

CS5011 A3

150006108

November 2018

1 Part1

The aim of part 1 was to design, build and train a neural net.

The first step was to properly encode the training data. The original file is a .csv file, with each column header representing a question, and the subsequent values within that column are either “Yes” or “No”, depending on how the aforementioned question was answered. The final column contains the country that results from that particular combination of question answers.

To successfully train the neural net, the data file must first be encoded into a usable format. The question data was simple. “Yes” and “No” can be encoded using 1 bit binary of 1 and 0 respectively. This was done by loading all the answer columns into a pandas dataframe, and using the *replace* function, replacing any “Yes” with 1, and any “No” with 0.

The location column could not be encoded in as simple a manner. However, scikit-learn provides an ample solution. Within the preprocessing package, there is a class called *OneHotEncoder*. By passing this a dataframe containing the “Dream Location” column, it successfully encodes all the locations with unique values.

With the data successfully encoded, it's time to build and train a neural network. Since we are only dealing with inputs and outputs of 0's and 1's, a sigmoid function is valid activation function. With a learning rate starting at 0.5, and a momentum of 0.3, we should avoid any local minimums. The only decision left was the size of the hidden layer.

To decide this, I built models using different values, and ran a *score* test to evaluate the models accuracy. The results can be seen below:

- 1 hidden node - $\approx 20\%$ accuracy
- 2 hidden nodes - $\approx 75\%$ accuracy
- 3 hidden nodes - $\approx 95\%$ accuracy
- 5 hidden nodes - 100% accuracy
- 10 hidden nodes - 100% accuracy

The score test was done on the entire training set, meaning that in the instance of 1 hidden node, the net was only getting 20% on data **it had already seen!**

In the end, I decided to use 5 hidden nodes, as the net was achieving consistently a 100% score on it's trained data with a minimal amount of nodes, meaning a minimal amount of computation.

Once the classifier is made and fitted, it is saved to a .joblib file with the same name as the name of the file containing the training data.

The system well generalised, meaning it will still function if other locations or features are added to files. This functionality will be seen in later parts.

2 Part 2

Part 2 allows a user to answer questions, and the system will take these answers and attempt the guess the users dream location using a neural net trained on the same file as the file being used in part 2.

At the start, the program will read in the .csv file given as an argument. It will then load the classifier related to the given file. It is assumed that a model already exists for the given file. A new encoder is also made, the same as in part 1 (unfortunately, you cannot use joblib to save encoders). This builds the same encoding map, and will later be used to decode the neural nets prediction.

Once all this is done, the system will start to ask the user questions. These are simply the question column header, but with a “?” appended to it. Originally, I had a switch statement, where each column header would output an actual question, like “Is your dream destination know for its rich history?”, however I found that when adding new features to the dataset, I wouldn’t be able to create full questions for these new features. With this in mind, I decided to go for a more generalised method, allowing all features within the file to be output in question form.

The system will only accept an answer in the form of either “Y” or “y” for yes, and “N” or “n” for no. These then get appended to a list as either 1 or 0 respective to the given answer.

After the system has asked half its questions, it will attempt to make a guess with the answers it has already gathered. This is done by substituting future answers with an answer of “No”.

If the guess is correct, the system simply stops. However, if the user says no, the system will start asking the rest of the questions. Once complete, the system will make another guess and ask the user to verify if it’s correct. This is where part 2 ends.

If an instance occurs where the neural net finds no match to the combination of answers entered, it will print “I have no guess”.

3 Part 3

The aim of part 3 was to expand on part 2, allowing users to help the system learn if it guesses incorrectly. They can do this by telling the system what the answer should have been, as well as letting it know a key feature that wasn’t asked about in the previous question set that would be a defining feature for this location.

Using these inputs, the system is able to build a new training data set, and train itself again, essentially adapting to and learning from this newly gained knowledge.

The learning process is quite simple. If the system guesses wrong, it asks the user to enter the correct answer. This is stored. It then asks the user if there is any new feature it should know that would help differentiate this new location from any other. If the user says yes, it then asks for this new feature. This is also stored.

