Report Foundation of Data Analysis

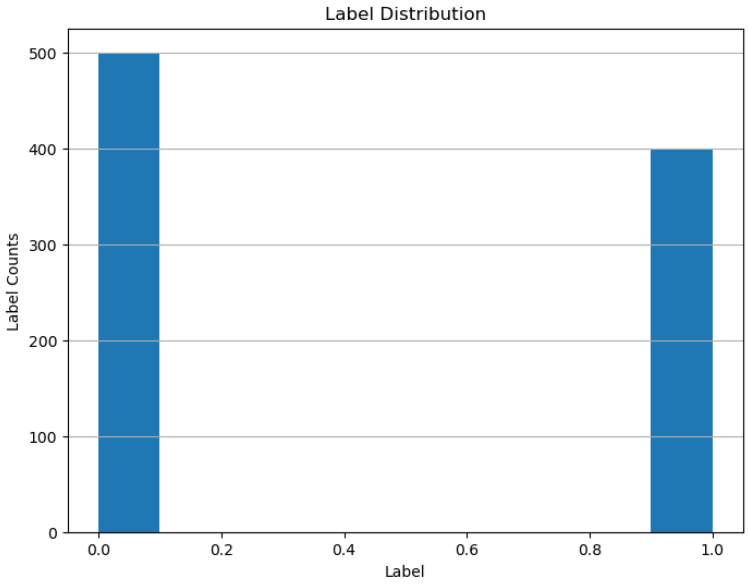
Part I

1. Understanding the Data

a) There are 900 samples and 2 features.

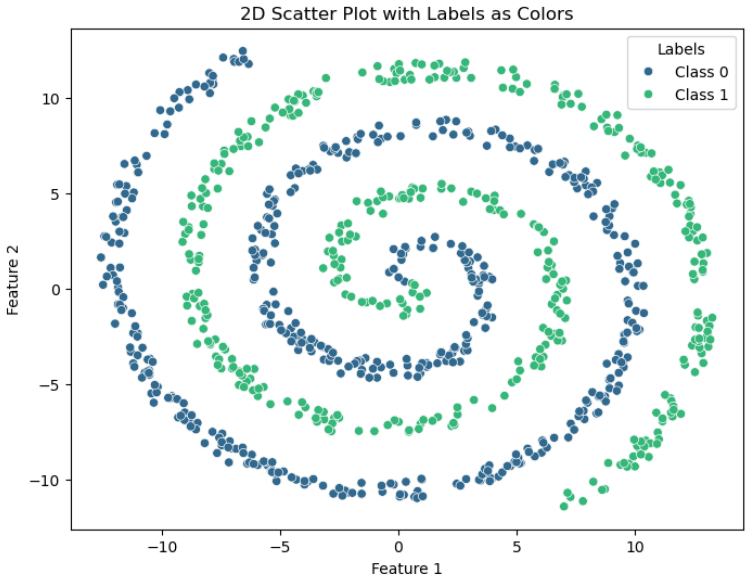
b) binary classification, because there are just two unique labels (0 and 1)

c)



2. Data visualisation

a)



b) No, it's not. The data is not linearly separable at first glance, because plot of the data is obviously spiral-shaped and it's not possible here to separate it by a straight line.

c) No, because again the shape of the data is spiral-shaped, since linear model tries always to separate data by a straight line, in our case here it's not possible. So, for this reason it's not possible here to achieve zero empirical risk.

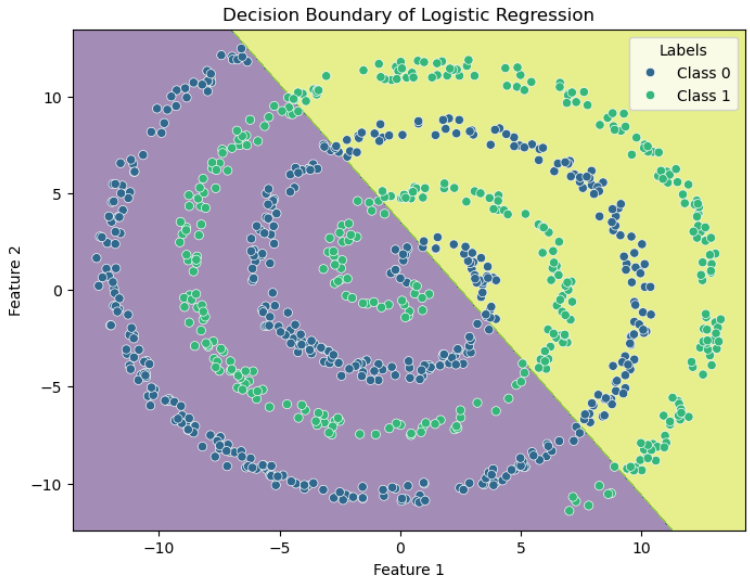
3. Fitting the first classifier

a) The split is necessary, because we want to make sure the model can handle the new data, and not just the data it was already trained with.

