Github link for project: <https://github.com/Talha188344/503_PROJECT/>

All usable code for the project is contained within the FINAL area

Python requirements:

pandas

sklearn

scikit

datetime

numpy

matplotlib

Classifier Chains:

There are 4 different CC python files, each with its own purpose:

CC\_CORE.py is the most foundational of all 4, with its content serving as the base for the other 3. We create the classifier chain and ovr, get accuracies for all chains and OVR. Create and judge the ensemble chain and graph the results. It was heavily inspired by [this sklearn guide](https://scikit-learn.org/stable/auto_examples/multioutput/plot_classifier_chain_yeast.html):

https://scikit-learn.org/stable/auto\_examples/multioutput/plot\_classifier\_chain\_yeast.html

CC\_VS\_OVR.py compares the OVR and ensemble classification methods in terms of accuracy, printing the results in the terminal

CC\_RUNTIME.py is the longest python file. It measures the runtime of the classification chains vs the number of labels and vs the number of chains used in the ensemble, then graphs them

CC\_F1\_SCORES.py is similar to CC\_VS\_OVR but specifically compares F1 Scores and graphs them

**For all python files I would only recommend altering the dataset used for testing or the algorithms used (logistic regression and decision tree) by the chains and OVR, this is explained in comments near the top of code and by relevant lines. All programs can then directly be run and graphs and data will be shown/printed in terminal.**

All code related to classifier chains was created by Evan Lang, if you have any questions or encounter any problems please contact:

evanlang@bu.edu

Calibrated Label Ranking:

All code related to Calibrate Label Ranking was created by Samarth Singh(Samarths@bu.edu) and Talha Jahangir(Talha98@bu.edu), if you have any questions or encounter any problems please contact:

File :- CLR.py

**Functions**

* add\_calibration\_label(Y): Adds a calibration label to the dataset.
* multilabel\_to\_calibrated\_ranking(Y): Converts multilabel data to a calibrated ranking format.

**Models**

The script currently supports the following models:

* **Logistic Regression**: Implemented using scikit-learn's LogisticRegression.
* **Decision Tree**: (Optional) Uncomment the relevant lines in the script to use DecisionTreeClassifier.

**Outputs**

The script outputs the Jaccard scores for each model and each label pair from the test data. These scores indicate how well the models are performing.

Datasets used: PC dataset

OPENML: Reuters

Taken from OPENML

PC handmade Dataset:

There are 4 csv files included as well, serving as the test and training data of the PC Dataset. These must be located in the same area as the code if you intend to use the dataset. If you do there are comments in code showing how to do this. The non pc datasets were taken from OPENML, there are also comments explaining how to choose which one to use.

PC Dataset code was created manually by ripping values from user runs on the website <https://www.userbenchmark.com/>

Labels were created in excel, using IF and AND statements to check if criteria for performance were met. Labels for upgrades were made by first detecting if the performance goals were met, then suggesting upgrades in an order that corresponds to how user benchmark weights each pc part. For example, if gaming performance was not met it would first check if the GPU is underperforming and recommend it be upgraded if it was. If not it then moves to cpu, then ram, then memory etc.

You can find the postprocessing of the dataset in the equations for columns N-Q within PC\_DATASET.xlsx within the final folder. Some equations (Met expectations equations) were unfortunately cut out between the datasets conversion to CSV and back

Dataset feature creation performed by Samarth Singh

Dataset postprocessing performed by Evan Lang