AML 2021: Assigment

Alan Chalk

May 23, 2021

The deadline for this assignment is given on Moodle.

The assessment for AML 2021 is based around a private Kaggle in-class competition. The link will be released on Moodle during the first few weeks. Please do not share it with anyone outside of our AML class. You enter the competition by downloading the data and accepting the rules. Set up a team and invite your team members to join the team. Group members that do not join their Kaggle team will not receive any marks for this assignment. Teams that invite external members to the team will not receive any marks.

Please note that all scripts will be checked for plagiarism. In previous cohorts, students have been failed for this. If you satisfy the pre-requisites, keep up with lectures and work on the assignment, in a group, throughout the course you should be able to complete this assignment. There is no negative marking and a good pass is possible without attempting Part 5 if a conscientious and thoughtful attempt is made on Parts 1-4.

Your work may end up consisting of various python notebooks, screenshots of Kaggle entries and a word document. These should be zipped and uploaded to Moodle before the coursework deadline.

The competition metric is Area Under the ROC curve (AUCROC). We did not cover this in class (though one of the links covers it). Briefly:

- You may think of it as the probability that two customers chosen at random will be correctly ranked for the risk of carrying out fraud.
- If AUCROC is 0.50 it means there is a 50/50 chance of ranking any two randomly chosen customers correctly. You could have done this by simply assigning model predictions at random. Therefore if your AUCROC is about 0.50, your model is no better than a naive model.
- If AUCROC is 1 it means that you rank any two randomly chosen customers correctly every single time. This is a perfect model. If you attain

an AUCROC of 1 on the train data, then you have completely overfitted your model. If you get an AUCROC of 1 on the test data it means I have made a mistake and you should post this on Moodle immediately!

There are 5 parts to this assignment:

- Part 1. Explaining your choice of environment. Full details are below. This part counts for 2% of your final grade.
- Part 2. Linear regression - based on week 2/3 material. Full details are below. This part counts for 8% of your final grade.
- Part 3. Penalised logistic regression (elastic-net) with H2O - based on week 4 material. Full details are below. This part counts for 30% of your final grade.
- Part 4. Gradient boosting with one of H2O, lightbgm or catboost - based on week 5 material. Full details are below. This part counts for 30% of your final grade.
- Part 5. A choice of one of the topics covered in week 6. Full details are below. This part counts for 30% of your final grade.

Part 3 is meant to test general knowledge and understanding and marks will be granted for all reasonable attempts (provided they cover each of the five points below). You should find that the Week 3 coursework as well as the notebooks provided for Week 4 give you a good starting point.

Parts 4 is more challenging than Part 3. Marks will be awarded for detailed and carefully thought out work - more details on this are given below.

Part 5 is hard. You are expected to research the topic you choose in more detail than provided in lecture. Further details are provided below.

To get you started, two notebooks will be provided:

- 01a ReadData.ipynb: This helper notebook reads the train and test data. You don't need to use it - you can read the data in however you like.
- 04b DecisionTree.ipynb: (Provided later.) An example use of h2o to create a decision tree which you can run to check that your setup is working. This notebook is based on the 250,000 line train data and if you submit of Kaggle you should get an AUC of around 0.75.

Part 1

- Describe the environment that you and your team used for this assessment.
- List the advantages and disadvantages compared to other choices of environment (such as Kaggle kernels, Cloud based environments, conda based environments and so on).
- Explain the reason for your choice of environment.

Part 2

Train a (simple) penalised linear regression model on the competition data.

- You should use a model from sklearn. You can use Ridge, LASSO or Elastic Net (or their CV versions).
- For purposes of the competition, your prediction needs to be in the range 0-1. Since you are using linear regression your predictions are likely to fall outside of this range. You should therefore clip your predictions to fall in this range.
- You can use the smaller dataset for training and you can choose just a small number of the features. But you should:
  - Choose at least two numeric variables
  - Spline at least one of the numeric variables that you choose
  - Choose at least one categorical variable and one-hot it
- You should carry out some form of search for the best lambda. It does not need to be an exhaustive search.
- Predict your model on the test data, clip the predictions to 0-1 and submit an entry. Take a screen-shot of your entry. Note that you should not expect a particularly good score.

All your Python code should be in one notebook. The name should be Group_x_part2.ipynb. Your screen shot should also be included in your zip file and should be called Group_x_part2_screenshot.png

Standard requirements apply to your Python work. In particular it should be commented and reproducible. The comments should persuade me that you understand what you are doing. "Reproducible" includes that if I run your code and submit predictions to the competition that I get the same score. I cannot give marks for work which is not reproducible.

Part 3

Train a good penalised logistic regression model on the competition data. The only type of model you should use for this part, is the H2OGeneralizedLinearEstimator model.

You should:

1. Deal appropriately with missings (for all numeric variables, -99 means missing).
2. Deal with numerics - i.e. for at least some try linear splines (or another method of your choice to deal with non-linear effects)
3. Deal with hccvs (eg using the feature encoding library that we looked at in lecture) (You do not need to deal with low cardinality categorical features since H2O will one-hot them for you.)
4. Try out some interactions
5. Try out some other features (eg division of numerics).

