CS5011 Artificial Intelligence Practice Assignment 4 Report

University of St Andrews - School of Computer Science

Student ID: 150014151

20th December, 2019

Contents

1 Introduction

The following requirements were attempted:

- Basic Agent:
  - Multilayer feedforward neural network.
  - Data encoding and training/testing split.
  - Training/testing result visualisation in plots and heatmaps.
  - Grid search algorithm for determining optimal hyperparameters.
- Intermediate Agent:
  - CLI text-based interface.
  - Early predictions.
  - Neural network re-training.
- Advanced Agent:
  - Decision tree classifier.

1.1 Installation

Create a virtual environment for the project and activate it:

```
virtualenv ~/Environments/A4
source ~/Environments/A4/bin/activate
```

Once you have the virtual environment activated, *cd* into the project directory and install the requirements needed to run the app:

```
pip install -r requirements.txt
```

1.2 Usage

To compile the program, navigate to the *A4src* directory and run the following command:

```
python A4Main.py [-h] -a <AGENT> -c <CSV> [-g] [-d]
```

where:

- *AGENT* is the type of agent to run: *[Bas, Int, Adv]*:
  - *Bas*: Train and test the neural network with the optimal parameters, or run the Grid Search algorithm to determine the optimal parameters.
  - *Int*: CLI text-based application to submit a new ticket and predict to which response team it should go.

  - *Adv*: Train and test a decision tree classifier.

- *CSV* is the CSV file containing the data used to train/test the data.
- *-g*: is a flag that run the grid search algorithm when set.
- *-d*: is a flag that enters *debugging* mode, printing more statements to the command line, when set.
- *-h*: is a flag for printing help on how to use the program.

Examples Here are a few examples that can be used to run the program:

- "*python A4Main.py -a Bas -c tickets -d*" to train/test the neural network.
- "*python A4Main.py -a Bas -c tickets -g*" to run the grid search algorithm.
- "*python A4Main.py -a Int -c tickets*" to submit a new ticket through the CLI text-based interface.
- "*python A4Main.py -a Adv -c tickets*" to train/test the decision tree.
- "*python A4Main.py -h*" for help on how to run the agent.

1.3 Tools Used

- Scikit[1] and related Python libraries (e.g. NumPy[2], Pandas[3], Matplotlib[4]).
- PyCharm[5]: an IDE developed by JetBrains to write Python code with support for some of the above libraries.
- Git and GitHub: to back and version control the code.
- PEP8[6] coding guidelines and docstring followed throughout the entire code.

[1]Scikit: `https://scikit-learn.org`
[2]NumPy: `https://numpy.org/`
[3]Pandas: `https://pandas.pydata.org/`
[4]Matplotlib: `https://matplotlib.org/`
[5]PyCharm: `https://www.jetbrains.com/pycharm/`
[6]PEP8: `https://www.python.org/dev/peps/pep-0008/`

2 Design, Implementation & Evaluation

2.1 Design: PEAS Model

This section defines the PEAS model for a ticketing routing-based agent that uses a multilayer feedforward neural network in order to learn how to predict the appropriate response team for future tickets.

Performance measure The basic agent's training efficiency can be evaluated with the error loss curve, and the testing accuracy by using a confusion matrix. The intermediate agent can be assessed by the correctness of its predictions.

Environment This is a single-agent and fully-observable environment represented by a neural network and the text-based interface used to interact with a user logging a new ticket. Additionally, it can be said that the basic agent environment is deterministic (the next state is determined by the current state and the action executed) and stochastic for the intermediate agent (user input is unknown), according to the definitions set by Russell & Norvig in *Artificial intelligence: a modern approach* [2].

Actuators The agent may accept input and target data it can learn from by backpropagating the calculated error between the input and target data in order to adjust the weights of the neural network (Stochastic Gradient Descent). It may also predict an output based on unseen data.

Sensors The basic agent is always aware of the environment and can calculate feedback error based on input and target data, while the intermediate agent can receive responses to questions by interacting with a human user.

2.2 Implementation

2.2.1 Project Structure

The project is divided into three main sections:

- The *agents* module, containing functions to implement the execution of flow of each agent.
- The *neural_network* module, which contains classes to create and manipulate a multilayered perceptron, a data encoder/processor and the grid search algorithm.
- The *main* section, containing the program's entry point *A4Main.py* for parsing the command line arguments, global configuration settings in *config.py* and printing methods in *helpers.py*.

The *data* directory stores the CSV data, the *neural_networks* directory the trained neural nets in *"*.joblib"* format, and the *results* directory the various output of the agents. See Appendix A for the full project structure.

2.2.2 Neural Network Training & Testing (Basic Agent)

The basic agent takes a CSV file as input to be used for training and testing. It instantiates the custom *DataProcessor* class containing functions for parsing the CSV file (retrieving tags and categories) and splitting the data between inputs and targets before encoding/decoding it. When being passed around, the data is often held in a *DataFrame*[7], which allows for more manipulation than Python's built-in *list*. The input data is encoded in either 0s or 1s, while the target data is encoded using a one-hot encoding. Figure 1 depicts how the data is split and encoded.

Input (tags)									*Output (categories)*
Request	Incident	WebServices	Login	Wireless	Printing	IdCards	Staff	Students	Response Team
No	Yes	Yes	Yes	Yes	No	No	No	Yes	Emergencies
No	No	No	No	Yes	Yes	No	No	Yes	Credentials

Input (tags)									*Output (categories)*
Request	Incident	WebServices	Login	Wireless	Printing	IdCards	Staff	Students	Response Team
0	1	1	1	1	0	0	0	1	[0 0 1 0 0]
0	0	0	0	1	1	0	0	1	[1 0 0 0 0]

Figure 1: Encoding of the data. At the top, the decoded data; at the bottom, the encoded data.

The data must be fed to the neural network in binary form. One-hot encoding is therefore chosen as it suits the sparse representation of the data, which is made up of only five target categories. Only a single digit may have the value 1 in one-hot encoding, while the others remain at value 0 [3]. The one-hot encodings of the target categories can be seen in Figure 2.

String Category	One-hot encoding
Credentials	1 0 0 0 0
Datawarehouse	0 1 0 0 0
Emergencies	0 0 1 0 0
Equipment	0 0 0 1 0
Networking	0 0 0 0 1

Figure 2: One-hot encoding of the categories.

Once the data is encoded, an instance of the custom *MultiLayerPerceptron* class is created, containing functions to split the training/testing data, training & testing the neural network, displaying results and saving the network with the trained weights. This custom class allows it to be easily re-used in the intermediate agent, thus avoiding code duplication. The class uses *Scikit*'s *MLPClassifier*[8] to implement the neural network and is instantiated with the optimal

[7]Pandas DataFrame: `https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.html`

[8]Scikit MLPClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html`

hyperparameters determined in Section 2.3.1, which can be overwritten when creating a new *MultiLayerPerceptron* (see Listing below).

```
class MultiLayerPerceptron:
    def __init__(self, name, input_data, target_data, hidden_layers_size=(15,),
     solver="adam", activation_function="logistic", learning_rate_init=0.6,
    momentum=0.9, optimisation_tolerance=0.0001, num_iterations_no_change=1000,
    max_iterations=10000, verbose=config.debug):
```

An 80%/20% split is used for the training/testing data sets, with equal category distribution maintained between the training and testings sets, which is ensured through the use of the *stratify* option in Scikit's *train_test_split* function[9]. Because the data has 250 even entries (50 for each of the five outputs), the testing set will therefore have 10 entries for each of the five categories[10].

