes.
    - Update and implement new tests for all the models.
    - Update the related documentation.

- Week 8 [July 26]:
    - Make sure the Auto sklearn model and Spacy model are working with new changes.
    - Fix any bugs or make any required changes.
    - Update and implement new tests for both the models.
    - Update the related documentation.

- Week 9 [August 02]:
    - Make sure various Tensorflow models are working with new changes.
    - Fix any bugs or make any required changes.
    - Update and implement new tests for all the models.
    - Update the related documentation.

- Week 10 [August 09]:
    - Wrap up any remaining work and document any remaining parts
    - Make sure that the code coverage has not dropped and write more tests if it has dropped
    - Fix any new bugs that might have been introduced so far.
    - Prepare a final report and summarize my work

6. Stretch Goals:

- I plan to work on Local file based dataflow caching # 836
- I would like to implement following two models as well, because they are very commonly used in NLP use cases:
  - Fasttext models
  - Flair models

7. Other Commitments:

- College Examinations :
  - I would have 2 Mid-Semester and 1 End-Semester examinations during the GSoC period.
  - Each Mid-Semester exam would last for 3 days and would reduce my working hours by 2-3 hours during those 3 days.
  - The End-Semester examinations can last for up to a week and during that week my working hours would be upto 3-4 hours only.
  - The schedule for these examinations has not been declared by the university yet. I will update the mentors at-least a week before any exams.
- I am not applying to any other organizations this year.
- My current internship would end on 8th May , i.e. before the GSoC period starts so I would have no other commitments other than the exams mentioned above.