The original training file is then read into a dataframe. If a new feature has been given, a new column is built containing all “No”’s, as it can be assumed that for all current locations in the database, this feature has never been key for their selection. This new column is then appended to the original dataframe just before the “Dream Location” column. If a new feature has not been given, this step can be skipped.

The new data entry is then built. It is essentially the answers given by the user with the user given location appended to the end. The answers must first be decoded back to “Yes” and “No” form, which is done by simply iterating through the list. If a new feature was given, a “Yes” is appended onto the list before the location, as this is where the new column will be located. This new row is then appended to the bottom of the dataframe.

Each new location and feature the system learns is saved in a file “additions.csv”. This file contains the new addition, whether or not it was a new location or new feature, and what file it was added to.

Once all this is complete, the original .csv file is overwritten with the new dataframe. The training method in part1 is then called again on the overwritten file, retraining the model with the new data. When part 3 is ran again, and the same answers are given, the neural net should now guess the new location.

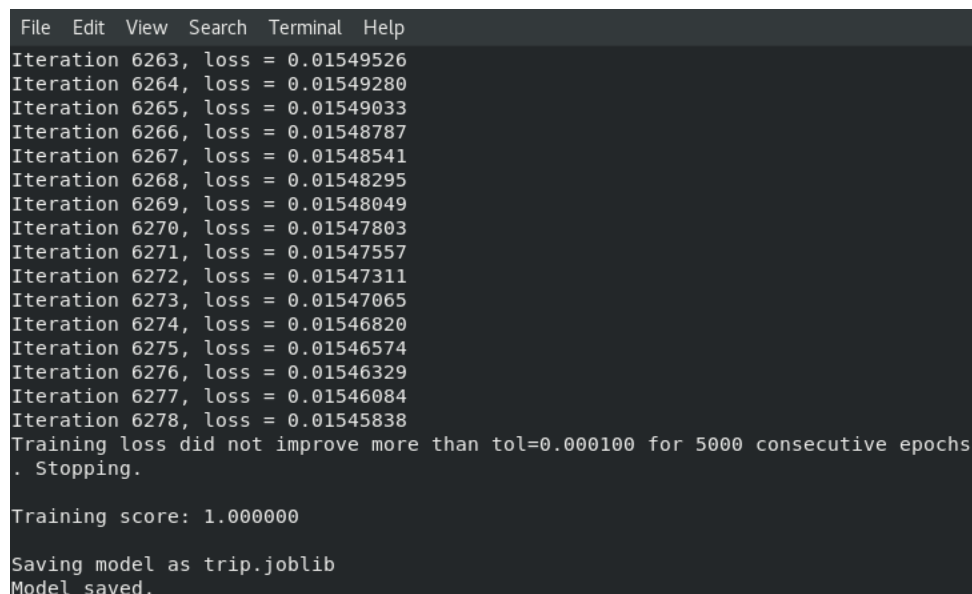
4 Testing

4.1 Part 1

4.1.1 Training model on trip.csv

Expected: The file is read, the dataframe is successfully encoded, and a model is successfully trained and tested. This model is then saved as “trip.joblib”.