b) I've chosen **Logistic Regression,** as it's the best algorithm used for binary classifications. So, exactly what we need for our example data.

c) “Logistic Regression model accuracy: 64.44%”.
In our case, where we’ve got 900 samples and as we can see from histogram, we have 500 samples for class 0 and 400 samples for class 1. This means for us, that always predicting class 0 gives us 55,56%. Since our Logistic Regression is 64.44%, it’s better than the baseline, but it’s still not perfect.

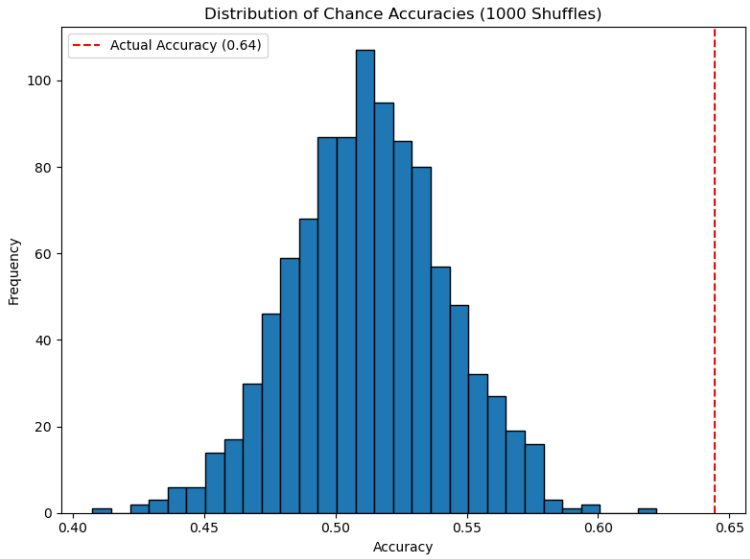
d)



4. Does the model perform better than chance?

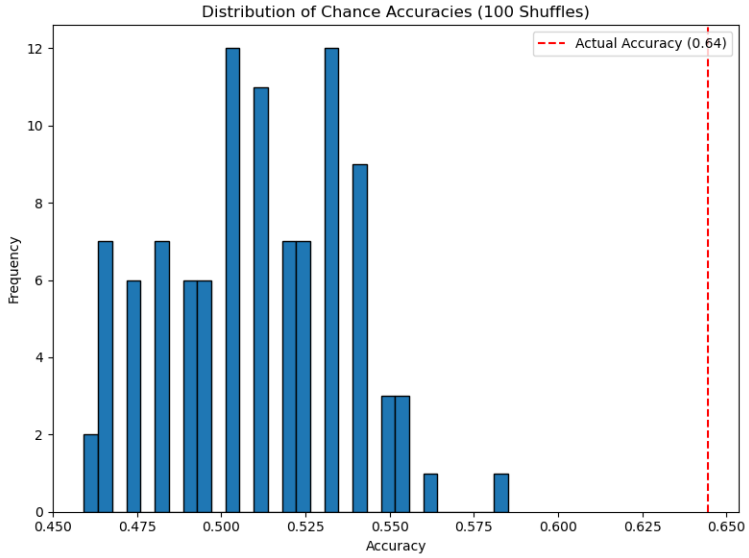
a) Accuracy after shuffling the data one time: 51.85%

b) As we see from histogram, the most chance accuracy lies within 51-54% and an actual accuracy is 64.44%, which is obviously higher than any chance accuracy.



c) This plot suggests us, that our model’s actual accuracy is much better than what we’ve got after 1000 shuffles. So, none of these random guesses could exceed or at least reach the actual accuracy, which means, our model’s accuracy hasn’t occurred by chance.

d) As was suggested at the task description, I repeated the shuffling 1000 times, because it provides more stable estimation of the chance distribution, and a fewer number of shuffling provides less trusted output. For comparison, see histogram below, where we obviously can see many free spaces between each bar chart.