All your Python code should be in one notebook. The name should be Group_x_part3.ipynb. Your screen shot should also be included in your zip file and should be called Group_x_part3_screenshot.png

Standard requirements apply to your Python work. In particular it should be commented and reproducible. The comments should persuade me that you understand what you are doing. "Reproducible" includes that if I run your code and submit predictions to the competition that I get the same score. I cannot give marks for work which is not reproducible.

Part 4

Choose one of the gradient boosting implementations we covered in week 5 i.e. one of H2O, lightbgm or catboost. You should carefully research relevant documents for the key parameters and how to tune them. Create your best model and submit on Kaggle.

Marks will be awarded for work which:

- Demonstrates that the user documents for version of boosting used have been thoroughly read

- Includes tuning of all relevant hyper-parameters, and a summary of which h-p's mattered most for the final result and why this might be the case.
- Shows efforts to use hyper-opt or other dedicated h-p training softwares
- Uses custom loss functions which penalise overfitting.
- Is easy to follow and very well commented. In particular where comments help me to follow your thought process and demonstrate an understanding of what you are doing.

All your Python code should be in one notebook. The name should be Group_x_part4.ipynb. Your screen shot should also be included in your zip file and should be called Group_x_part4_screenshot.png

Standard requirements apply to your Python work. In particular it should be commented and reproducible. The comments should persuade me that you understand what you are doing. "Reproducible" includes that if I run your code and submit predictions to the competition that I get the same score. I cannot give marks for work which is not reproducible.

Part 5

Choose one of the topics we will cover in week 6. For your chosen topic, experiment with it and document your experiments, findings and code in a Python Notebook. If you choose model interpretability, then include some text near the beginning discussing your findings and comparing the results of the different methods. If you choose another model type (such as stacked ensembles or auto-ML), then fit the models and document your model fitting process in a similar way to part 4.

If, for this part, you have chosen to fit another model, then include a screenshot of your competition entry as usual.

Make sure that your notebook is neat and well commented and uses markdown as appropriate to make it easy to follow.

Some suggestions for you to consider are below:

h2o Driverless AI - DAI

This is a placeholder - we hope to be able to add DAI to week 6 topics this year, however it is still under preparation.

Stacked Ensembles

- Take some time out to search for interesting blog posts which show how people have used ensembles. (Interviews with Kaggle winners may be a good place to start looking). Include near the start of your notebook a few interesting links.
- See how far you can get with the h2o stacked ensemble. But try to use other methods too. sklearn has a class for this. This link to a Jason Brownlee post may be of use Jason Brownlee on stacking ensembles.
- Try to create a stacked ensemble yourself as we did at the end of the last video and experiment with different meta-learners.
- Document which approach works best and why you think that is.
- Document difficulties you have and ideas you have for future work.

AutoML

- Choose a two or three autoML algorithms. H2O, tpot and sklearn are possibilities. This link has some other ideas Top AutoML Frameworks for Machine Learning Applications (May 2019) . Note that H2O's autoML is poorly documented (as far as I can see) and you need to use more than this do score well in this section.
- Read the documentation to make sure you understand how the autoMLs you are using work.
  - tpot uses genetic algorithms. Describe how it works (not more than 100 words). The early videos in this (amazing) playlist may help Daniel Shiffman: The evolution of code. (In fact, all of his playlists are amazing and if you are interested in ML, this is a good place to go.) You should explain the "generations" and "population-size" parameters of tpot and how it works in general. Run tpot autoML. How long does it take to find a good model? What is the best model it finds? Describe any experiments you carry out, and constraints (which may include compute time and costs). You should also describe the algorithm that it comes up with, to the best of your ability (given we did not cover all the sub-models it uses).
  - h2oML. Describe how it works (not more than 100 words). Read the basic documentation h2o autoML. Describe how it works. Run h2o autoML. Describe any experiments you carry out, and constraints (which may include compute time and costs). You should also describe the algorithm that it comes up with, to the best of your ability.
  - Other autoML's. As above.
- Absolutely no marks will be given for simply running some autoML algorithms and reporting the result.

Model interpretability

- It would be interesting to carry out any analysis over two different models and compare the results. For example you could use your penalised logistic regression and also a gbm model from part 2.
- Comment thoughtfully on the comparisons between the methods. For example; Do the three different approaches to variable importance give the same result on a tree based method? Do SHAP and LIME give similar explanations? If not, can you think of any reason why not?
- For the most important variables, do PDP's show a clear effect? Is it linear across the range of the feature? For the least important variable, does the PDP show anything?
- Find some potential interactions using the interaction feature importance method we discussed - or some other method. Can you find any package that helps you to do this? If interactions are suggested by your results, can you visualise them?
- Search online for some interesting tutorials on model interpretability and see if you can follow any of the ideas on our data.

vowpal wabbit

- Adjust the code provided in the notebooks to create data for vw in the correct format.
- Train vw with some reasonable default parameters.
- Make a note of the validation loss.
- Create predictions on test and submit on Kaggle. Compare your Kaggle result to your validation loss. It should be similar. If it is not - it may be hard to understand why but you should at least note the difference.
- There are various ways you could do h-p training. You could turn the above process into a loop and do a grid-search over some of the key parameters (the speed of vw should make this possible). Alternatively there is at least one set of code online which claims to use hyperopt - getting this to work would be impressive.
- How does your best vw model compare to your Part 3 model? Which is better? Why is it better? - after all, they are both penalised logistic regression models.