Result: The model is successfully trained, scored and saved, suggesting all data was successfully encoded correctly also. Model saved in the above directory named “trip.joblib”.



```
File Edit View Search Terminal Help
Iteration 6263, loss = 0.01549526
Iteration 6264, loss = 0.01549280
Iteration 6265, loss = 0.01549033
Iteration 6266, loss = 0.01548787
Iteration 6267, loss = 0.01548541
Iteration 6268, loss = 0.01548295
Iteration 6269, loss = 0.01548049
Iteration 6270, loss = 0.01547803
Iteration 6271, loss = 0.01547557
Iteration 6272, loss = 0.01547311
Iteration 6273, loss = 0.01547065
Iteration 6274, loss = 0.01546820
Iteration 6275, loss = 0.01546574
Iteration 6276, loss = 0.01546329
Iteration 6277, loss = 0.01546084
Iteration 6278, loss = 0.01545838
Training loss did not improve more than tol=0.000100 for 5000 consecutive epochs
. Stopping.

Training score: 1.000000

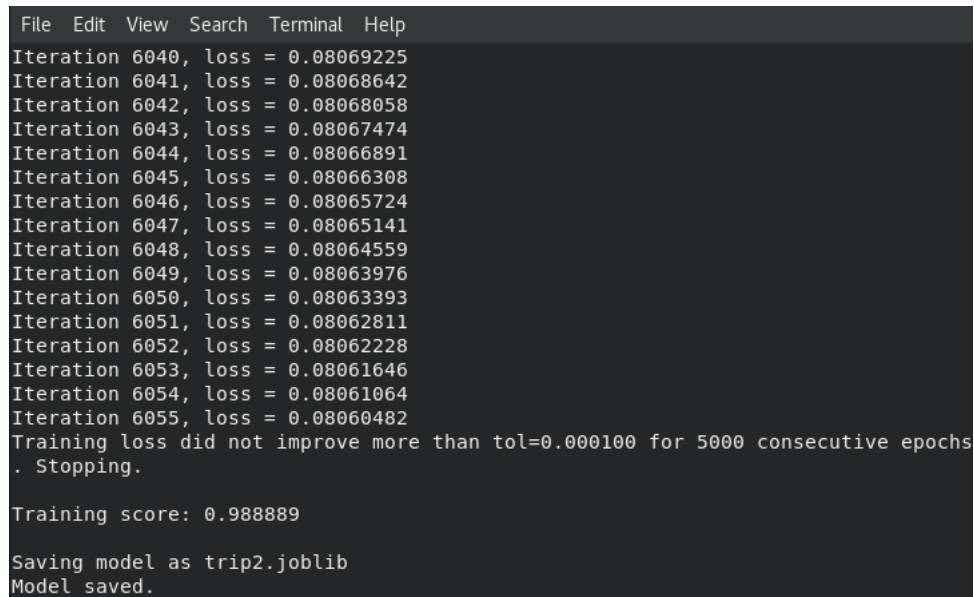
Saving model as trip.joblib
Model saved.
```

Figure 1: Output after running part 1 on trip.csv

4.1.2 Adding a new feature and location

Expected: No issues should be found. Using file “trip2.csv”, with the added column “In Europe” and the second row location changed from Greece to America, the data frame should be read and successfully encoded, and a model should be successfully trained, tested and saved as “trip2.joblib”.

Result: Like the first test, a model is trained, tested, and saved with no issues. The score takes a slight wavering, but this is most likely because a new location has been added without any distinctive feature, so there is a chance of guessing incorrectly.

A terminal window with a dark background and light text. At the top, there is a menu bar with 'File', 'Edit', 'View', 'Search', 'Terminal', and 'Help'. Below the menu, the terminal displays a series of lines showing training iterations from 6040 to 6055, each followed by a loss value. The loss values fluctuate slightly, generally staying between 0.0806 and 0.0807. After iteration 6055, a message states: 'Training loss did not improve more than tol=0.000100 for 5000 consecutive epochs . Stopping.' This is followed by 'Training score: 0.988889'. The final two lines are 'Saving model as trip2.joblib' and 'Model saved.'

```
File Edit View Search Terminal Help
Iteration 6040, loss = 0.08069225
Iteration 6041, loss = 0.08068642
Iteration 6042, loss = 0.08068058
Iteration 6043, loss = 0.08067474
Iteration 6044, loss = 0.08066891
Iteration 6045, loss = 0.08066308
Iteration 6046, loss = 0.08065724
Iteration 6047, loss = 0.08065141
Iteration 6048, loss = 0.08064559
Iteration 6049, loss = 0.08063976
Iteration 6050, loss = 0.08063393
Iteration 6051, loss = 0.08062811
Iteration 6052, loss = 0.08062228
Iteration 6053, loss = 0.08061646
Iteration 6054, loss = 0.08061064
Iteration 6055, loss = 0.08060482
Training loss did not improve more than tol=0.000100 for 5000 consecutive epochs
. Stopping.

Training score: 0.988889

Saving model as trip2.joblib
Model saved.
```

Figure 2: Output after running part 1 with trip2.csv

4.2 Part 2

4.2.1 Handling “no guess” case

Expected: When encountering a case where no prediction can be made, the system will print out a plain message, rather than crashing.