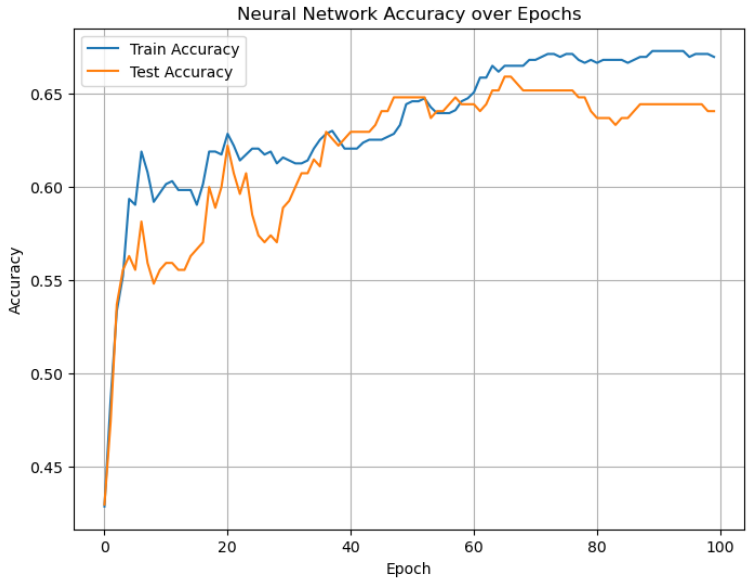
e) The most frequent class accuracy: 58.15%
No, it's not the same as chance accuracy, because baseline always picks the majority class, and chance accuracy is based on random guesses. So, baseline (dummy classifier) compares the performance of more sophisticated models.

f) After testing chance accuracy with both 100 and 1000 shuffles, and comparing it to a baseline classifier, we clearly can see, that none of them came close to the model's actual accuracy of 64.44%. It means for us, the performance of our model isn't that perfect, as we could wish, but still it learned something meaningful from the data. It's clearly better than random guessing or dummy classifier.

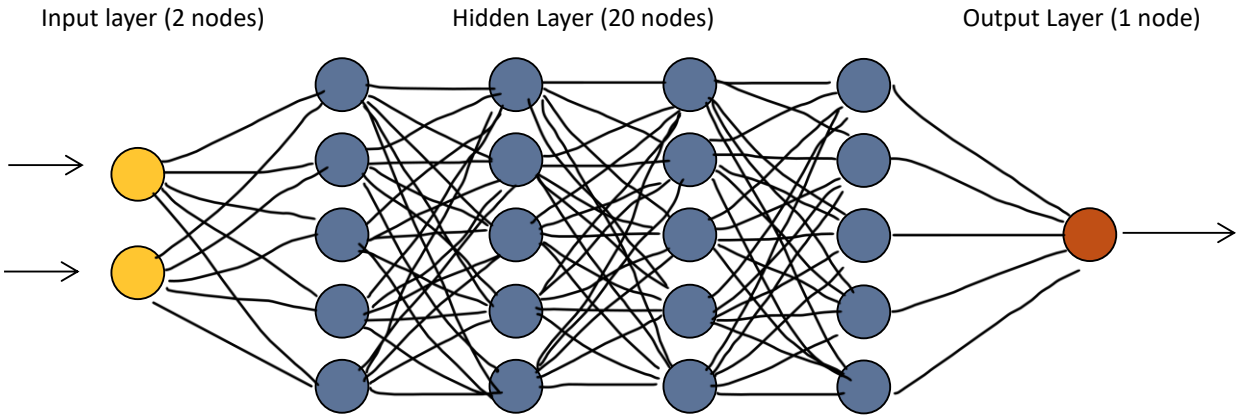
5. Neural nets

a) Created a neural network with 2 input layers, 20 hidden layers + ReLU, 1 output layer, "BCEWithLogitsLoss" loss, Adam optimizer with 0.01 learning rate, trained over 100 epochs.

