Group 23 - feedback for final project resit

PRELIMINARIES

1. **WORK ON LINUX OR WSL**
2. Communicate to the lecturer your GitHub usernames. You will be receiving an invitation to join a new GitHub repo as collaborators
    a. Your repo: rug-oop-2024/group-23-resit
3. On a local machine, delete the .git folder from the project files
4. Open the terminal on the folder, type “git init”. This will create another git repository detached from the previous one
5. In the same terminal, type “git remote add origin url_of_github_repo_from_pt_1”
6. Add every file to the new repo “git add -A”
7. Commit ‘git commit -m “First commit of resit”’
8. Push “git push origin master”. If branch master does not exist, replace “master” with “main”

- Modelling
    - Regression
        - Linear regression: fix pseudo-inverse and have common submethod for stacking column of 1’s to avoid code repetition
        - **Fix: We have implemented a method in the base model for stacking columns of 1’s which is now used in all of the models. We are also using the inverse instead of the pseudo-inverse for the multiple linear regression model.**

        - The models seem to be working, however, the metrics seem completely off. MSE, MAE and R^2 seem very good, however the training set

predictions are completely inapplicable (all negative, see image)

Train Set Predicti

value
-1.167
-0.9617
-1.302
-1.3162
-1.3371
-1.2021
-1.55
-1.1812
-1.3448
-1.0397

- In addition, the presentation of the metrics is very underwhelming and not readable:

Results

train_<autoop.core.ml.metric.MeanSquaredError object at 0x7f15e5198f40>: 0.0956893509783104

train_<autoop.core.ml.metric.MeanAbsoluteError object at 0x7f15e519b790>: 0.2399777699450778

train_<autoop.core.ml.metric.RSquared object at 0x7f15e5198040>: 0.8344586686792292

test_<autoop.core.ml.metric.MeanSquaredError object at 0x7f15e5198f40>: 0.0956893509783104

test_<autoop.core.ml.metric.MeanAbsoluteError object at 0x7f15e519b790>: 0.2399777699450778

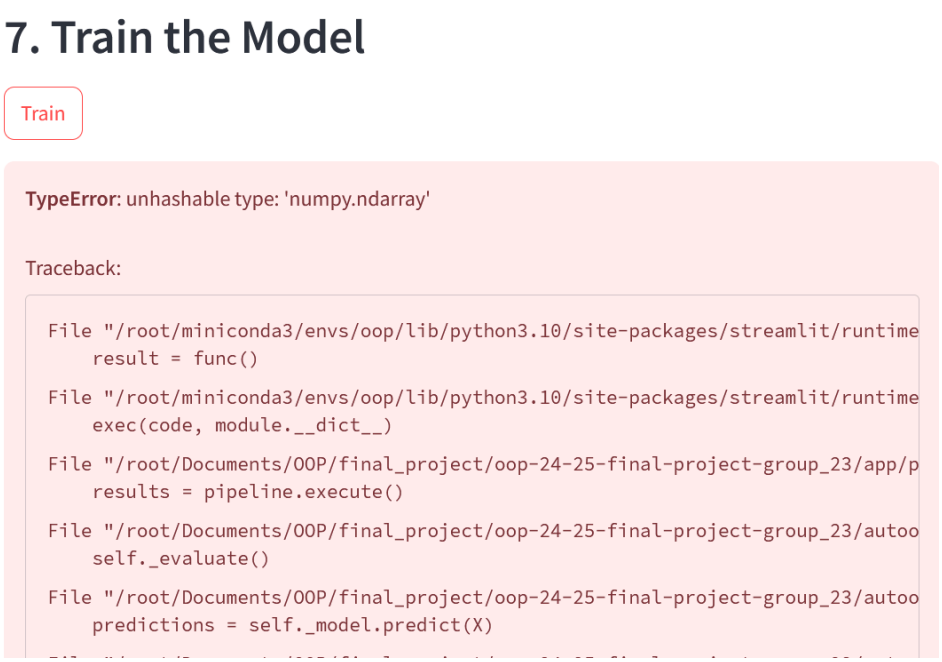
test_<autoop.core.ml.metric.RSquared object at 0x7f15e5198040>: 0.8344586686792292

Implement __str__ and __repr__ on the metric classes to allow them to be printed in a human-readable form

- **Fix: We have implemented the __str__ and __repr__ methods, and now the metric evaluations are printed without the unnecessary object value: “train/test_MetricName: value”**

- Classification
  - The models don’t seem to work (see error below)
  - **Fix: We had the same problem across all of the classification models, which is that y had the incorrect shape and it was not compatible with the labels for computations. We managed to solve this by converting y to the correct shape, using the numpy function np.argmax(y, axis=1) in the fit methods of the models and in the**

metrics used for classification. Now all classification models work as expected.



    - Saving as artifact does not work (no artifact is saved)
    - **Fix: The issue was with the Modelling page. Because the save pipeline depended on training it, (if train > if save, the if save was inside the if train) after pressing the save pipeline button, the interface disappeared and it once again prompted the user to train the model, so the process did not reach to actually save the pipeline. After separating the two if's, and using the st.session_state feature from streamlit to remember certain attributes in the execution, the pipeline save bug was fixed.**
- Deployment
    - Check that deployment works correctly when artifact save bug is fixed
    - In addition to the project request, give the possibility of comparing the model prediction to the ground truth (if provided) and let the user choose metrics for this comparison
    - **New functionality: On the Deployment page, the user has the possibility to compare the model's predictions to the ground truth if they provide it, in an input box, which requires a list of values, separated by commas. Then they can choose from the compatible metrics list (either for classification or for regression). The program will output the metric values for this comparison.**
- Style
    - There are some public attributes: be careful to use private attributes and use @property to give a public interface to it
    - **Fix: We made the attributes like name and type for the models private, by creating a setter and a getter for them, using @property, and @setter, in the base class Model.**

    - In the models, the get_parameters property is redundantly implemented in each concrete class. Moreover, please name the getters with the same name as the respective attribute. Also, if you use @property, try to stick to the name of the private property (minus the initial leading underscore). So, if your attribute is called _parameters, the method for accessing this property should be called “parameters”
    - **Fix: We implemented the getter (with @property) for the parameters in the base class and removed the other getters from each individual model. We also named it “parameters” to abide by the naming convention.**

SUBMISSION

- Submit the project on GitHub AND the dedicated resit assignment on Brightspace by Jan 26th
- Add to GitHub a copy of this document in which you briefly explain how the feedback has been implemented