Assignment 1:

Artificial Neural Networks

Nicole Cristine Dressler
University of Sunderland
Machine Learning and Data Mining (CETM26)

Table of Content

1. Introduction

The field of Machine Learning has grown significantly over the past years, with a special attention to deep learning and neural networks. Inspired by the brain's biological neural networks, the Artificial Neural Networks (ANN) have come to help in many human processes. This report aims to implement a feed-forward neural network for a binary classification, providing a practical understanding of the construction, training and evaluation using Python, Keras and Jupyter Notebooks.

This project focuses mainly on the network architecture and all of its aspects such as its parameters and hyperparameters to provide an optimal solution for the best architecture. It will present a comparison between different ANN architectures, showing data pre-processing steps, changes in the number of hidden layers, the number of nodes on each and so on to provide a comparison and explanation to the best solution/architecture.

In summary, this report aims to demonstrate a practical application of artificial neural networks in the marketing sector of a financial institution - a bank for this project -, showcasing the process of constructing, training, and evaluating a neural network.

2. Methods

The code for this project (Appendix A) was developed in a structured approach. The methodology involved will be elucidated below.

2.1 Programming Language and Framework

The programming language chosen was Python, due to its versatility and extensive libraries, it has a great performance for data science. Here are the libraries and versions used:

Python version: 3.8.12
Numpy version: 1.22.4
Pandas version: 2.0.0
SKLearn version: 1.2.2
Tensorflow version: 2.5.0
Matplotlib version: 3.4.3
Seaborn version: 0.11.2

Keras version: 2.5.0

2.2 Code Structure and Implementation

The code implemented used the Sequential API in Keras. The holdout method was used for the data splitting in a 70% ratio for the training set and 30% for the test set. For feature engineering techniques, scaling the data was employed to ensure compatibility and value normalisation. The code was developed on a Jupyter Notebook environment, making use of its ease for an interactive development and data visualisation possibilities.

The code has an organisational separation:

- Import libraries: get the code requirements and check the versions.
- Data: importing and exploring the data for the project.
- Initial pre-processing: data cleaning and formatting, and dividing the data between features and target.
- Function: function written for the creation of the different models to be evaluated and storing the results.
- Scaling: scaling of the data, comparing standardisation and normalisation with an evaluation of which method provides a better performance for our data and goal.
- Models: creation and evaluation of models (using the function created above) with different architectures by changing the number of hidden layers and nodes.
- Optimising the activation: after testing various numbers of hidden layers and nodes, this section tests and evaluates different activation methods for the hidden layers.

Certain parts of the code and the experiment are needed for the models to be complete and evaluated, as the optimizer (‘SGD’) and the number of epochs (50). Additionally, models 1 to 9 and model SLP used the standardised data as the chosen scaling method.

On another note, it is relevant to explain that the code for the single layer perceptron model created a simple single-layer model (not a perceptron in the strict sense) - Model SLP. This code is considered a basic implementation of a single-layer model in Keras, being equivalent to a perceptron.

2.3 Data and Ethical Considerations

The data provided for the project is about a marketing campaign for a Portuguese bank. It originated from the Bank Marketing Dataset available at the UCI Machine Learning Repository (Moro et al., 2014). Nor do the code or data involve sensitive information or pose ethical concerns; therefore, no specific ethical considerations were necessary for this project.

2.4 Control and Reproducibility

A remote repository was created on GitHub using Git as backup and made available as a public repository:
https://github.com/ndressler/Masters_U.Sunderland/blob/main/CETM26_Assignment1.ipynb

3. Results

Multiple models with different architectures were made to provide a comparison between the different constructions of the architectures by first evaluating the scaling methods (Fig. 1), then changing the number of hidden layers and the number of nodes in them (Fig. 2 and 3), and finally the activation function (Fig. 4 and 5).

