Dear fellow students,

Welcome to the project ImbalancedLearningRegression! My name is Wenglei Wu, one of the four initial members implementing the package. In this letter, I will briefly introduce the structure and functionality of this package. Hopefully, you will be familiar with the ideas involved and be more confident with your implementation.

Basically, your semester will be divided into four parts:

1. Comprehension
2. Coding
3. Packaging
4. Demonstration

<u>Comprehension</u> (week 1-3)

This part will take you roughly 3 weeks, during which you will understand the background of the problem, the structure of our package, the existing implementations, and your choice of re-sampling methods.

<u>Background of the problem</u>

To understand the background of the problem, I would suggest you answer the following questions:

1. What is classification? What is regression? What are the differences between classification and regression?
2. What is the target variable?
3. What is an imbalanced domain? What's an imbalanced dataset for classification?
4. When you are training a classification model, what are the issues resulting from an imbalanced dataset? Any real-world examples (abnormal cases are often rare)?

If you have some machine learning knowledge, for example, if you have taken the courses CSI4106, CSI4107, CSI4142, MAT2377, MAT3373, etc., you will be able to answer some or all the questions above. The questions below may be novel, and they are the core of this package.

5. What about regression? How do you know a dataset is imbalanced if the target variable is continuous?
6. Why do we also care about imbalanced regression? Any real-world examples?
7. What is the relevance function?
8. *Optional*: understand relevance function attributes. What is the distribution focus? What is the box plot? What is the box plot coefficient? What are the control points? How is the relevance function automatically estimated?
9. Is there any rough relationship between the probability density function (PDF) of a dataset and its relevance function in terms of their shapes in the coordinates plot?
10. What is the relevance value? What is the range of it?
11. If the relevance value is above/below the threshold, the sample is considered as minority (rare) or majority (normal)?

12. What are the bins and bump classification?
13. Given a sorted list of target variable values, the outputs of the "phi()" function are their corresponding relevance values. Then, how do you get the bump classifications using those relevance values?
14. How is the percentage of re-sampling determined? What is the difference between "balance" and "extreme"? Note that "manual_perc", "perc_u", and "perc_o", are introduced by me for fun, allowing users to give a fixed percentage of re-sampling for all bins, so you don't necessarily need to implement them.
15. Are the operations, such as re-sampling, random choosing, and K-NN searching, conducted over the entire dataset, or for each bin one by one? Note the differences between different techniques. BE CAREFUL WITH THIS QUESTION. Since the research paper may only focus on imbalanced classification and may not discuss the multiclass situation, you will discuss this with Dr. Branco when you are implementing a new method. We may have multiple minority bins and majority bins, and there are many possibilities. Here are some examples:
    A. In the "*Introduction of Gaussian Noise*" (GN) method, we perform over-sampling on the minority bins one by one. When we are over-sampling one minority bin, the choice of the nominal attribute value for a new sample is randomly chosen from all the rare cases in that bin only, and the Gaussian distribution of a numeric attribute is also for that single minority bin only.
    B. In the "*Edited Nearest Neighbor*" (ENN) method, although we perform under-sampling one by one on each majority bin, the K-NN classifier searches neighbors of a sample from the entire original dataset.
    C. In the "*TomekLinks*" method, we do not even use bump classification. The re-sampling and K-NN searching are both conducted over the entire dataset directly.

You are not asked to answer all these questions within one day or one week. You can try to understand these questions when you are reading the research papers and comparing them with the code. You can also have some discussions with Dr. Branco during the weekly meeting. You may also include your answers to these questions in the introduction part of your presentation and report.

Structure of our package

My group implemented Random Over-sampling (RO), Introduction of Gaussian Noise (GN), Condensed Nearest Neighbor (CNN), and Edited Nearest Neighbor (ENN). The other four methods, namely Random Under-sampling (RU), TomekLinks, SMOTE, and ADASYN, were implemented by another friend group. Since I checked the code quality of all, I know some aspects of their implementations.

According to Dr. Branco, the first and the only Python implementation for solving the imbalanced regression problem is SMOGN, which was implemented by Nick Kunz in his package SMOGN. In fact, our package was forked from SMOGN, so you may find the structure of the two packages quite similar. We re-used his code for the relevance function and maintained the overall structure. Similarly, I expect you to use the same structure to maintain consistency.

As you open the GitHub repo, you will find that there are six folders. Firstly, all codes for the re-sampling methods are included in the folder "ImbalancedLearningRegression". Each re-sampling technique is represented by two files. The main function is in the file named by the method name or abbreviation (e.g., ro.py, cnn.py, smote.py, tomeklinks.py). The supporting function is coded in the file whose name has the prefix “over_sampling” or “under_sampling” (e.g., over_sampling_ro.py, under_sampling_cnn.py). Basically, the pre-processing steps, the bump classification, and the combination of results are tasks for the main function; on the other hand, the re-sampling of each bin is completed by the supporting function one bin at a time.