The actual training is carried out using the *fit* function, with the *error_loss_* plotted with regards to the number of epochs; while the testing is performed using the *predict* and *predict_proba* functions and its accuracy visualised with a confusion matrix (counting the number of true positives/negatives and false positives/negatives) in a heatmap. These results can be visualised in Section 2.3.2.

2.2.3 CLI-Based Ticketing-Routing Agent (Intermediate Agent)

The intermediate agent is built on a CLI[11] as an interactive text-based application allowing a human user to submit a new ticket. The agent records the *"Yes"*/*"No"* answers to each tag to create a new ticket, which is then used as input for the previously trained neural network to make a prediction. The neural net is loaded using *joblib* and tested using the *predict* and *predict_proba* function.

To avoid the repetitive task of answering all questions, the agent makes early predictions after the third, fifth and seventh questions. The prediction is made by filling out the missing fields in the ticket with the most common value (*mode*) in the original CSV data.

After an early prediction is made, the user can state whether he is happy with the result or not. If he is not, the agent asks more questions until the next early prediction or until all questions have been answered. Finally, if the user is not happy with the final prediction, he can specify the desired response team to send the ticket to (only if he has answered all questions and entered *"No"* to every early prediction). This information is added to the end of a copy of the original CSV data file that is used to re-train the neural network in a similar fashion mentioned in Section 2.2.2. The newly trained network is then saved and used for future predictions.

[9] Scikit train_test_split: `https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.train_test_split.html`

[10] 250 data entries, 50 per response team, 80%/20% data split so 10 of each end up in the training set.

[11] Command Line Interface

2.2.4 Decision Trees (Advanced Agent)

Decision trees are implemented using Scikit's *DecisionTreeClassifier*[12]. Most of the code used in the previous agents is re-used for this classifier: the decision tree is trained and tested using the same functions mentioned in Section 2.2.2.

In brief, decision tree classifiers are similar to flowcharts: they divide the data set into smaller sets based on filtering criteria (*"Yes"*/*"No"* answers to each tag) until each set only contains one class. The splitting is carried out based on the Gini score, a measure indicating the effectiveness of the split: if the Gini score is 0, then the resulting split successfully isolated all inputs from one class [1]. A minimum sample split of 20 is set to avoid overfitting the data (the tree would become too deep). A pair of diagrams representing the trained decision tree is generated using the *Graphviz*[13] tool, and can be found in Appendix B.

2.3 Evaluation

2.3.1 Optimal Hyperparameters

To find the optimal hyperparameters to train the neural network, a grid search algorithm is implemented. The algorithm exhaustively trains and tests the neural network with every combination from a pre-defined set of parameters and measures its accuracy. The parameters tested include the number of neurons in the hidden layer, the solver, the activation function, the learning rate, the momentum, the tolerance and the maximum number of iterations after which training stops if the error does not improve by the tolerance. These can be found in the Listing below:

```
grid_search_params = {
    'hidden_layer_sizes': [(3,), (5,), (7,), (9,), (15,), (25,)],
    'solver': ["sgd", "adam"],
    'activation': ["logistic", "relu", "tanh"],
    'learning_rate_init': [0.05, 0.1, 0.2, 0.4, 0.6, 0.8, 1.0],
    'momentum': [0.1, 0.3, 0.5, 0.7, 0.9],
    'tol': [1, 0.1, 0.01, 0.001, 0.0001],
    'n_iter_no_change': [100, 1000],
    'max_iter': [10000]  # Parameter not being tested.
}
```

Overall, 12,600 unique combinations of parameters were tested in 1h26m. From these tests, 14 combinations of hyperparameters with a +98% mean accuracy (over 5 runs for even 80%/20% data splits) emerged as optimal, as depicted in Figure 3. These optimal parameters can be found in more detail in Appendix C.

[12]Scikit DecisionTreeClassifier: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`

[13]Graphviz: `https://graphviz.org/`

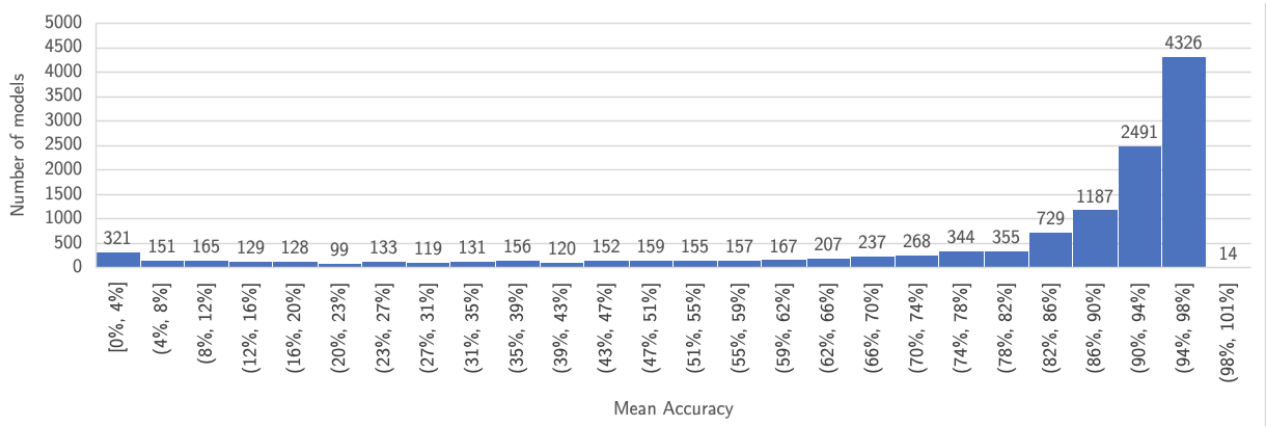


Figure 3: Distribution of the accuracies of the 12,600 combinations of hyperparameters tested with the grid search algorithm.

It is interesting to note that over half of these tests result in accuracies ranging between [90%-98%[. This is likely due to the fact that there is very little data to begin with (only 250 entry points), meaning that unless the chosen hyperparameters are very poor performance-wise (e.g. 3 hidden units with momentum 0.1 and tolerance 1, resulting in a mean accuracy of 5.5%), the neural network will always train to an acceptable level. The lack of data is also a convincing argument to avoid finding optimal general parameters through validation.

2.3.2 Error Loss and Confusion Matrices

Four optimal combinations of hyperparameters have been chosen for comparison, along with one set of poor hyperparameters to point out differences. These can be found in Table 1.

Activation	**Hidden Neurons**	α	β	**N Iter No Change**	**Solver**	**Tol**	**Mean Accuracy**
logistic	(15,)	0.6	0.9	1000	adam	0.0001	98.5%
tanh	(5,)	0.8	0.1	100	sgd	0.0001	98.0%
logistic	(9,)	0.2	0.9	1000	adam	0.1	98.0%
tanh	(25,)	1	0.1	1000	sgd	0.001	98.0%
logistic	(3,)	1	0.5	1000	adam	0.001	38.0%

Table 1: The 5 neural network hyperparameters used for evaluation, including 4 optimal and 1 sub-optimal.