Figure 1. Results of comparison of scaling models

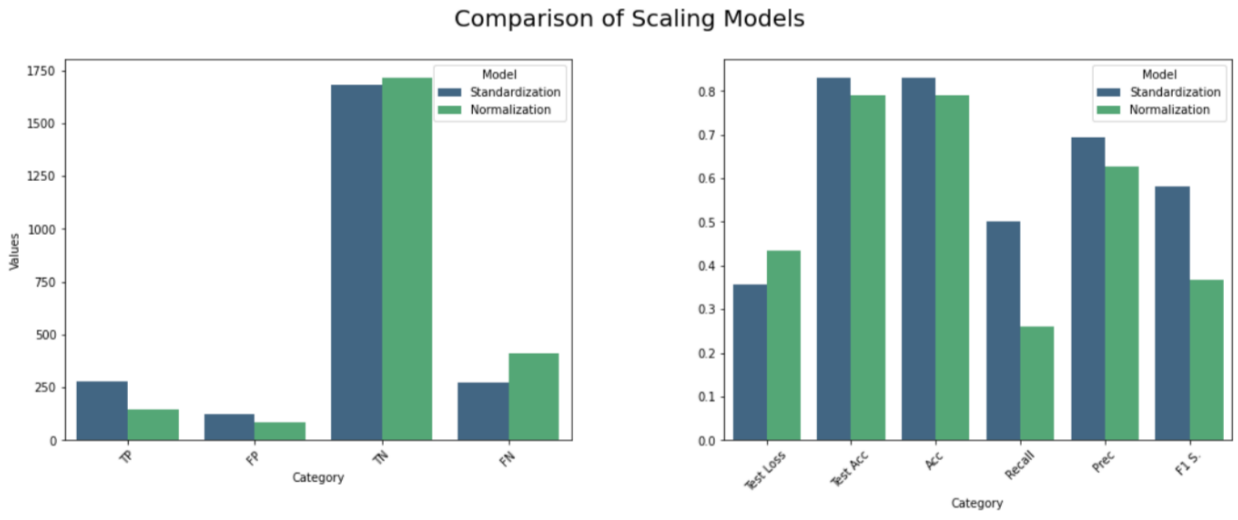


Figure 2. Table of comparison for confusion matrix results

Category	TP	FP	TN	FN	Score
Model 9 - 3 hl, 64 n	356	173	1629	195	867.0
Model 6 - 3 hl, 32 n	352	169	1633	199	855.0
Model 5 - 2 hl, 32 n	317	148	1654	234	750.0
Model 3 - 3 hl, 16 n	313	173	1629	238	738.0
Model 8 - 2 hl, 64 n	313	134	1668	238	738.0
Model 2 - 2 hl, 16 n	292	137	1665	259	675.0
Model 7 - 1 hl, 64 n	279	120	1682	272	636.0
Standardization	276	122	1680	275	627.0
Model 4 - 1 hl, 32 n	273	114	1688	278	618.0
Model 1 - 1 hl, 16 n	261	127	1675	290	582.0
Single Layer Perceptron	180	89	1713	371	339.0
Normalization	143	85	1717	408	228.0

Figure 3. Table of comparison for evaluation metrics results

Category	Test Loss	Test Acc	Acc	Recall	Prec	F1 S.
Model 9 - 3 hl, 64 n	0.340897	0.843604	0.843604	0.646098	0.672968	0.659259
Model 6 - 3 hl, 32 n	0.343799	0.843604	0.843604	0.638838	0.675624	0.656716
Model 5 - 2 hl, 32 n	0.346957	0.837654	0.837654	0.575318	0.681720	0.624016
Model 8 - 2 hl, 64 n	0.340707	0.841904	0.841904	0.568058	0.700224	0.627255
Model 3 - 3 hl, 16 n	0.359367	0.825329	0.825329	0.568058	0.644033	0.603664
Model 2 - 2 hl, 16 n	0.358285	0.831704	0.831704	0.529946	0.680653	0.595918
Model 7 - 1 hl, 64 n	0.348295	0.833404	0.833404	0.506352	0.699248	0.587368
Standardization	0.355440	0.831279	0.831279	0.500907	0.693467	0.581665
Model 4 - 1 hl, 32 n	0.356436	0.833404	0.833404	0.495463	0.705426	0.582090
Model 1 - 1 hl, 16 n	0.368405	0.822779	0.822779	0.473684	0.672680	0.555911
Single Layer Perceptron	0.412823	0.804505	0.804505	0.326679	0.669145	0.439024
Normalization	0.433057	0.790480	0.790480	0.259528	0.627193	0.367137