Result: At both the mid point guess and the final guess, the net is unable to come up with a guess when given all “No”’s as answers. However, the system handles the case both times, reaching the end of its runtime without any issues.

```
pc2-041-l:~/Documents/CS5011/A3/CS5011-A3/part2 ljw26$ python3 part2.py trip.csv
Short Stay?
Please answer Y for yes, N for no: n
Penguins?
Please answer Y for yes, N for no: n
Longest rivers?
Please answer Y for yes, N for no: n
Island?
Please answer Y for yes, N for no: n

I have no guess.
Let's continue!

Seaside?
Please answer Y for yes, N for no: █
```

(a) Output of mid-point guess

```
pc2-041-l:~/Documents/CS5011/A3/CS5011-A3/part2 ljw26$ python3 part2.py trip.csv
Short Stay?
Please answer Y for yes, N for no: n
Penguins?
Please answer Y for yes, N for no: n
Longest rivers?
Please answer Y for yes, N for no: n
Island?
Please answer Y for yes, N for no: n

I have no guess.
Let's continue!

Seaside?
Please answer Y for yes, N for no: n
Historical?
Please answer Y for yes, N for no: n
Speaking Spanish?
Please answer Y for yes, N for no: n
Food?
Please answer Y for yes, N for no: n

I have no guess.
pc2-041-l:~/Documents/CS5011/A3/CS5011-A3/part2 ljw26$ █
```

(b) Final output

Figure 3: Outputs from running part 2 with trip.csv

4.2.2 Testing known data

Expected: When inputting data already trained on, the system should guess correctly.

Result: When inputting the first answer set from trip.csv, it gives the correct answer.

```
Short Stay?
Please answer Y for yes, N for no: n
Penguins?
Please answer Y for yes, N for no: n
Longest rivers?
Please answer Y for yes, N for no: n
Island?
Please answer Y for yes, N for no: n

I have no guess.
Let's continue!

Seaside?
Please answer Y for yes, N for no: n
Historical?
Please answer Y for yes, N for no: y
Speaking Spanish?
Please answer Y for yes, N for no: y
Food?
Please answer Y for yes, N for no: y
I think your dream destination is Spain
Am I correct? y
Hooray!
```

Figure 4: Output for correctly guessing location.

4.2.3 Testing with random entry data

Expected: The system should give a guess at both the mid-point and the final guess.

Result: The system originally guesses Australia, the user then says incorrect, so it continues asking questions, and gives a final guess of Argentina. This shows both times guessing functionality is working as expected.

```

Short Stay?
Please answer Y for yes, N for no: n
Penguins?
Please answer Y for yes, N for no: y
Longest rivers?
Please answer Y for yes, N for no: y
Island?
Please answer Y for yes, N for no: n
I think your dream destination is Australia
Am I correct? n
Oh dear. Let's continue...
Seaside?
Please answer Y for yes, N for no: y
Historical?
Please answer Y for yes, N for no: y
Speaking Spanish?
Please answer Y for yes, N for no: y
Food?
Please answer Y for yes, N for no: y
I think your dream destination is Argentina
Am I correct? █

```

Figure 5: Output of a random data entry

4.3 Part 3

4.3.1 Testing addition of new location and feature and retraining

Expected: When told it's incorrect, system asks for new locations and features. When these are input, data is altered and model is retrained, with additions.csv being updated. When program is ran again, the new location is given as the guess.

Result: All tests pass. New data is asked for. When data is entered, new dataframe is built and original file is overwritten. Model is then retrained using new file, with additions now containing the new location and feature. When program is ran again, new question is asked, and new location is given as answer.

```

Please answer Y for yes, N for no: y
Penguins?
Please answer Y for yes, N for no: n
Longest rivers?
Please answer Y for yes, N for no: n
Island?
Please answer Y for yes, N for no: y
I think your dream destination is Greece
Am I correct? n
Oh dear. Let's continue...
Seaside?
Please answer Y for yes, N for no: y
Historical?
Please answer Y for yes, N for no: y
Speaking Spanish?
Please answer Y for yes, N for no: n
Food?
Please answer Y for yes, N for no: y
I think your dream destination is Greece
Am I correct? n
Please tell me your dream destination: America
Your dream destination is America correct? (y/n): y
Is there a defining feature to this destination that we have not asked about? y
Please tell me this new feature! Nature Parks █