Plotting the error function with respect to the number of epochs required to reach the designated tolerance gives interesting insights about the training, as seen in Figure 4. Indeed, the error loss curves are all similar when using optimal parameters, quickly converging towards the desired error. A notable difference is the effect of a low momentum $\beta = 0.1$ on the convergence speed, causing the light blue slope (5 hidden neurons) to reach the desired error more slowly than neural networks with high momentum.

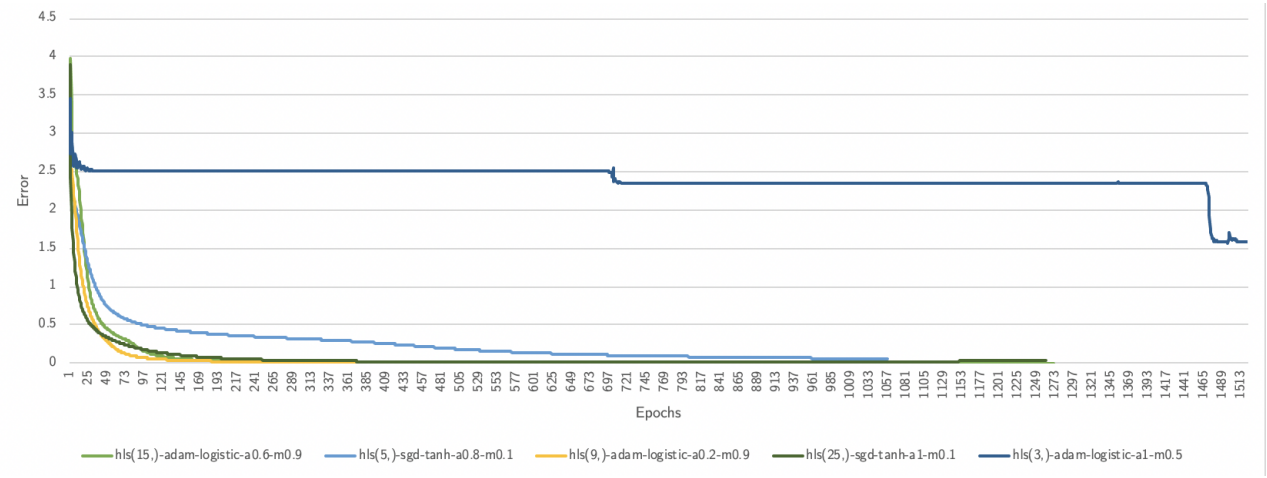


Figure 4: Error loss curves during the training of 5 different combinations of hyperparameters.

However, the most notable difference between training with optimal parameters (accuracy $\geqslant 98\%$) and sub-optimal parameters (accuracy $= 38\%$) is that the error loss curve is less smooth (small peaks can be seen across the curve before epoch 50) due to the high learning rate and lack of neural units (data is under-fitted). The curve is also characterised by multiple plateaus (straight lines with no improvements) and small ravines (minor increases in error), which would be solved with increased momentum.

Once the training is completed, an aggregated confusion matrix over five runs is calculated and plotted in a heatmap for each neural network. This clearly shows the superior accuracy of neural networks using optimal hyperparameters compared to sub-optimal hyperparameters. The results can be seen in Appendix D.

2.3.3 Ticketing Routing Agent

The intermediate agent is evaluated by interacting with the CLI-based agent. The full interaction used for this evaluation can be found in Appendix E. It can be seen that when the user creates a new ticket with *"No"* answered to every question, the agent predicts it should go to *Networking*. The user then suggests sending it to *Emergencies* and the neural network is consequently retrained (lines 49 to 53). When submitting a new ticket with the re-trained neural network, the agent immediately predicts that the ticket should go the *Emergencies* after only three questions (line 62), proving that it correctly learned from its mistake.

However, this is not always the case, as for cases where tags are similar for different outputs, a single team suggestion is not enough to shift the weights of the neural network for the agent to correctly predict the answer for the next iteration. If a dozen or more users all suggested the same correction to the agent, then the weights would change enough to allow the suggested team to be predicted.

2.3.4 Decision Trees

The neural network and decision tree classifiers' accuracy can be directly compared with their confusion matrices, as seen in Figure 5, depicting that the neural network is more accurate than the decision tree. However, it is important to note that the MLPClassifier is being used with optimal hyperparameters, whereas hyperparameters recommended by SciKit are used for the DecisionTreeClassifier.

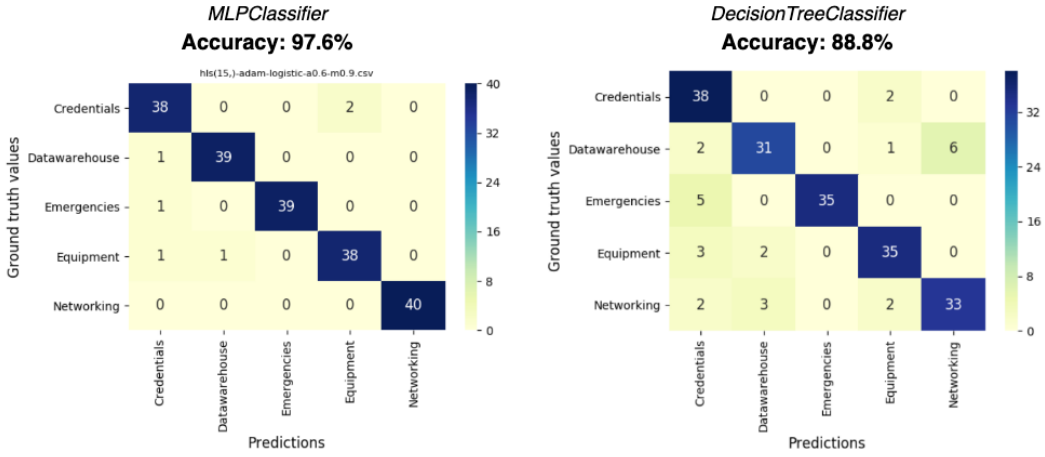


Figure 5: 5-run aggregate confusion matrices for the a neural network classifier and a decision tree classifier.

Despite being less accurate, the decision tree classifier runs on average more quickly than the neural network classifier, averaging at 0.3 seconds in contrast to 0.94 seconds for the MLP. This betrays the key difference between "deep" learning structures: even a neural network with a single hidden layer is more exhaustive than "shallow" learning models like decision trees.