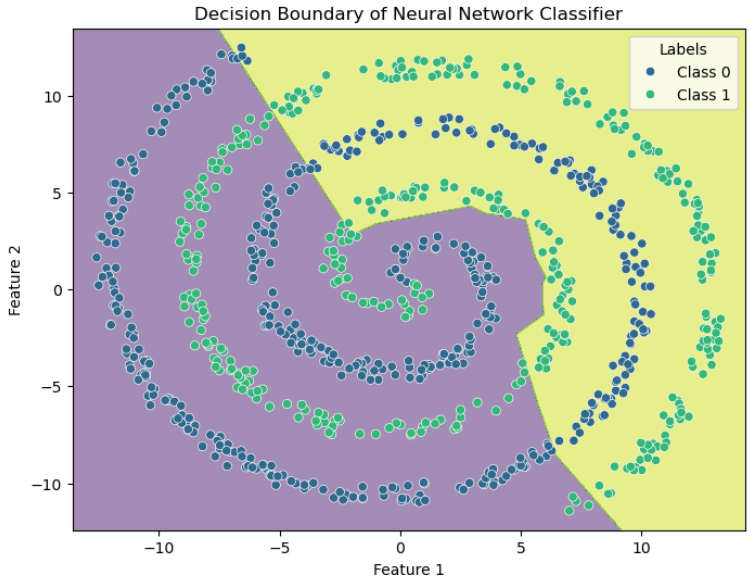
b) From the training curve, we can deduce, that training accuracy improves and reaches around 67%. Test accuracy has nearly the same curve but stabilizes at around 64%. As expected, the model is learning non-linear structure. For now, it seems the model is slightly underfitting, due to small gap between train and test accuracies.



c) I used ReLU activation in the hidden layer, applied BCEWithLogitsLoss and Adam optimizer with 0.01 learning rate. It seemed to me simple and efficient. I preferred BCEWithlogitsLoss over BCELoss, so I don't have to deal with sigmoid, and it's also very common for binary classification. I've also chosen 20 hidden units, which gives the network enough capacity without overfitting.



d)



Part II

1. Download the MUTAG dataset

Promt: "How to download MUTAG dataset from TUDatasets using PyTorch geometrics?"

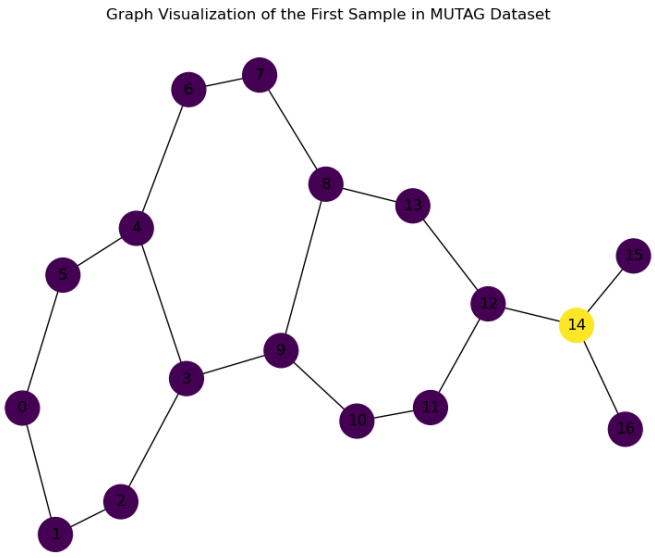
a) 188 samples (graphs), 7 features, 2 classes

b) The MUTAG dataset represents a collection of chemical compounds divided into two classes according to their mutagenic effect on a bacterium.

- Nodes – represent atoms labelled by their type (e.g., carbon, nitrogen, oxygen, fluorine, …)
- Edges – represent chemical bonds between atoms (aromatic, single, double, triple)
- Features – encode atomic properties
- Labels – provides information whether molecule is mutagenic (i.e. can cause mutation in DNA)

