6CCS3ML1 (Machine Learning)

Coursework 1 FAQ

(Version 1.3)

- *Running 'python3 pacman.py -p ClassifierAgent' with a clean copy of the downloaded file throws the exception: "The agent ClassifierAgent is not specified in any *Agents.py.". Looking at the code in pacman.py it looks like trying to import classifierAgent is throwing an ImportError. How can we resolve this?*

  Ensure you run everything in the environment we have set up in the lab machines (see coursework document). This is the standard Anaconda Python 3 distribution, with `scikit-learn` also installed (and the `scikit-learn` distribution includes `numpy`). If you run this on your own machine, ensure you use the same environment. For the error, see also Section 2.2 of the coursework document.

- *"Code for a sophisticated classifier included (i.e. student has created a sophisticated classifier, not just called one from the library.) (20 marks)". What classifiers that we have learnt are considered sophisticated?*

  We are assessing your ability to identify sophisticated models, so we cannot answer this. But we can give you examples of what we do not consider to be sophisticated: naive bayes, knn, and ensembles of these.

- *Let's say that a naive bayes algorithm using scikit-learn (and/or an implementation made by the student alone has made the algorithm more robust) has been implemented that is sophisticated somewhat. What would this return as a final mark? Let's also assume that the code is well written and documented (extra 20 marks). Could this result in 60 marks awarded?*

  We will not assess your coursework, whether it is presented as "I did this", or as a hypothetical, in any way at all, until it has been submitted and the coursework deadline date has passed.

- *I have implemented a Naive Bayes classifier and combined 2 other sklearn classifiers to make an Ensemble Classifier. The code is below 100 lines but I believe I have satisfied all the requirements. I have written a classifier, it is sophisticated. Is it however sophisticated enough?*

  Similarly, we do not give feedback (which is what you are asking for) before you submit your work.

- *Are we allowed to implement a classifier that has not been covered in lectures?*

  Yes, that is up to you.

- *Should we just train in our own environment and just load the parameters to the classifier (i.e. do not show the training process.)*

  No, you need to include everything.

- *Runs on a different training set – does this simply mean different versions of good_moves.txt, and not different format of the input data?*

Correct.

- *I am confused about how we are supposed to show that we used different training data, if we are only submitting the classsifier.py file?*

  You are not supposed to show this per se. You just need to ensure that your code runs on different training sets (we have given you code to generate additional data). Then, we will be doing the testing ourselves with our own training files to check whether your code works.

- *I am concerned about the runtime when a large dataset is provided. For the initial dataset, K-fold takes a few minutes, but the prediction bit performs well. However, with much larger datasets (in the neighbourhood of thousands of rows), it takes too long to run k-fold and even the agent takes a few seconds between actions. Will this be counted against me?*

  We will test your code on similarly sized datasets as the one you have been given, so that should not be an issue. Once your model is trained, however, it should be quick in making decisions.

- *From what I understand, each feature vector represents information about the game, and the corresponding move performed at this state. However, are the feature vectors in good-moves.txt from a single continuous run of a Pacman game in sequential order, or do they not represent an actual game but just a bunch of moves from games or that have been generated? And I would also like to know the same for the data set that we will be tested on, if they will be continuous or not.*

  That information is not relevant for the task you are required to solve — which is to build a classifier to determine what to do, not to build any kind of sequential model. If it is helpful for whatever you are doing, the best answer that I have is "probably".

- *I noticed that for some grids, the number array representing features may differ (length 17 for smallGrid while 25 for mediumClassic). So just to confirm, do we need to make our agent work on any grid, or only those having 25 numbers as features?*

  You are right that the length of the feature vector depends on the grid, or more accurately on what the Pacman code calls the "layout", which is a combination of grid size and number of ghosts. Since the feature vector includes a bit for each ghost, layouts with less ghosts will have smaller feature vectors.

  None of which needs to concern you. All the datasets that your code will be tested on will be based on mediumClassic, and so will have the same number of features as the examples in good-moves.txt.

- *Can we use functions from numpy: mean, zeros, etc.? (Will we lose mark for using these?)*

  The only limitations on what you can use in the coursework are written down in Section 3.3 of the coursework document.