Design, Implementation & Evaluation Section word count: 1978

3 Test Summary

3.1 Basic Agent Testing

The basic agent mainly consists of implementing a neural network using the Scikit library for Python. There is therefore no need to test the functions provided by Scikit. However, one test that can be carried out is running the basic agent and checking that the outputs are correct, namely the error loss curve and the confusion matrix. Running the basic agent using the optimal hyperparameters produces the result seen in Figure 6, indicating that the agent is behaving as intended,

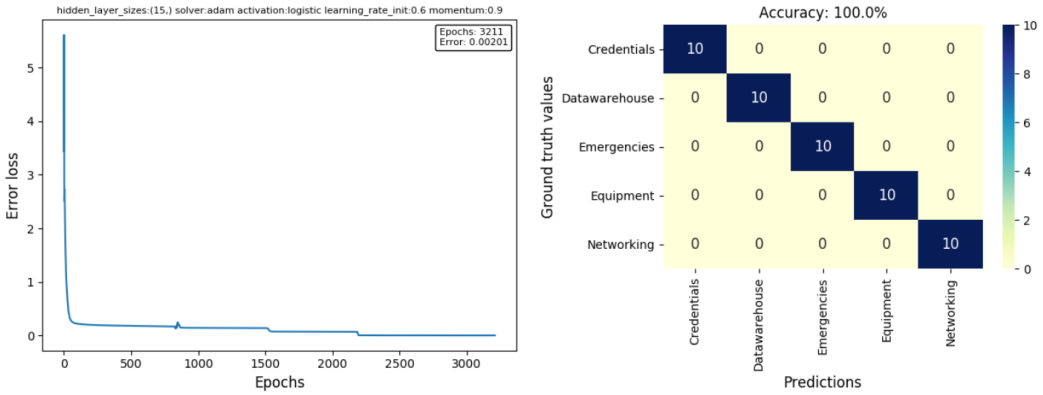


Figure 6: Error loss curve and confusion matrix produced when training/testing the neural network with optimal hyperparameters.

3.2 Intermediate Agent Testing

The intermediate agent's behaviour is tested by ensuring that each prediction (early or not) are correct. This is done by checking that the answers to each question specified in Figure 7 (retrieved from the original *tickets.csv* data file) correspond to the expected output. The green categories indicate a correct prediction. The full interaction between the user and the agent can be found in Appendix F, with each correct prediction highlighted in yellow.

Input (tags)									*Response Team*	
Request	Incident	WebServices	Login	Wireless	Printing	IdCards	Staff	Students	Expected output	Agent Prediction
No	Yes	Yes	Yes	Yes	No	No	No	Yes	Emergencies	Emergencies
No	No	No	No	Yes	Yes	No	No	Yes	Credentials	Credentials
Yes	No	Yes	No	Yes	No	Yes	Yes	No	Networking	Networking
No	Yes	Yes	No	No	No	Yes	Yes	No	Datawarehouse	Datawarehouse
No	No	No	No	No	Yes	No	Yes	No	Equipment	Equipment

Figure 7: Testing table for the intermediate agent, showing correct predictions for each response team.

Regarding the actual functionality of the agent's text-based CLI, edge cases are entered on the command line to show the robustness of the program. The full interaction can be

found in Appendix G. Various inputs are used to answer *"Yes"/"No"* questions, such as *y*, *yy*, *YeS*, *YES*, *No*, *NO*, *nn* and *n* (lines 11 to 28). The same is repeated for the response team suggestion: the agent does not let the user input a wrong team, and will ask the question again if wrong input is given (lines 33 to 37). The user can correctly exit the program at any time by typing *"quit"*.

References

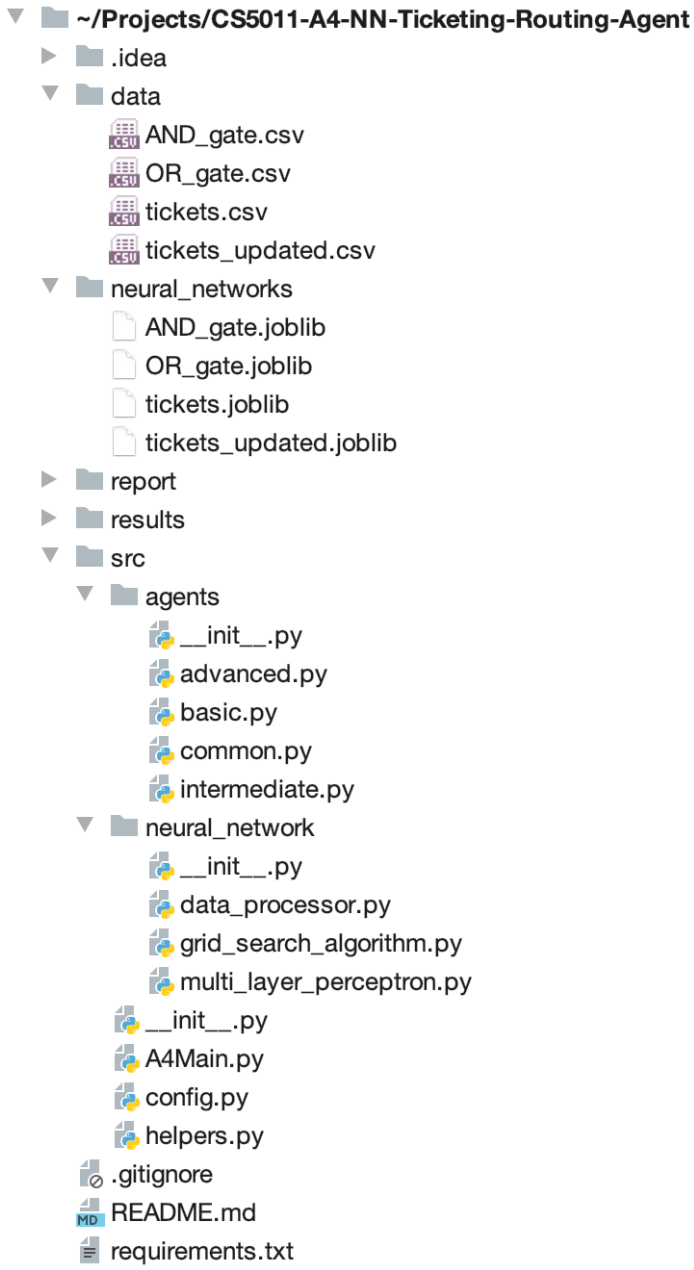
[1] D. Nelso. What is a decision tree? https://www.unite.ai/what-is-a-decision-tree/. [Online] Accessed: 2019-12-20.

[2] Stuart J Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Pearson Education Limited, 2016.

[3] A. Toniolo. Cs5011 - learning part a (slide 36). https://studres.cs.st-andrews.ac.uk/CS5011/Lectures/L16_w9.pdf. [Online] Accessed: 2019-12-19.