c) Number of nodes: 17, Number of edges: 38

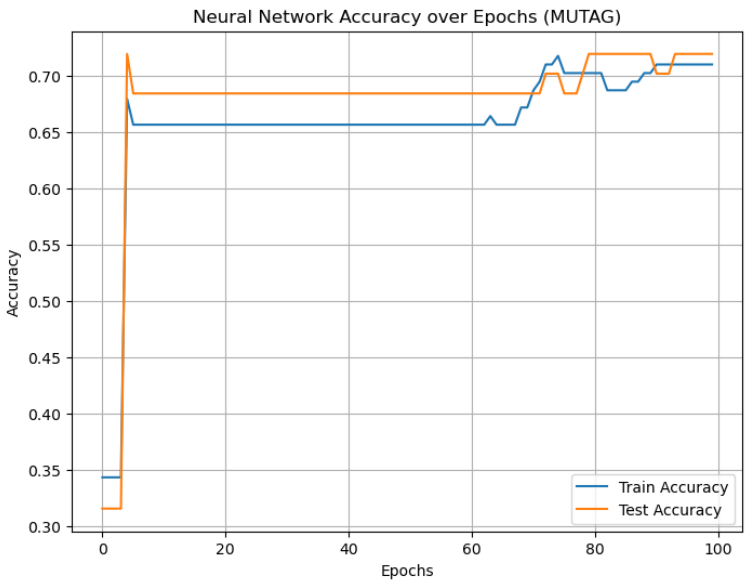
d) Promt: How can I plot the first graph? (use suitable package and colour the nodes according to the second feature)



2. Train and evaluate as in Part I

a) Mostly, I just took my previous code from part I and just adapted all variables with "_MUTAG" ending. For feedforward neural network I also referred to my previous code in part I from "NeuralNetwork" class and also to this website. For data preparation, I just asked copilot to prepare MUTAG data and provided my code, so it could adapt the data based on this code. (Promt: Make a load and data preparation for my code below: `Code snippet`).

b)



5. Summarise

a) First, I read all questions asked in the tasks and then asked an AI as summarizing prompt, and also not like a question, but rather as "command". For example, if question says, **"How many samples, features and classes are in the dataset?",** I ask in the prompt **"Give me code for identifying numbers of samples, features and classes".**

b) I always was checking the respond from AI in google for truthfulness. All my sources documented in the "Source" section below. Everything regarding part II begins with source 27 and below.

c) For me it was mostly helpful and I can tell only positive about using AI chatbot. Everywhere, when I needed it, it gave me a clear and clean answer.

Source:

1. numpy.histogram — NumPy v2.2 Manual
2. pandas.Series.map — pandas 2.2.3 documentation
3. numpy.load — NumPy v2.2 Manual
4. Histograms in Python
5. Data Analysis with Python: Zero to Pandas | Jovian
6. seaborn.scatterplot — seaborn 0.13.2 documentation
7. Choosing color palettes — seaborn 0.13.2 documentation
8. matplotlib.pyplot.legend — Matplotlib 3.10.3 documentation
9. Logistic Regression in Machine Learning | GeeksforGeeks
10. Plot Decision Boundary in Logistic Regression: Python Example
11. How to Shuffle a Pandas DataFrame? | by Hey Amit | Medium
12. python - What does numpy.random.seed(0) do? - Stack Overflow
13. python - Randomly shuffle items in each row of numpy array - Stack Overflow
14. A Dummy Classifier, A Baseline Classifier, or a Null Model | by Sahel Eskandar | Medium
15. (1) How to Build a Neural Network Classifier in PyTorch? (Technical Deep Dive - Part 1) - YouTube
16. How to Build & Train a Neural Network for Classification in PyTorch? (Part 2)
17. Neural Networks — PyTorch Tutorials 2.7.0+cu126 documentation
18. 02. PyTorch Neural Network Classification - Zero to Mastery Learn PyTorch for Deep Learning
19. Pytorch Neural Network Binary Classification
20. Build a Graph Neural Network Classifier in PyTorch | by Rjnclarke | Medium
21. Use PyTorch to train your image classification model | Microsoft Learn
22. PyTorch-Tutorial (The Classification)
23. Building a Binary Classification Model in PyTorch - MachineLearningMastery.com
24. A Simple Neural Network Classifier using PyTorch, from Scratch | by Jeril Kuriakose | Analytics Vidhya | Medium
25. PyTorch Tutorial | GeeksforGeeks
26. python - Plot the Decision Boundary of a Neural Network in PyTorch - Stack Overflow
27. MUTAG Dataset | Papers With Code
28. torch_geometric.datasets.TUDataset — pytorch_geometric documentation
29. torch_geometric.data.Dataset — pytorch_geometric documentation
30. PyTorch Geometric Tutorial. “Data is the new oil,” they say — but… | by Hey Amit | We Talk Data | Medium
31. Feedforward Neural Networks (FNN) - Deep Learning Wizard