COSC401-2019S2 - Assignment 1

Due date: Friday 23 Aug 2019 at 5pm

General notes and instructions

- This assignment is marked out of 10. It contributes 22.5% towards your final grade.
- The assignment must be submitted electronically on Learn.
- You must submit one (and only one) document which has to be an IPython notebook (`.ipynb`) containing all your code, graphs, and discussions.
- Do not submit any data file. If a question requires you to generate some data, it must be done by a program and the program must be included in the IPython notebook.
- You can assume that packages such as `numpy`, `sklearn`, `statsmodels`, `seaborn`, `graphvix`, etc are installed on the machine that will be used to re-run your programs.
- Make good use of libraries in Scikit-learn[1] and pandas[2].
- Your program must not write into disk, perform network activities, or anything other than what is being asked in the assignment.
- For all the questions, you must include the related programs in the notebook.
- The readability and style of the document (programs, outputs, plots, discussions, etc) are all important and will have allocated marks.
- At no point in this assignment you are required to write lengthy descriptions. Keep your text succinct but rich and meaningful. Think of using tables, figures, or equations to express your thoughts and ideas.
- Set the random seed at the beginning of the document so the results can be reproduced.

[1]`https://scikit-learn.org/stable/`
[2]`https://pandas.pydata.org`

- What you submit must be your own work, however, you are allowed to reuse code or content from publicly available and indexed sources as long as you cite the said source if you use more than a few lines of code or content. In such cases, you must clearly identify which parts are your work.
- You are not allowed to give or take (directly or indirectly) code or content to or from another student in this classes or other classes if you or someone else will be receiving credit for it.

1 Tuning Hyper-parameters [3 marks]

Many machine learning algorithms have *hyper-parameters.* These are parameters that must be specified before running the search/optimisation process that constructs a model. The value of hyper-parameters influence the quality and performance of the constructed model. The process of tuning heyper-parameters is called model selection.

A common hyper-parameter for decision-tree (DT) learning algorithms is *maximum depth.* In this task you have to tune this hyper-parameter in order to make DT models for the following problems[3]:

- Iris plant recognition;
- Breast cancer diagnosis; and
- Digit recognition.

Suppose the goal is to maximise the performance (accuracy) and intelligibility (i.e. minimise complexity). You need to have a reasonable and measurable definition for complexity. One idea is to use the number of nodes in the tree as a measure of complexity. You also need to come up with a rule that specifies under what condition you are willing to sacrifice model size for performance. For example, "the maximum depth can be increased by 1 if the increase in accuracy on validation set is 1% or more."

Divide the data into training and test sets. Put the test set aside for the very end, when you measure the performance of the final constructed model. Do not use the test set for learning or tuning parameters. Since the training set may not be large enough, use k-fold cross validation to determine the best value of maximum depth in each problem. Pick a reasonable fixed k, for example a value between 6 and 10. [4]

Consider a range for maximum depth (from 1 to some reasonable number). For each problem, display a table (a dataframe) that for each depth shows (average) number of nodes and accuracy. Select the best value of hyper-parameter

[3]These data sets are available in `http://scikit-learn.org/stable/datasets/index.html#toy-datasets`.

[4]Some tools here may be useful: `http://scikit-learn.org/stable/model_selection.html`.

and then use the entire training data to build a classifier. Report the number of nodes and the accuracy of the classifier on the test set.

2 Decision trees vs linear models [**4 marks**]

Different hypothesis sets ($\mathcal{H}$) are good for different problems. A hypothesis set (together with a learning algorithm) that performs well on one class of problems, may not do well on another class of problems.

In this question, you must construct a binary classification problem in $\mathbb{R}^2$ such that the problem is easily solvable with logistic regression (or Perceptron) but the decision tree learning algorithm will have to construct a deep tree to solve the problem.

- Construct the problem by defining a function (formula) that decides the class label of a point in $\mathbb{R}^2$. This is the true f which is typically unknown but not here since you are defining the problem and generating the data. You must then generate some training and test data in $\mathbb{R}^2 \times \{P, N\}$, where P and N indicate the positive and negative class respectively. Visualise a sample of points.
- Show how various tree sizes (from shallow to deep) perform with different training sizes. Iterate over a range tree depths and training data sizes and report the test accuracy in a table.
- Provide a visualization of the actual and learned class boundaries (for a single tree). The visualisation must demonstrate the issue.
- Very briefly (in a few sentences) discuss what would happen if the problem was $\mathbb{R}^d$ with a large d.

3 Regression on mixed data types [**3 marks**]

You are given a data set for a regression problem. You must train a linear regression model for the problem[5]. The model must be linear both in weights and features. The problem has a categorical feature with more than two levels. You must use an appropriate method to handle it. Use the entire data set for training. Provide the following:

- Print the mean squared error (MSE) and R^2.
- Round the coefficients of the learned model to integers and write a case-based definition of the functions identified by the regression model(s).

[5] See `http://scikit-learn.org/stable/modules/linear_model.html` or if you are familiar with R see `https://www.statsmodels.org/dev/example_formulas.html`

Each case must be a linear expression of the input variables (one expression per level of the categorical variable). For example:

$$y = \begin{cases} w_{1,1}x_1 + w_{1,2}x_2 + w_{1,3}x_3 + b_1, & c = l_1 \\ w_{2,1}x_1 + w_{2,2}x_2 + w_{2,3}x_3 + b_2, & c = l_2 \\ w_{3,1}x_1 + w_{3,2}x_2 + w_{3,3}x_3 + b_3, & c = l_3 \end{cases}$$

where $w_{i,j}$ is the integer coefficient of the j-th numerical feature corresponding to the i-th level in the domain of the categorical feature; b_i is the intercept in each case, and l_i is the i-th level in the domain of the categorical feature c.

- Depending on the method you used for regression, described how a new (unseen) data point (an input vector of size 4) can be assigned a predicted value.