GSoC'2022 Project Proposal

Creating a model class for ease in development of PyMC models

Mentors: Thomas Wiecki, Micheal Osthege

Personal Details:

Name: Shashank Kirtania

Email: shashankkirtania123@gmail.com (primary), skirtania_be19@thapar.edu (secondary)

Telephone: +91 96545 78112

Country of Residence: India (UTC +05:30)

Primary Language: English

PyMC Discourse: 5hv5hvnk

Profiles: Github, LinkedIn

Project Proposal:

Introduction

Many users using PyMC face difficulty in deploying or saving their designed PyMC model because deploying/saving/loading a user-created model one of the reasons behind this is there is no direct way to save or load a model in PyMC like scikit-learn or tensorflow. To combat this, I propose to create a model class to improve workflow and use direct APIs to

build, fit, save, load and predict. Secondly, to improve the deployment workflow, I want to work on writing tutorials on how to deploy the above functions in commonly used deployment tools such as Docker, sagemaker, ML-Flow airflow and dask.

Why this project?

Having first encountered PyMC while working with a colleague on a project related to Bayesian modeling, initially, I found PyMC to be very helpful for working on projects related to probabilistic machine learning. While surfing through the pymc-devs repository earlier in the year, I found some beginner-friendly issues which I could fix like 'conver_size and convert_shape wrongly assuming the sizes have to be scalar #5394' link. I plan to work on more issues in the Pre-GSoC period.

Being part of the team that worked on the deployment of the Bayesian model, it is clear to me that refinement in the development procedure is very desirable.

I believe GSoC will be a great step for me to contribute in the open source community

Deliverables

The ultimate goal is to make a model class that wraps PyMC models with a save and load method. The following protocol sections will be implemented. The secondary goal is to document and provide a proper tutorial for users to easily deploy their models on various standard production systems.

1. Add the API call for model building, loading, saving, fitting.

   Existing PyMC developers sometimes find it difficult to save their model and load it later to predict or fit. This issue can be resolved by introducing a wrapper to support the user by making it easier to load and save their model.

   Currently, two main packages have implemented this in order to make the development experience better, namely

   a. PyMC-learn link
   b. Pymc3_models link

   These two packages are heavily inspired by features offered by scikit-learn, which is a Python module for machine learning built on top of SciPy.

Scikit-learn allows pickle to save models. However, the main problem faced by PyMC is there is no direct way to save aesara graphs in JSON or HDF5 format to load them later in the model.

This leaves us with the option to make a class that contains the model while loading the model.

Serialization in Scikit-learn models majorly relies on pickling, and the metadata is saved along with the pickled file. Scikit-learn also supports the joblib for the same task to overcome big data and is optimized for numpy.

To encounter this following approach can be used:

In order to save and load a user created model PyMC and the model artifacts we can make a python class to save the model with its metadata. These python classes would be saved in a general fileformat system.

```
./path/
  ./pymcModel: configuration
  <code> : code packaged with model (as specified in pymcModel file)
  <data> : data packaged with model (as specified in pymcModel file)
  <env> : conda environment used in deploying the model (as specified
in pymcModel file)
```

File system format

Using a file system like this will allow us to save all the dependencies of the model via the conda environment, and it will contain other important information in order to load and use it.
Storing a conda environment can be a very heavy task and to encounter that packaging the environment separately and storing the hash or guid of the environment to cross check the dependencies of the environment while loading an environment back using APIs seems to be a viable approach.

2. Document and improve deployment of PyMC Models
   There currently is no documented way of deploying PyMC models in production.
   The above class will lay the foundation for the secondary goal of this project: write tutorials on how to deploy PyMC models (using the above class) in the most common production systems, including Docker, Kubernetes, airflow, and dask.
   The goal is that a data engineer who is not familiar with PyMC but is tasked with bringing a model into production can go on the website and find the right resources

to make this process easy. Mike Krieger documented his experience using the Airflow link.

This can be achieved by maintaining a proper log of how to deploy a model on these productions for which I will deploy some basic models on these production systems.

Approximate timeline

Time Period	Milestone
Pre GSoC 20th April - 19th May	● Engage with the PyMC community ● Work on currently present issues on the PyMC repository
Community Bonding Period 20th May - 12th June	● Research and gain experience with PyMC ○ Understand the current codebase of PyMC ○ Learn about the current model deployment pipeline PyMC offer ○ Understand model class available in Scikit learn, mlflow, TensorFlow
Week 1 & 2 13th June - 26th June	● Setup development repository ● Work on base class to save the model configuration and load it from memory.
Week 3 & 4 27th June - 10th July	● Implement API calls to predict and fit a user created PyMC model by loading it back to the program and making functions to predict and fit.
Week 5 & 6 11th July - 24th July	● Buffer to work on any issues faced during the development of the model class
Week 7 & 8 25th July - 7th August	● Update the current Docker container with added features and fix on the existing errors and bugs ● Work on documentation of deployment using docker, kubernetes, airflow and dask
Week 9 & 10 8th August - 21st August	● Continue on documentation of deployment using docker, kubernetes, airflow and dask
Week 10 & 11 2nd August - 4th September	● Complete with white paper documentation of the model class and the development procedure of various production systems

Final week 5th September - 12th September	Buffer for debugging/unintentional delays. Compile resource list that future developers can follow. Write down and submit the final project report.

Deployment Experience

I've been contributing to PyMC for a couple of months now by resolving beginner-friendly issues. I haven't used PyMC to deploy any real-world project but I plan to do so pre in the pre-GSoC period. I have gone through PyMC example notebooks present on the PyMC docs website (link). These example notebooks gave me insight into how PyMC works and has great use while working with complex probability problems. Moreover, I have worked with the deployment of a few scikit-learn models using pickle and believe I will be able to complete a project like this with constant effort and the skills I possess.

Personal Information:

- ☐ I am a pre-final year engineering student at Thapar Institute of Engineering and Technology, pursuing my Bachelors of Engineering in Computer Engineering. I have been programming in python primarily since 2019. My interest in python grew over time as my experience in it scaled through working on various projects mainly related to image processing and natural language processing.
- ☐ I recently completed my work with IIITA where I worked with development of CNN and GNN using python for classification of images in semi-compressed domain.
- ☐ I have always been keen to participate in an open source project and I know there couldn't be a better platform than GSoC for the same.
- ☐ I assure to work for ~40 hours in a week and around at least 10 hours on the weekend. I won't be taking any holidays of a span greater than one or two days