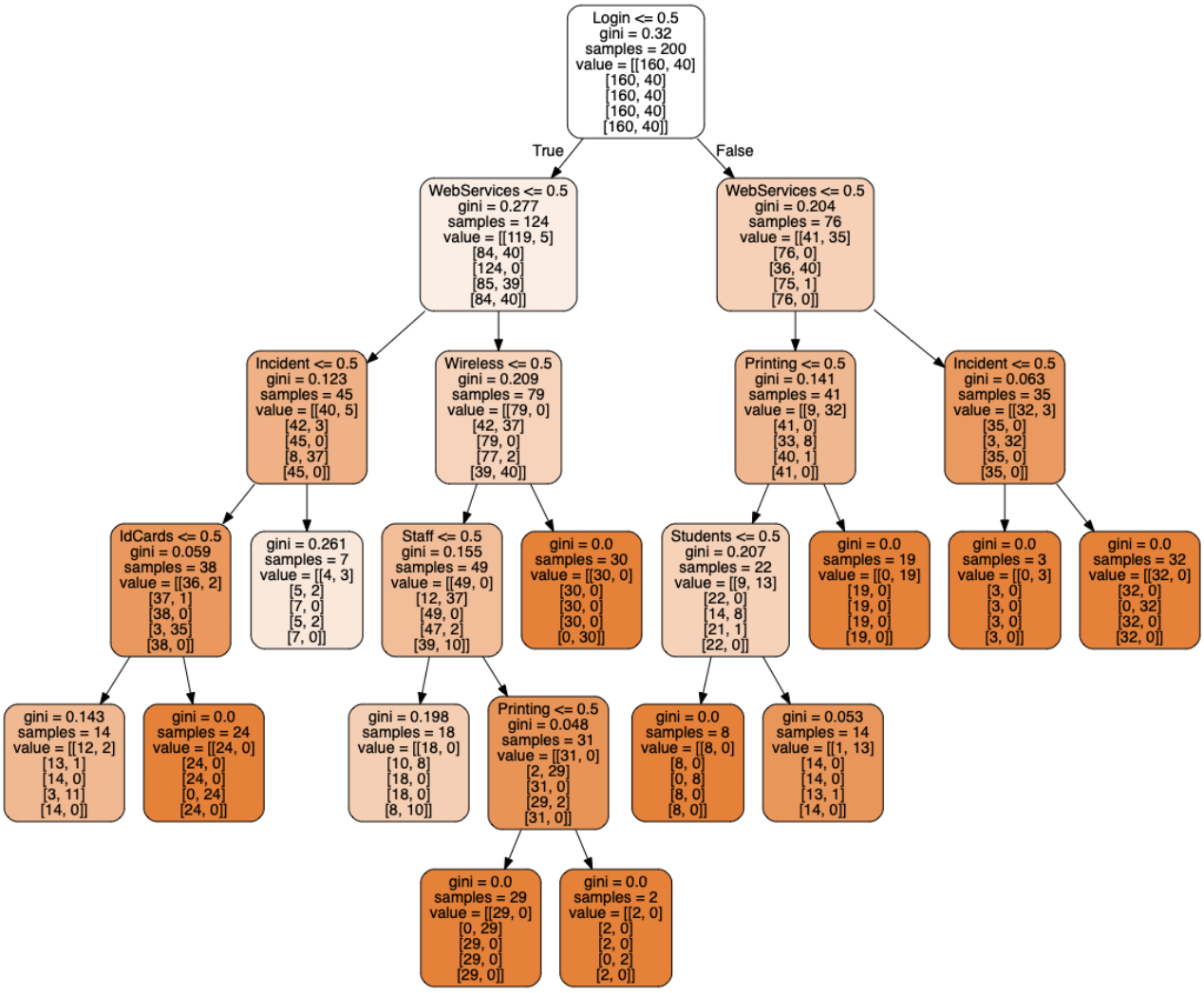
Appendix A Project File Structure

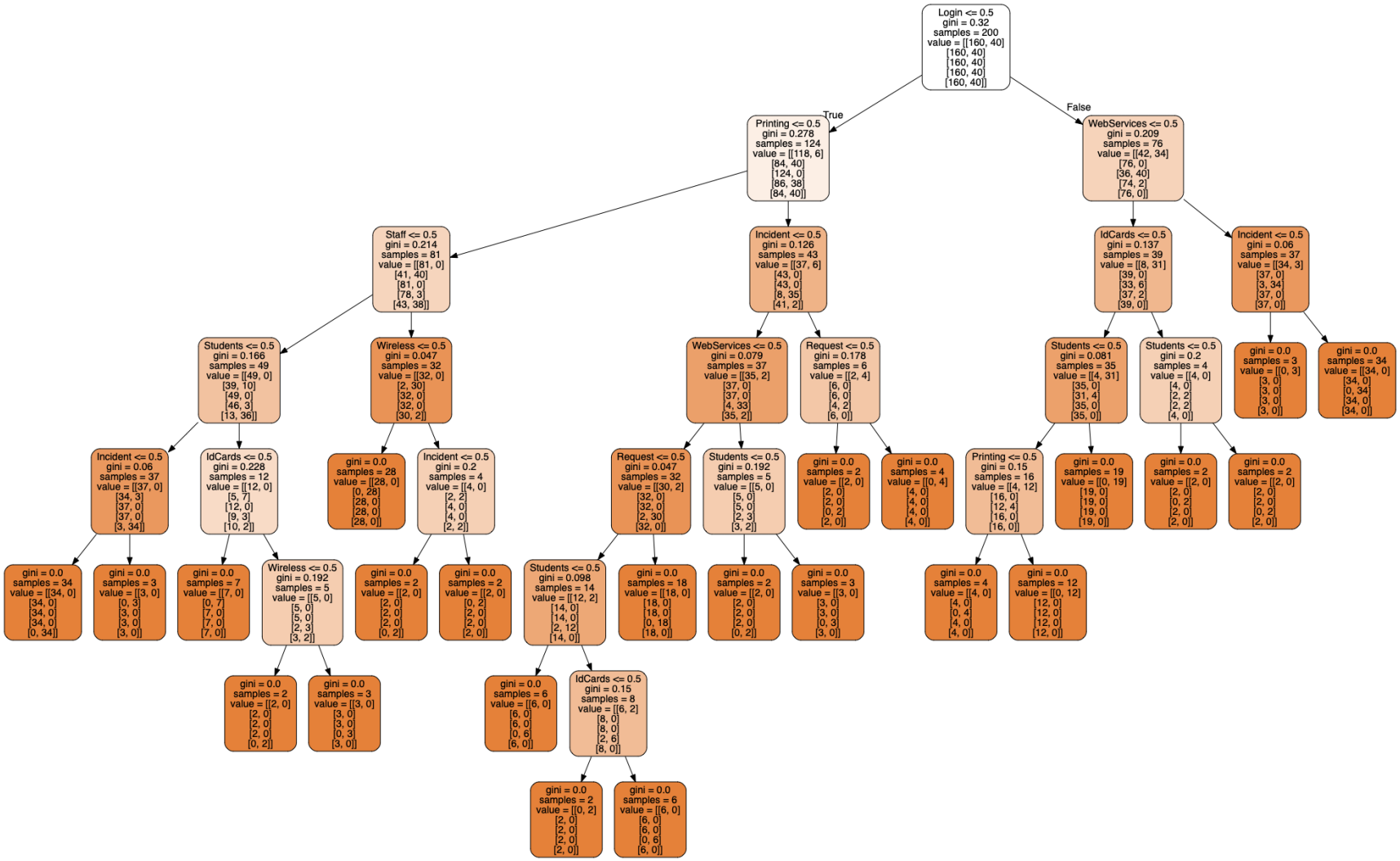
A screenshot of the project's file structure in the PyCharm IDE to illustrate the organisation of the Python modules and the data used.



Appendix B Decision Trees

The trained decision trees generated with *Graphviz*. The first one corresponds to the one used for the evaluation (with minimum sample split of 20), while the second corresponds to an overfitted decision tree ((with a default minimum sample split of 2).