Figure 4. Table of activation comparison for confusion matrix results

Category	TP	FP	TN	FN	Score
Model 9 relu	356	173	1629	195	867.0
Model 6 relu	352	169	1633	199	855.0
Model 5 relu	317	148	1654	234	750.0
Model 9 tanh	297	119	1683	254	690.0
Model 6 tahn	285	134	1668	266	654.0
Model 5 tanh	248	138	1664	303	543.0

Figure 5. Table of activation comparison for evaluation metrics results

Category	Test Loss	Test Acc	Acc	Recall	Prec	F1 S.
Model 9 relu	0.340897	0.843604	0.843604	0.646098	0.672968	0.659259
Model 6 relu	0.343799	0.843604	0.843604	0.638838	0.675624	0.656716
Model 5 relu	0.346957	0.837654	0.837654	0.575318	0.681720	0.624016
Model 9 tanh	0.343860	0.841479	0.841479	0.539020	0.713942	0.614271
Model 6 tahn	0.350287	0.830004	0.830004	0.517241	0.680191	0.587629
Model 5 tanh	0.382021	0.812580	0.812580	0.450091	0.642487	0.529349

4. Evaluation

To recapitulate, the project's goal was to solve a binary classification task by constructing, training and evaluating an ANN, testing and comparing different types of architectures for the problem at hand to then define which architecture is optimal.

First, to discuss the evaluative methodology of the task, the data is about a marketing campaign for a bank, the objective is to predict if the clients will subscribe to the term deposit. Therefore, the true-positive results tell us about a client who will make a term in reality and who is also predicted to make a term, we want this number to be as high as possible, so that the marketing team can focus on the right clients to convert; the true-negative results are important to not waste marketing resources, as are the false-positive, we want both values to be as low as possible; most importantly are the false-negative results, they will make the marketing team lose

clients as they would actually sign up for the term deposit but were predicted as negatives, resulting in the marketing team not reaching out and not making the sale.

Thus, the metrics with higher importance are obtaining the lowest false-negative results followed by highest true-positives and the recall followed by F1 score metrics, since a high recall minimises the false-negatives and F1 score measures both recall and precision.

Having the evaluation methodology clarified, we can access the first evaluation still on the pre-processing step, which is the scaling of the data. The standardisation method was superior from the normalisation method for obtaining lower false-negatives and higher recall (Fig. 1).

After the scaling method is selected we move to the architectural choice of number of hidden layers and their nodes for an optimal network model is tested - range from one to three hidden layers with either 16, 32 or 64 nodes on each, plus a perceptron model (Fig. 6, 7 and 8).