“dist_metrics.py” is only called by SMOTE, ADASYN, and TomekLinks. These three methods re-used SMOGN’s distance metrics. CNN and ENN, however, used the “KNeighborsClassifier” from “scikit-learn” to accelerate the process. When you are implementing your methods, use your intuition to choose the most appropriate and efficient way (you may want to use external packages)! **Once you finished implementing a method, please update the file “__init__.py”.** The other files in the same folder are related to the relevance function. You are only required to have a rough idea of what the relevance function is, and what the inputs and outputs related to the relevance function are, so it is not necessary to spend too much time reading those codes. But if you have time and interest, feel free to read them.

Secondly, the folder “data” contains the datasets for testing purposes. Thirdly, the folder “docs” contains the files for the documentation. Fourthly, the folder “examples” contains the Jupyter Notebook use cases. Fifthly, the folder “tests” contains the test files that run the methods with different parameters to ensure our code is running correctly and smoothly. Finally, the folder “archive” contains other files we want to store, either temporarily or permanently.

Our package used external packages “pandas”, “numpy”, “scikit-learn”. If you have some machine learning application background, you should be familiar with these, which could save you a lot of time understanding the data frame, different methods, the pipeline, etc. If not, you can learn from the documentation of those packages. “pandas” and “numpy” are included in all files, while whether to use “scikit-learn” depends on the methods you are going to implement. You may also use any K-NN classifier from other packages or your own implementation if you prefer, as long as the code is working and efficient.

The package “tqdm” simply provides a progress bar for loops. For example, your code can be changed from “for i in range(n)” to “for i in tqdm(range(n), ascii = True, desc = "your choice of name")”. Python will display the progress for the loops.

Existing implementations

You may want to understand the existing codes while reading the research papers. Paper titles can be found in the package documentation. If the authors include Dr. Branco, I am sure she has copies of her papers, and you can ask her. Otherwise, you can log in uOttawa online library using your uOttawa account and search the papers.

SMOGN was the only Python package for us to learn when we began to build our package. You, however, have eight more methods to refer to. I would suggest you learn the overall process by comparing the papers and codes of SMOGN, SMOTE for regression, and GN. Note that there are errors in Nick’s SMOGN implementation, and I don’t know if he fixed those errors when you read this. Those errors were fixed in our SMOTE and GN. If you are uncertain when you are comparing, you can discuss your concerns with Dr. Branco. If you find any bugs in our eight implementations, feel free to email me to discuss your findings and thoughts.

I would not suggest you learn from the implementations of ADASYN and TomekLinks unless your method waiting to be implemented is a variation of the two. Because these two methods have special processes and runtime, which are different from the other six methods and are controversial. However, if you have plenty of time and want to distinguish among all the methods, you are free to do so.

You can also discuss with Dr. Branco to figure out what to learn. She can give you more appropriate suggestions based on your choices of re-sampling methods.

Your choice of re-sampling methods

Once you are familiar with the overall structure and process, you can begin to read the paper associated with one of your methods.

If you are working alone, you may be asked by Dr. Branco to choose one over-sampling method and one under-sampling method to code. The workload will be doubled if you work in a group of two.

Other than the existing eight methods, I don't know much about the methods you are going to implement. If you are implementing any variations of SMOTE, you may refer to the implementation of SMOTE. If you are implementing Repeated ENN, you can refer to my implementation of ENN.

No matter what method you choose, I am sure you need the relevance function to classify majority and minority samples. For most methods, you need bump classification to be able to re-sample each bin one by one.

In addition to the research papers, you may also refer to the code in imbalanced-learn and Dr. Branco's UBL packages (see useful links). Keep in mind that the solution for imbalanced classification is not equivalent to the solution for imbalanced regression, especially when we have multiple majority bins and minority bins. For example, if we have two majority bins, they may represent two distinct classes, or they may both be treated as belonging to one class "majority". Reconsider question 15.

Coding (week 4-9)

Once you read a research paper and understand a re-sampling technique, you can begin to code it. You will likely spend three weeks on the over-sampling methods and another three weeks on the under-sampling methods. Should you have concerns about the paper you read, please discuss them with Dr. Branco.

Please ask Dr. Branco to add you as a collaborator on GitHub. I would suggest you together create a new branch and merge it with the master branch by the end of the semester.

If you want to debug your codes as separate files instead of a package (to save the unnecessary time to upload, uninstall, and install the package from GitHub repeatedly), you can simply remove all "ImbalancedLearningRegression." When your load the dependencies. For example, change "from ImbalancedLearningRegression.over_sampling_gn import over_sampling_gn" to "from over_sampling_gn import over_sampling_gn". Don't forget to change them back before building the package.