Login <= 0.5
gini = 0.32
samples = 200
value = [[160, 40]
[160, 40]
[160, 40]
[160, 40]
[160, 40]]
True
False
Printing <= 0.5
gini = 0.278
samples = 124
value = [[118, 6]
[84, 40]
[124, 0]
[86, 38]
[84, 40]]
WebServices <= 0.5
gini = 0.209
samples = 76
value = [[42, 34]
[76, 0]
[36, 40]
[74, 2]
[76, 0]]
Staff <= 0.5
gini = 0.214
samples = 81
value = [[81, 0]
[41, 40]
[81, 0]
[78, 3]
[43, 38]]
Incident <= 0.5
gini = 0.126
samples = 43
value = [[37, 6]
[43, 0]
[43, 0]
[8, 35]
[41, 2]]
IdCards <= 0.5
gini = 0.137
samples = 39
value = [[8, 31]
[39, 0]
[33, 6]
[37, 2]
[39, 0]]
Incident <= 0.5
gini = 0.06
samples = 37
value = [[34, 3]
[37, 0]
[3, 34]
[37, 0]
[37, 0]]
Students <= 0.5
gini = 0.166
samples = 49
value = [[49, 0]
[39, 10]
[49, 0]
[46, 3]
[13, 36]]
Wireless <= 0.5
gini = 0.047
samples = 32
value = [[32, 0]
[2, 30]
[32, 0]
[32, 0]
[30, 2]]
WebServices <= 0.5
gini = 0.079
samples = 37
value = [[35, 2]
[37, 0]
[37, 0]
[4, 33]
[35, 2]]
Request <= 0.5
gini = 0.178
samples = 6
value = [[2, 4]
[6, 0]
[6, 0]
[4, 2]
[6, 0]]
Students <= 0.5
gini = 0.081
samples = 35
value = [[4, 31]
[35, 0]
[31, 4]
[35, 0]
[35, 0]]
Students <= 0.5
gini = 0.2
samples = 4
value = [[4, 0]
[4, 0]
[2, 2]
[2, 2]
[4, 0]]
gini = 0.0
samples = 3
value = [[0, 3]
[3, 0]
[3, 0]
[3, 0]
[3, 0]]
gini = 0.0
samples = 34
value = [[34, 0]
[34, 0]
[0, 34]
[34, 0]
[34, 0]]
Incident <= 0.5
gini = 0.06
samples = 37
value = [[37, 0]
[34, 3]
[37, 0]
[37, 0]
[3, 34]]
IdCards <= 0.5
gini = 0.228
samples = 12
value = [[12, 0]
[5, 7]
[12, 0]
[9, 3]
[10, 2]]
gini = 0.0
samples = 28
value = [[28, 0]
[0, 28]
[28, 0]
[28, 0]
[28, 0]]
Incident <= 0.5
gini = 0.2
samples = 4
value = [[4, 0]
[2, 2]
[4, 0]
[4, 0]
[2, 2]]
Request <= 0.5
gini = 0.047
samples = 32
value = [[30, 2]
[32, 0]
[32, 0]
[2, 30]
[32, 0]]
Students <= 0.5
gini = 0.192
samples = 5
value = [[5, 0]
[5, 0]
[5, 0]
[2, 3]
[3, 2]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
gini = 0.0
samples = 4
value = [[0, 4]
[4, 0]
[4, 0]
[4, 0]
[4, 0]]
Printing <= 0.5
gini = 0.15
samples = 16
value = [[4, 12]
[16, 0]
[12, 4]
[16, 0]
[16, 0]]
gini = 0.0
samples = 19
value = [[0, 19]
[19, 0]
[19, 0]
[19, 0]
[19, 0]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
gini = 0.0
samples = 34
value = [[34, 0]
[34, 0]
[0, 34]
[34, 0]
[34, 0]]
gini = 0.0
samples = 3
value = [[3, 0]
[0, 3]
[3, 0]
[3, 0]
[3, 0]]
gini = 0.0
samples = 7
value = [[7, 0]
[0, 7]
[7, 0]
[7, 0]
[7, 0]]
Wireless <= 0.5
gini = 0.192
samples = 5
value = [[5, 0]
[5, 0]
[5, 0]
[2, 3]
[3, 2]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
Students <= 0.5
gini = 0.098
samples = 14
value = [[12, 2]
[14, 0]
[14, 0]
[2, 12]
[14, 0]]
gini = 0.0
samples = 18
value = [[18, 0]
[18, 0]
[18, 0]
[0, 18]
[18, 0]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[2, 0]
[0, 2]]
gini = 0.0
samples = 3
value = [[3, 0]
[3, 0]
[3, 0]
[0, 3]
[3, 0]]
gini = 0.0
samples = 4
value = [[4, 0]
[4, 0]
[4, 0]
[4, 0]]
gini = 0.0
samples = 12
value = [[0, 12]
[12, 0]
[12, 0]
[12, 0]
[12, 0]]
gini = 0.0
samples = 2
value = [[2, 0]
[2, 0]
[0, 2]
[2, 0]
[2, 0]]
gini = 0.0
samples = 3
value = [[3, 0]
[3, 0]
[0, 3]
[3, 0]]
gini = 0.0
samples = 6
value = [[6, 0]
[6, 0]
[0, 6]
[6, 0]]
IdCards <= 0.5
gini = 0.15
samples = 8
value = [[6, 2]
[8, 0]
[8, 0]
[2, 6]
[8, 0]]
gini = 0.0
samples = 2
value = [[0, 2]
[2, 0]
[2, 0]
[2, 0]
[2, 0]]
gini = 0.0
samples = 6
value = [[6, 0]
[6, 0]
[6, 0]
[0, 6]
[6, 0]]

Appendix C Optimal Hyperparameters

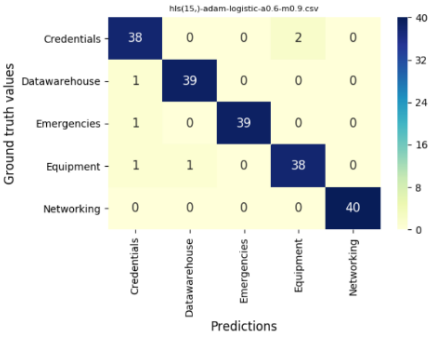
The 14 optimal hyperparameters determined after running the grid search algorithm on 12,600 combinations.