Figure 6. Comparison of confusion matrix results for all models

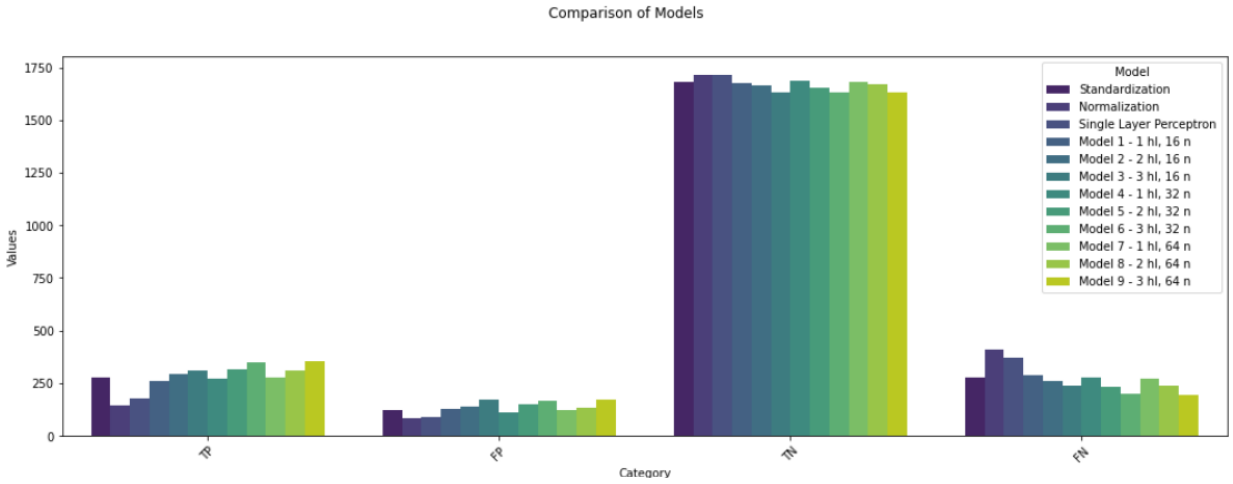


Figure 7. Comparison