- *I know that we will lose marks if we simply use a classifier from a library, but are we allowed to utilise stuff from scikit learn that are not classifiers? E.g. the train_test_split function, or other useful things. Or will we need to implement all the functionality ourselves in order to gain top marks?*

  The way I read your question, you are asking two things: 1) Can I use functions from scikit_learn? 2) Will I lose marks if I use a function from scikit_learn rather than writing

it myself? The answer to the first question is given in the coursework specification section 3.3 (b) — you can use code from external libraries. The answer to the second question is derived from a general principle that underlies all the marking for this module (and broadly for all modules) — you get credit for work you do and you don't get credit for work you don't do. So, to set this in the context of your question, lets assume you write a function A which calls another function B. You will get credit for writing function A. If you also wrote B, then you will get credit for writing B. If B comes from a library, then you will not get credit for writing B. Thus, to a first approximation, the total amount of credit will be more if you write both A and B. However, this has to be balanced with other considerations. You don't get more credit for writing lots and lots of lines of redundant code (because you should be better than that) so more code does not necessarily equal more marks, and if you use libraries (as I am sure you know) you can typically produce code that has more functionality for the same effort, and that additional functionality gains credit as well. So, there is not a simple answer — like much of life, there are lots of considerations, and it is not possible to give a completely clear cut answer.

- *If we decided to implement our own classifier, how should we go about citing sources/references? Is a README allowed in those cases, or does everything have to be contained within the python file?*

  Please add any citations, or descriptions, whatever you want us to know in the python file. I appreciate that you would normally use a README for that, but it is much easier to mark if you put in the code (and with >280 students, "easier" is much appreciated and worth working towards).

- *In the CW description it says "you will only be credited for your own work", however I believe very few people in the field actually come up with their own classifiers. Will using existing classifier algorithms lower my CW grade because it is "not entirely my own work"?*

  You get credit for what you do, and you get more credit the more you do. Coding an existing classifier algorithm, lets call it C, is more work than just calling an implementation, call it S, from scikit, so you would get a better mark for handing in code that uses C than one that just uses S. Creating an ensemble that combines a call to C and a call to S is more work than doing either on their own, so would get a better mark than either. Writing a whole new approach to classification, would be more work than that, so would get a better mark. And so on. As you suggest, I don't think anyone is likely to come up with whole new approach to classification (and if they do, it is probably worth more than a few coursework marks, more like a PhD).

- *I have implemented my own classifier despite the performance (accuracy) is disappointing. However, I have read the coursework description and marksheet several times, it is unclear that whether we should have a high performance classifier. It only says we should use classifier to make decision, to get more marks, we should implement sophisticated classifier by ourself. Is it OK if I implemented a classifier but it has low accuracy (due to limited data)?*

  The relevant section of the coursework description is Section 3.1 (e), which says: "To get full marks, your code has to run until either Pacman wins a game, or until Pacman gets eaten by a ghost (and loses a game). In other words, your code should not crash or otherwise fail while we are running it. Losing a game is not failing. In fact, from the point of view of marking, we don't care if your Pacman wins, loses, gets a high score or a low score. We only care that your code successfully use a classifier to decide what to do." From the absence of any remark about accuracy, one might infer that classifier accuracy is not important here. That

would be correct. What we care about is not whether the classifier provides good accuracy on an arbitrary dataset, but whether the classifier can help Pacman choose moves. (In addition, we made the coursework easier by not caring whether the moves are good moves). The idea that accuracy is not an important consideration is not restricted to this coursework. In many applications ML is part of a larger system, and what we care about is not the performance of the ML component per se, but the performance of the wider system. System performance may be strongly correlated with ML accuracy, but it also may not.

- *Should we check if the action chosen by the classifier is legal or not, so pacman doesn't get stuck at a wall, due to an illegal move, or completely leave it to the classifier.*

  That is up to you. Remember that your aim is to create a Pacman controller that does not crash — so you need to make sure that whatever the classifier returns, it is not an instruction that will cause the game to crash. If you use the API/code correctly, it should not send an illegal move to the game engine, but I suggest that you test your code very carefully.