Activation	Hidden Layer Size	Learning Rate	Momentum	N Iter No Change	Solver	Tolerance	Mean Accuracy	Mean Fit Time (seconds)
logistic	(15,)	0.6	0.9	1000	adam	0.0001	98.5%	1.61
tanh	(5,)	0.4	0.7	100	sgd	0.0001	98.0%	0.51
tanh	(5,)	0.8	0.1	100	sgd	0.0001	98.0%	0.61
tanh	(5,)	1	0.3	1000	sgd	1	98.0%	0.77
logistic	(9,)	0.2	0.9	1000	adam	0.1	98.0%	0.78
tanh	(5,)	0.05	0.9	1000	adam	0.1	98.0%	0.80
tanh	(15,)	0.8	0.7	1000	sgd	0.01	98.0%	0.81
tanh	(5,)	1	0.7	1000	sgd	0.001	98.0%	0.86
tanh	(9,)	0.2	0.3	1000	adam	0.1	98.0%	0.92
tanh	(25,)	0.4	0.9	1000	sgd	1	98.0%	0.93
tanh	(5,)	0.05	0.9	1000	sgd	0.001	98.0%	0.93
logistic	(25,)	0.4	0.9	1000	adam	0.01	98.0%	0.95
tanh	(25,)	1	0.1	1000	sgd	0.001	98.0%	1.12
logistic	(9,)	0.4	0.3	1000	sgd	0.0001	98.0%	1.65

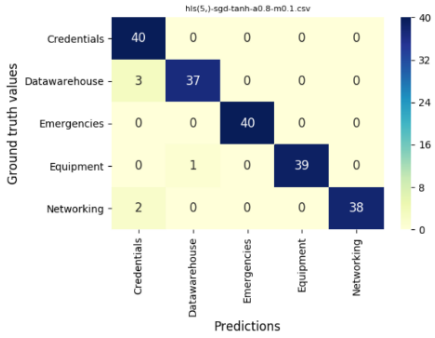
Appendix D Aggregate Confusion Matrices

Five-run aggregate confusion matrices of the 5 different combinations of hyperparameters.

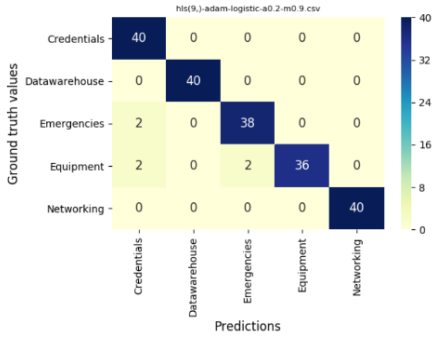
hls(15,)-adam-logistic-a0.6-m0.9
Accuracy: 97.6%

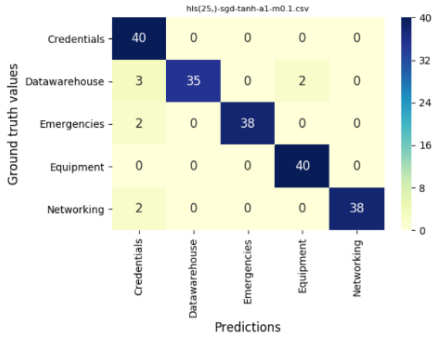


hls(5,)-sgd-tanh-a0.8-m0.1
Accuracy: 97.6%

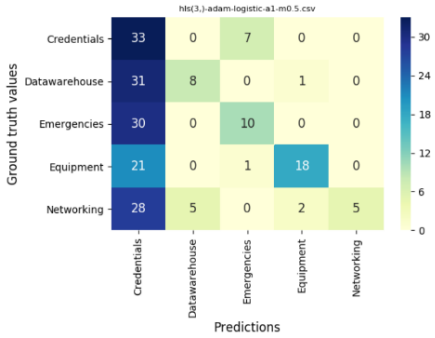


hls(9,)-adam-logistic-a0.2-m0.9
Accuracy: 97.6%





Accuracy: 96.4%
hls(25,)-sgd-tanh-a1-m0.1



Accuracy: 50%
hls(3,)-adam-logistic-a1-m0.5

Appendix E Ticketing-Routing Agent Console Output Example

File - tickets Intermediate

```
 1 /Users/[redacted]/Environments/CS5011-A4-NN-Ticketing-Routing
   -Agent/bin/python /Users/[redacted]/Projects/CS5011-A4-NN-
   Ticketing-Routing-Agent/src/A4Main.py --agent Int --csv
   tickets
 2
 3 ╔╦╗╦╔═╗╦╔═╔═╗╔╦╗╦  ╔═╗╔═╗╔═╗╔═╗╔═╗╦═╗
 4  ║ ╠╩╗╠╩╗║╣  ║ ║  ║ ║║ ╦║ ╦║╣ ╠╦╝
 5  ╩ ╩ ╩╩ ╩╚═╝ ╩ ╩═╝╚═╝╚═╝╚═╝╚═╝╩╚═
 6 You can quit at any time by typing 'exit' or 'quit'.
 7 Note: default answer is: 'Yes'.
 8
 9 New ticket
10 Please answer the following questions to log a new ticket:
11 Request [Yes/No]: y
12 Incident [Yes/No]: n
13 WebServices [Yes/No]: n
14 The system predicts that your ticket will be directed to
   Networking
15 Are you happy with this prediction [Yes/No]: n
16 Please continue answering the questions below to narrow
   down the choices:
17 Login [Yes/No]: y
18 Wireless [Yes/No]: n
19 The system predicts that your ticket will be directed to
   Credentials
20 Are you happy with this prediction [Yes/No]: y
21
22 Are you happy with the response team chosen [Yes/No]: y
23
24 Do you want to submit another ticket? [Yes/No]: y
25
26 New ticket
27 Please answer the following questions to log a new ticket:
28 Request [Yes/No]: n
29 Incident [Yes/No]: n
30 WebServices [Yes/No]: n
31 The system predicts that your ticket will be directed to
   Networking
32 Are you happy with this prediction [Yes/No]: n
33 Please continue answering the questions below to narrow
   down the choices:
34 Login [Yes/No]: n
35 Wireless [Yes/No]: n
36 The system predicts that your ticket will be directed to
   Networking
37 Are you happy with this prediction [Yes/No]: n
38 Please continue answering the questions below to narrow
   down the choices:
39 Printing [Yes/No]: n
40 IdCards [Yes/No]: n
```

Page 1 of 2

```
41 The system predicts that your ticket will be directed to
   Networking
42 Are you happy with this prediction [Yes/No]: n
43 Please continue answering the questions below to narrow
   down the choices:
44 Staff [Yes/No]: n
45 Students [Yes/No]: n
46
47 Your ticket is being directed to: Networking
48
49 Are you happy with the response team chosen [Yes/No]: n
50 Please choose a response team that you think suits your
   problem best from the following choices:
51 ['Credentials', 'Datawarehouse', 'Emergencies', 'Equipment
   ', 'Networking']: emergencies
52 Accuracy: 98.04%
53 Your change will be taken into consideration for future
   tickets.
54
55 Do you want to submit another ticket? [Yes/No]: y
56
57 New ticket
58 Please answer the following questions to log a new ticket:
59 Request [Yes/No]: n
60 Incident [Yes/No]: n
61 WebServices [Yes/No]: n
62 The system predicts that your ticket will be directed to
   Emergencies
63 Are you happy with this prediction [Yes/No]: y
64
65 Are you happy with the response team chosen [Yes/No]: y
66
67 Do you want to submit another ticket? [Yes/No]: n
68
69 Exiting Ticket Logger.
70
71 Process finished with exit code 0
72
```