<u>Packaging</u> (week 10-13)

During week 10, you will test your code. You can imitate the test files of the initial eight methods, or you can invent something new to test your code.

During week 11, you will create Jupyter Notebook examples for the methods you implemented. Again, you can mimic other example files.

During week 12, you will update the "read the docs" documentation. You should follow the same structure as other ".rst" files. **<u>Please update "index.rst" when you create a ".rst" file for a method.</u>** You can enter "pip install sphinx" to install sphinx and enter "make html" in your console to see the updates locally. Make sure you are in the folder "docs". You should be able to see the documentation updates online after you make a new "push" request to the master branch or merge your branch to the master branch.

During week 13, you will release the new version of the package on PyPI. Please coordinate with other groups if applicable. Please register for a "PyPI" account and ask Dr. Branco to add your account. Only one student is required to upload the updated package per semester.

Before you build your package, please complete the following checklist.

1. Update version number in "setup.py", "docs/conf.py", "ImbalancedLearningRegression/__init__.py".
2. Add the reference of the papers to your source files, documentation, and examples. Copy all new references of the papers to "README.md", "docs/intro.rst".
3. Write your contributions in the "UPDATES" file.
4. If you used new external packages, please update "install_requires" in "setup.py", as well as "Requirements" in "README.md" and "docs/intro.rst".

<u>Demonstration</u> (week 13 & exam period)

During week 13, you will also prepare the slides for your presentation. You may start editing your slides earlier if you want some feedback from Dr. Branco.

Your presentations may be scheduled by the end of week 13 or during the exam period. Your report will be due during the exam period.

Some useful links:

GitHub repo: https://github.com/paobranco/ImbalancedLearn-Regression

Package documentation: https://imbalancedlearningregression.readthedocs.io/en/latest

Read the Docs documentation: https://docs.readthedocs.io/en/stable/index.html

Python package instruction: https://realpython.com/pypi-publish-python-package

Pandas documentation: https://pandas.pydata.org/docs

NumPy documentation: https://numpy.org/doc/stable

Scikit-learn documentation: https://scikit-learn.org/stable

Scikit-learn pre-processing: https://scikit-learn.org/stable/modules/preprocessing.html

List of re-sampling methods for classification in imbalanced-learn: https://imbalanced-learn.org/stable/references/index.html

Dr. Branco's UBL package codes in the R language: https://github.com/paobranco/UBL/tree/master/R

Some useful commands:

"pip install pandas"

"pip install numpy"

"pip install sklearn"

"pip install tqdm", used for the progress bar.

"pip install sphinx", for editing documentation.

"pip install seaborn", used for plotting.

"pip install matplotlib", used for plotting.

"pip install ImbalancedLearningRegression" to install the PyPI release.

"pip install git+https://github.com/paobranco/ImbalancedLearningRegression.git" to install the developer version.

"pip uninstall ImbalancedLearningRegression" to uninstall "ImbalancedLearningRegression".

"pip install build twine" to install package building and uploading tools.

"pip install jupyter notebook" to install jupyter notebook.

"python -m build" to build the package files.

"twine upload dist/*" to publish to PyPI.

"make html" generates the HTML files to view your documentation updates locally.

"jupyter notebook" to open the notebook.

"git clone https://github.com/paobranco/ImbalancedLearningRegression" to clone files from GitHub.

"git remote add origin https://github.com/paobranco/ImbalancedLearningRegression.git" to replace the URL with origin.

"git checkout -b [your branch]" to create a new branch.

"git add -A" to include all your modified files.

“git commit -m ‘[your comment]’” to make a new commit.

“git push origin [your branch]” to push to GitHub.

“git pull origin [your branch]” to pull the updates on GitHub before pushing.

“git checkout [your branch]” to switch branches.

“git merge [your branch]” to merge your branch to master, please make sure you are switched to the master branch.

Git commands provided here may not be accurate, please check them carefully and make the adjustment, if necessary, based on your local Git status and settings. It’s always safe to check online for an operation.

Please uninstall “ImbalancedLearningRegression” before you install your updated “ImbalancedLearningRegression” on your device since the package version number is fixed during your semester. Otherwise, you are still using the old one.

If you have any questions about the details of the re-sampling techniques you are implementing, please discuss them with Dr. Branco. If you have concerns regarding the initial eight methods or the project design, and they cannot be answered by her, you can email me. I will check my uOttawa email after my graduation, there is no guarantee as to the time of reply though. Thank you in advance for your contribution to the package. Good luck with your semester!

Best regards,

Wenglei Wu (wwu077@uottawa.ca)

June 30, 2022