```

Figure 6: Entering new location and feature

```

Iteration 6559, loss = 0.02169639
Iteration 6560, loss = 0.02169219
Iteration 6561, loss = 0.02168799
Iteration 6562, loss = 0.02168379
Iteration 6563, loss = 0.02167960
Iteration 6564, loss = 0.02167540
Iteration 6565, loss = 0.02167121
Iteration 6566, loss = 0.02166702
Iteration 6567, loss = 0.02166283
Iteration 6568, loss = 0.02165864
Iteration 6569, loss = 0.02165446
Iteration 6570, loss = 0.02165027
Iteration 6571, loss = 0.02164609
Iteration 6572, loss = 0.02164191
Training loss did not improve more than tol=0.000100 for 5000 consecutive epochs
. Stopping.

Training score: 1.000000

Saving model as trip.joblib
Model saved.

Classifier retrained...
pc2-041-l:~/Documents/CS5011/A3/CS5011-A3/part3 ljw26$

```

Figure 7: Model retrained using new data

```

Short Stay,Penguins,Longest rivers,Island,Seaside,Historical,Speaking Spanish,Food,Nature Parks,Dream Location
No,No,No,No,No,Yes,Yes,Yes,No,Spain
Yes,No,No,Yes,Yes,Yes,No,Yes,No,Greece

```

Figure 8: New feature has been added as a new column in data file (see “Nature Parks” before “Dream Location”)

```

90 No,No,Yes,No,No,Yes,No,Yes,No,Egypt
91 No,No,Yes,Yes,No,Yes,No,Yes,No,Egypt
92 Yes,No,No,Yes,Yes,Yes,No,Yes,Yes,America
93

```

Figure 9: New row been added for users answer set and new location

```

Short Stay?
Please answer Y for yes, N for no: y
Penguins?
Please answer Y for yes, N for no: n
Longest rivers?
Please answer Y for yes, N for no: n
Island?
Please answer Y for yes, N for no: y
Seaside?
Please answer Y for yes, N for no: y

I have no guess.
Let's continue!

Historical?
Please answer Y for yes, N for no: y
Speaking Spanish?
Please answer Y for yes, N for no: n
Food?
Please answer Y for yes, N for no: y
Nature Parks?
Please answer Y for yes, N for no: y
I think your dream destination is America
Am I correct?

```

Figure 10: When entering same answer set, new location now guessed

```

Name,Type,File
America,Location,../trip.csv
Nature Parks,Feature,../trip.csv

```

Figure 11: New contents of additions.csv, which was originally empty.

5 Evaluation

I believe I have fulfilled all requirements of the specification. In part one, I have implemented a program which can read in a data file, encode the given data file into a usable format, and then build, train, test and save a Multi-Layered Perceptron for later use. The solution is generalised to allow for any number of question columns or any amount of given locations.

In part 2, I implemented a program which can take the users answers, encode them into a usable format, and use the previously saved neural net classifier to guess what the users dream destination is. It is able to attempt a guess halfway through, and if incorrect, it will continue until all questions are asked, where it will guess again with the new answers given.

In part 3, I extended part 2 to allow the system to learn, providing the user a chance to tell the system what the correct answer was, hereby allowing the system to expand its current knowledge base. The model for the chosen file is then retrained, and the system will now give the correct answer if the newly learned input pattern is given again. A new file is also used to track newly added locations and features, as well as to what file they were added to, allowing any system administrators to see what alterations have been made to any models since launching them.

6 Running

To run any of the parts, simply navigate into their respective directories and run the command:

```
python3 part[n].py [csv file to use]
```

Make sure you have the latest installations of pandas and scikit-learn installed in your python interpreter.

7 Literature Review

The article I read discusses multiple uses of machine learning used in every day life. These include route optimisation for commuters, commercial flight autopilots, spam filters etc... All have machine learning algorithms, including neural networks, running in the background, aiding the computers in making decisions that can have large effects on each of our lives.

8 Bibliography

<https://www.techemergence.com/everyday-examples-of-ai/>

https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html#examples-using-sklearn-neural-network-mlpclassifier

https://scikit-learn.org/stable/modules/model_persistence.html