Appendix F Intermediate Agent Prediction Testing

File - tickets Intermediate

```
 1 /Users/[redacted]/Environments/CS5011-A4-NN-Ticketing-Routing
   -Agent/bin/python /Users/[redacted]/Projects/CS5011-A4-NN-
   Ticketing-Routing-Agent/src/A4Main.py --agent Int --csv
   tickets
 2
 3 ╔╦╗╦╔═╗╦╔═╔═╗╔╦╗╦  ╔═╗╔═╗╔═╗╔═╗╦═╗
 4  ║ ╠╩╗╠╩╗║╣  ║ ║  ║ ╦║ ╦║ ╦║╣ ╠╦╝
 5  ╩ ╩ ╩╩ ╩╚═╝ ╩ ╩═╝╚═╝╚═╝╚═╝╚═╝╩╚═
 6 You can quit at any time by typing 'exit' or 'quit'.
 7 Note: default answer is: 'Yes'.
 8
 9 New ticket
10 Please answer the following questions to log a new ticket:
11 Request [Yes/No]: n
12 Incident [Yes/No]: y
13 WebServices [Yes/No]: y
14 The system predicts that your ticket will be directed to
   Datawarehouse
15 Are you happy with this prediction [Yes/No]: n
16 Please continue answering the questions below to narrow
   down the choices:
17 Login [Yes/No]: y
18 Wireless [Yes/No]: y
19 The system predicts that your ticket will be directed to
   Emergencies
20 Are you happy with this prediction [Yes/No]: y
21
22 Are you happy with the response team chosen [Yes/No]: y
23
24 Do you want to submit another ticket? [Yes/No]: y
25
26 New ticket
27 Please answer the following questions to log a new ticket:
28 Request [Yes/No]: n
29 Incident [Yes/No]: n
30 WebServices [Yes/No]: n
31 The system predicts that your ticket will be directed to
   Networking
32 Are you happy with this prediction [Yes/No]: n
33 Please continue answering the questions below to narrow
   down the choices:
34 Login [Yes/No]: n
35 Wireless [Yes/No]: y
36 The system predicts that your ticket will be directed to
   Networking
37 Are you happy with this prediction [Yes/No]: n
38 Please continue answering the questions below to narrow
   down the choices:
39 Printing [Yes/No]: y
40 IdCards [Yes/No]: n
```

Page 1 of 3

```
41 The system predicts that your ticket will be directed to
   Equipment
42 Are you happy with this prediction [Yes/No]: n
43 Please continue answering the questions below to narrow
   down the choices:
44 Staff [Yes/No]: n
45 Students [Yes/No]: y
46
47 Your ticket is being directed to: Credentials
48
49 Are you happy with the response team chosen [Yes/No]: y
50
51 Do you want to submit another ticket? [Yes/No]: y
52
53 New ticket
54 Please answer the following questions to log a new ticket:
55 Request [Yes/No]: y
56 Incident [Yes/No]: n
57 WebServices [Yes/No]: y
58 The system predicts that your ticket will be directed to
   Networking
59 Are you happy with this prediction [Yes/No]: y
60
61 Are you happy with the response team chosen [Yes/No]: y
62
63 Do you want to submit another ticket? [Yes/No]: y
64
65 New ticket
66 Please answer the following questions to log a new ticket:
67 Request [Yes/No]: n
68 Incident [Yes/No]: y
69 WebServices [Yes/No]: y
70 The system predicts that your ticket will be directed to
   Datawarehouse
71 Are you happy with this prediction [Yes/No]: y
72
73 Are you happy with the response team chosen [Yes/No]: y
74
75 Do you want to submit another ticket? [Yes/No]: y
76
77 New ticket
78 Please answer the following questions to log a new ticket:
79 Request [Yes/No]: n
80 Incident [Yes/No]: n
81 WebServices [Yes/No]: n
82 The system predicts that your ticket will be directed to
   Networking
83 Are you happy with this prediction [Yes/No]: n
84 Please continue answering the questions below to narrow
   down the choices:
```

```
 85 Login [Yes/No]: n
 86 Wireless [Yes/No]: nn
 87 The system predicts that your ticket will be directed to
    Networking
 88 Are you happy with this prediction [Yes/No]: n
 89 Please continue answering the questions below to narrow
    down the choices:
 90 Printing [Yes/No]: y
 91 IdCards [Yes/No]: n
 92 The system predicts that your ticket will be directed to
    Equipment
 93 Are you happy with this prediction [Yes/No]: y
 94
 95 Are you happy with the response team chosen [Yes/No]: y
 96
 97 Do you want to submit another ticket? [Yes/No]: n
 98
 99 Exiting Ticket Logger.
100
101 Process finished with exit code 0
102
```

Appendix G Intermediate Agent Text-Based Interface Input Testing

File - tickets Intermediate

```
 1 /Users/[redacted]/Environments/CS5011-A4-NN-Ticketing-Routing
   -Agent/bin/python /Users/[redacted]/Projects/CS5011-A4-NN-
   Ticketing-Routing-Agent/src/A4Main.py --agent Int --csv
   tickets
 2
 3 ╔╦╗╦╔═╗╦╔═╔═╗╔╦╗╦  ╔═╗╔═╗╔═╗╔═╗╔═╗╦═╗
 4  ║ ║╠═╣╠╩╗║╣  ║ ║  ║ ║║ ╦║ ╦║╣ ╠╦╝
 5  ╩ ╩╩ ╩╩ ╩╚═╝ ╩ ╩═╝╚═╝╚═╝╚═╝╚═╝╩╚═
 6 You can quit at any time by typing 'exit' or 'quit'.
 7 Note: default answer is: 'Yes'.
 8
 9 New ticket
10 Please answer the following questions to log a new ticket:
11 Request [Yes/No]: y
12 Incident [Yes/No]: NO
13 WebServices [Yes/No]: nn
14 The system predicts that your ticket will be directed to
   Networking
15 Are you happy with this prediction [Yes/No]: n
16 Please continue answering the questions below to narrow
   down the choices:
17 Login [Yes/No]: YES
18 Wireless [Yes/No]: YeS
19 The system predicts that your ticket will be directed to
   Credentials
20 Are you happy with this prediction [Yes/No]: No
21 Please continue answering the questions below to narrow
   down the choices:
22 Printing [Yes/No]: yy
23 IdCards [Yes/No]: n
24 The system predicts that your ticket will be directed to
   Credentials
25 Are you happy with this prediction [Yes/No]: n
26 Please continue answering the questions below to narrow
   down the choices:
27 Staff [Yes/No]: NO
28 Students [Yes/No]: YES
29
30 Your ticket is being directed to: Credentials
31
32 Are you happy with the response team chosen [Yes/No]: N
33 Please choose a response team that you think suits your
   problem best from the following choices:
34 ['Credentials', 'Datawarehouse', 'Emergencies', 'Equipment
   ', 'Networking']: wrong
35 The team you have chosen cannot be recognised.
36 Please choose a response team that you think suits your
   problem best from the following choices:
37 ['Credentials', 'Datawarehouse', 'Emergencies', 'Equipment
   ', 'Networking']: emergencieS
```

Page 1 of 2

```
38 Accuracy: 96.08%
39 Your change will be taken into consideration for future
   tickets.
40
41 Do you want to submit another ticket? [Yes/No]: y
42
43 New ticket
44 Please answer the following questions to log a new ticket:
45 Request [Yes/No]: n
46 Incident [Yes/No]: y
47 WebServices [Yes/No]: quit
48
49 Exiting Ticket Logger.
50
51 Process finished with exit code 0
52
```