FLASK INTEGRATION DOCUMENT

Flask is a web application framework written in Python.

Directory Structure:

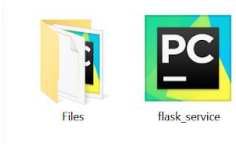
Project Directory >>

- flask_service.py
- **Files >>**
  - roll_no.pkl (Model Pickle File)
  - roll_no.py (Preprocessing and Feature Engineering Python File)

Flask Zip File:

- First and foremost download the Flask zip file from the noticeboard.
- Extract them and store it in the Project Directory.
- Rename all the files in the **FILES** directory by your roll_number.
  - Eg. 1405043.py → 17******.py
  - And the class name as well in that file.
    - class _1405043(): → class _17*******():
- Before you proceed with Flask Deployment, if you have applied multiple models in your notebook. Select only one and remember the features passed in that.
  - Eg. Linear Regression:
    - Features List: amount, days, mean, median.
    - During Deployment, remember the features list and how to recreate them only. No Feature selection in Deployment.
- Go to your final notebook, select which fitted model you are going to use. Create a pickle file. Follow the Code snippet below.
- And store the Pickle (.pkl) file in the directory mentioned above.

Installation:

1. Open CMD / Conda Prompt / Jupyter Notebook (If CMD not working, try CONDA Prompt)
2. Type “pip install Flask” in CMD/Conda Prompt (https://pypi.org/project/Flask/)
3. After installation, please cross check your Project Directory Structure
4. 

5. Inside **Files** Folder, there should be two things present ->
    a. Pickle File (Incase you don't have pickle library, pip install pickle):

```
import pickle
filename = ‘roll_number.pkl’
pickle.dump(model, open(filename, 'wb'))
```

    b. Preprocessing & Feature Engineering Script: roll_number.py

Integration:

6. Just like any python program, In your CMD or Anaconda Prompt whichever available run ***python flask_service.py***. This ***flask_service.py*** command runs a Python file and sets `__name__ == "__main__"`. If the main block calls `app.run()`, it will run the development server.
7. Flask_service.py is the flask application. Inside it we have a route named **‘/predict’** , which will be called from the client side . The route is responsible for returning the output from the model in the form of a json object.
8. Inside ‘predict’ we instantiate an object of ‘_rollno’ class and call getPredictions function which will return us the result.

9. Inside _rollno class we have written different transformation functions which will do the preprocessing and feature engineering , followed by obtaining predictions from the prebuilt model(roll_no.pkl) and return the predicted value to Flask_service.
10. Once the Flask Server is up and running you can call it from React by using the below function:

```
export function prediction(data) {
  return axios.post(
    'http://127.0.0.1:5000/predict?',
    {},
    {
      headers: { 'Content-Type': 'application/json' },
      params: {
        data: data,
      },
    }
  );
}
```

11. The format of the data sent should be:

```
{
        "id": "<ROLL_NO>",
         "data": [
                        {
                          .......
                         },
                        {
                          .......
                         }
                  ]
}
```

12. The column id to display the response from Flask should be ‘predictions’ and ‘predicted_payment_type’.