of evaluation metrics results for all models

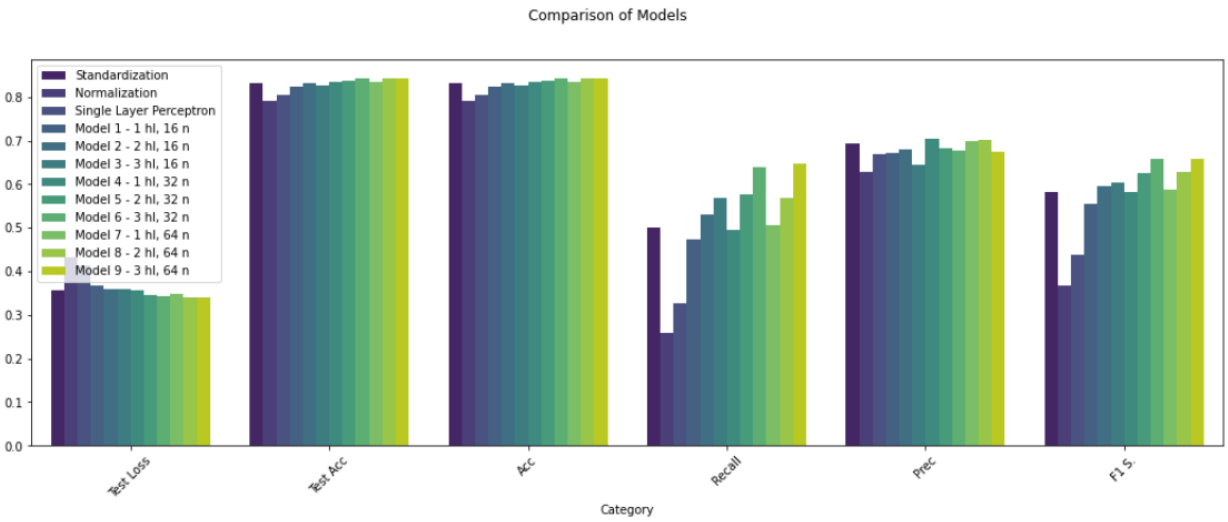
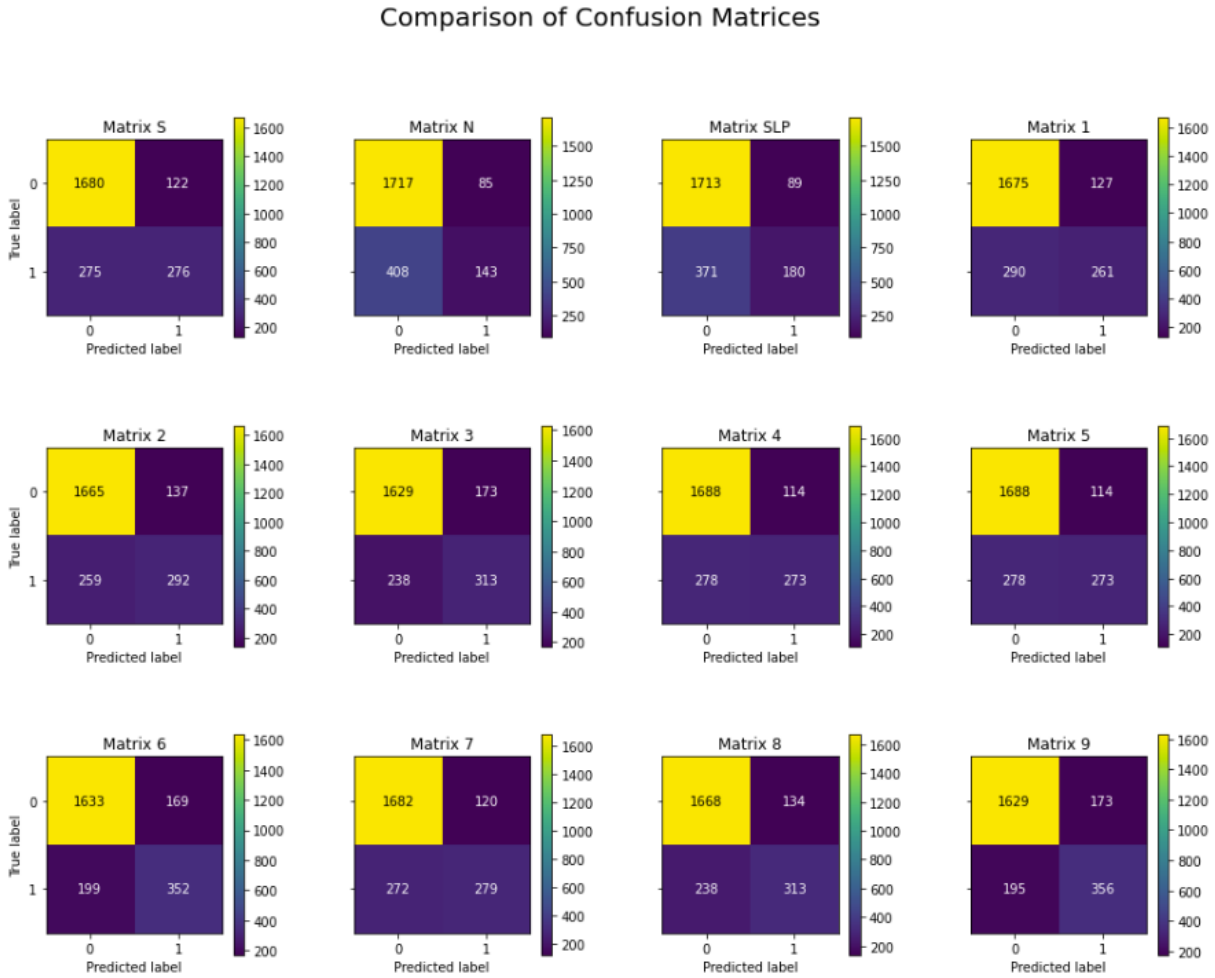


