Computing Project Proposal

Problem:

When wanting to choose the right machine learning algorithm for a classification job it is often difficult and time-consuming. The reason for this is that to find the best algorithm you need to build and test all available ones. In addition to this, you must refine the data and only keep features that show the best relationships and produce the greatest accuracies. On top of that, you then need to find the best hyperparameters to increase accuracy and avoid overfitting. Furthermore, many people often default straight to using neural networks as they've recently gained massive popularity through the success of the use in deep learning in self-driving cars and image recognition. [1] [2] Neural networks although powerful can become very complex when choosing the right hyperparameters, human means can be very unintuitive. [3] Then if there are simpler algorithms that can produce similar accuracies, we surely should use them or at least use automated hyperparameter selection.

Solution:

I propose a program that when given a set of raw labelled data it will choose the best features, the best algorithm in terms of complexity vs accuracy and then tune the proposed algorithm's hyperparameters for you. The program should not only save time but achieve better classification results than found manually.
To produce the best feature set the program will:

- Deal with missing data with a mixture or one of the following techniques: by removing whole instances, replace missing values with the class mean, using values from similar instances. A fall-back approach could also be to use logistic/linear regression to find missing values by designating them as an output and using similar instances as training variables. [4]
- Refine the features by using a mixture or one of the following techniques: remove features that share high correlation with other instances, remove features that share high mutual information with other features, remove features with low variance.

Next, the algorithms will be sorted in order of theoretical suitable for the data in terms of complexity and accuracy. The algorithms I will use are: Decision Tree, Random Forest, Logistic Regression, K Nearest Neighbour, Support Vector Machine (Radial Based Function) and Artificial Neural Network. Finally, the individual algorithms will be tested in the sorted order using cross-validation and parallel programming to save time. The best hyperparameters will be found by using either random search, grid search or Bayesian optimisation depending on the algorithm. [5] Testing of the algorithms will be able to be stopped when a user inputted accuracy is reached. All results will be published when found immediately.

Plan:

In the first stage of my project will research and conduct tests in pythons Keras library on the following:

- The best combination of feature removal techniques and testing if they are algorithm dependent.
- The types of data that works best for each algorithm and why.
- The time and space complexities of the different algorithms.
- The most effective method for hyperparameter selection for each algorithm.

The second stage of my project I will build a prototype in Keras using the information I found from testing. The datasets that I will use to test the program are seen in figure 1. I have chosen

datasets of various sizes and number of features so ensure my program doesn't generalise to a specific type of dataset. I will perform manual tests on the datasets to find the optimum features, algorithm and hyperparameters while using guidance from papers that have used these. The performance measures I will use are a confusion matrix to produce accuracy, precision, recall and F1 score. I will then test and change my program so that it automatically meets the results I found manually or exceeds them. I also plan to record the time taken to achieve the optimum results manually and then automatically. As I want my program to be much quicker than the manual process because the main aim of it is to save time.

Figure 1

Dataset	Instances	Features
Soybean	302	35
Mushroom	8124	22
Census Income	48842	14

The final stage of my project will involve me writing the production level program in C++ with unit tests. Parallel programming and other optimisation techniques will also be used in this stage. I will then design a basic but functional user-interface and perform validation testing. The datasets that I will use to evaluate the program are in figure 2. During validation I will repeat my steps during the first testing phase, but this time expect the results to match my manual findings without any refinement to my program.

Figure 2

Dataset	Instances	Features
Breast Cancer Wisconsin	699	10
Wine Quality	4898	12
default of credit card clients	30000	24

Risks:

- Testing and finding the correct hyperparameters for more complex algorithms such as neural networks could take too much time. In this case I can perform tests to get a table of approximate time values depending on the size and complexity of data that will be presented to the user.

- The tuned algorithm proposed could overfit to the initial raw data given. To overcome this an abundance of data must be given so that after cross-validation has found the best algorithm, validation testing can be done with the extra data.

- The best algorithm given to the user might be good right now but if the users next round of data collection contains a few more new features it may not be able to adapt. To solve this the program could allow the user to save their results from their previous run. The program can then use these results to find the best algorithm quicker.

References:

1. Yuchi Tian, Kexin Pei, Suman Jana, Baishakhi Ray. (2018). DeepTest: Automated Testing of Deep-Neural-Network-driven Autonomous Cars.Available: https://arxiv.org/pdf/1708.08559.pdf. Last accessed 09/12/2018
2. Google. (2018). Neural Networks Search Trends. Available: https://trends.google.com/trends/explore?date=today%205-y&q=neural%20network. Last accessed 09/12/2018.
3. Sean C. Smithson, Guang Yang, Warren J. Gross, Brett H. Meyer. (2016). Neural Networks Designing Neural Networks: Multi-Objective Hyper-Parameter Optimization. Available: https://arxiv.org/pdf/1611.02120.pdf. Last accessed 09/12/2018
4. Tomasz ORCZYK, Piotr PORWIK. (2013). INFLUENCE OF MISSING DATA IMPUTATION METHOD ON THE CLASSIFICATION ACCURACY OF THE MEDICAL DATA. JOURNAL OF MEDICAL INFORMATICS & TECHNOLOGIES. 22 (0), 111-116.
5. Katharina Eggensperger, Matthias Feurer, Frank Hutter. (2013). Towards an Empirical Foundation for Assessing Bayesian Optimization of Hyperparameters. Available: https://www.cs.ubc.ca/~hoos/Publ/EggEtAl13.pdf. Last accessed 09/12/2018.