Figure 8. Confusion Matrices



As it becomes clear from the graphs above, and observing with attention the false-negatives and recall metrics, the worst model constructed so far is the model with the data normalised, followed by the Perceptron model who shows results similar to the normalised model with high test-loss and false-negatives and very low recall.

Comparing the number of hidden layers on the models, it is evident that the models with only one hidden layer on their architecture - independent of the number of nodes - had a worse outcome than the ones with 2 or 3 layers.

Taking into consideration the metrics more suited for the marketing problem in hand, models 9 and 6 were the most successful of them all so far. Both models with 3 hidden layers with 64 and 32 nodes, successively, had the lowest false-negatives and highest recall values.

After comparing models by number of hidden layers and nodes, one hyperparameter was still changed and compared, and that is the activation of the hidden layers. The top 3 performing models (Model 9, 6 and 5 respectively) were tested on different activations, ‘relu’ or ‘tanh’ (Fig. 9 and 10).

Figure 9. Comparison of confusion matrix results for change in activation

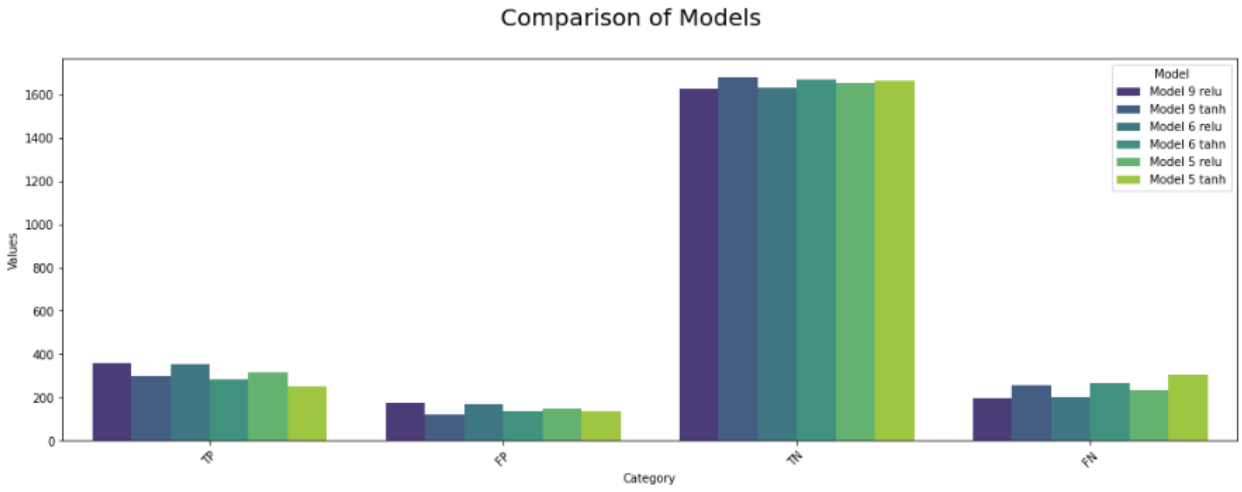
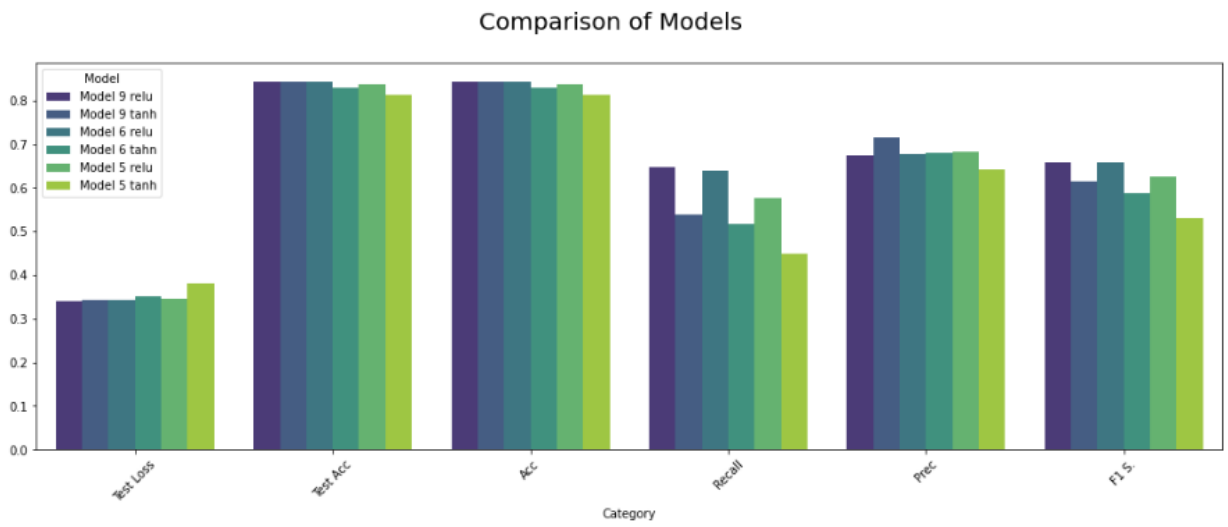


Figure 10. Comparison of evaluation metrics results for change in activation



As demonstrated above, the activation ‘relu’ brings better results for the models in comparison to the ‘tanh’ activation; it provided lower false-negatives results and higher recall.

Overall, the results indicate that the implemented feedforward neural network model 9 (with ‘relu’ activation) achieved the optimal performance in its architecture construct for the binary classification task.

However, it is always necessary to take into consideration on a real world task the whole construct of the model, not only its architecture - which was the goal and focus of this project at hand. Further optimization and fine-tuning of hyperparameters on the steps following the architectural structure could enhance its performance, for example changing the optimizer during compiling and by a regularisation method. Therefore, Model 9 might have obtained the best architectural results, but an overall analysis considering, for example, complexity, budget and overfitting, might point to a different model.

5. References

Buduma, N. (2017) Fundamentals of Deep Learning. O’Reilly Media, Inc.

Chollet, F. (2018) Deep Learning with Python. New York, NY: Manning Publications.

Géron, A. (2019) Hands-On Machine Learning with Scikit-Learn, Keras & Tensorflow. O'Reilly Media, Inc.

Moro, S., Laureano, R. and Cortez, P. (2014) Bank Marketing Data Set, UCI Machine Learning Repository. Available at: https://archive.ics.uci.edu/ml/datasets/bank+marketing

6. Appendices

Appendix A: Code for Assignment 1

The code developed for this assignment was stored in GitHub and can be accessed through the following link: CETM25_Assigment1