Neural Computing – CS4287 Assignment01: Multi-Layer Perceptron

21344256: Kevin Collins

21318204: Olan Healy

Contents

The Data Set

We chose to use the "**Are Your Employees Burning Out?"** dataset we found on Kaggle. We felt this was a fitting dataset for this assignment as World Mental Health day took place on the 10th of October. Mental health is a very serious issue in the world which we feel strongly about and we wanted to build a model based on this. It contains 9 datatypes; Employee ID, Date of Joining, Gender, Company Type, WFH Setup Available, Designation, Resource Allocation, Mental Fatigue Score and Burn Rate.

Clarifying some of the data:

- Company Type: The type of company where the employee is working (Service/Product)
- WFH Setup Available: Is the work from home facility available for the employee (Yes/No)
- Designation: The designation of the employee of work in the organisation. In the range of [0.0, 5.0] bigger is higher designation.
- Resource Allocation: The amount of resource allocated to the employee to work, i.e.. number of working hours. In the range of [1.0, 10.0] (higher means more resource)
- Mental Fatigue Score: The level of fatigue mentally the employee is facing. In the range of [0.0, 10.0] where 0.0 means no fatigue and 10.0 means completely fatigue.
- Burn Rate: The value we need to predict for each employee telling the rate of Bur out while working. In the range of [0.0, 1.0] where the higher the value is more is the burn out.

The dataset also included this data on employees for 22,751.

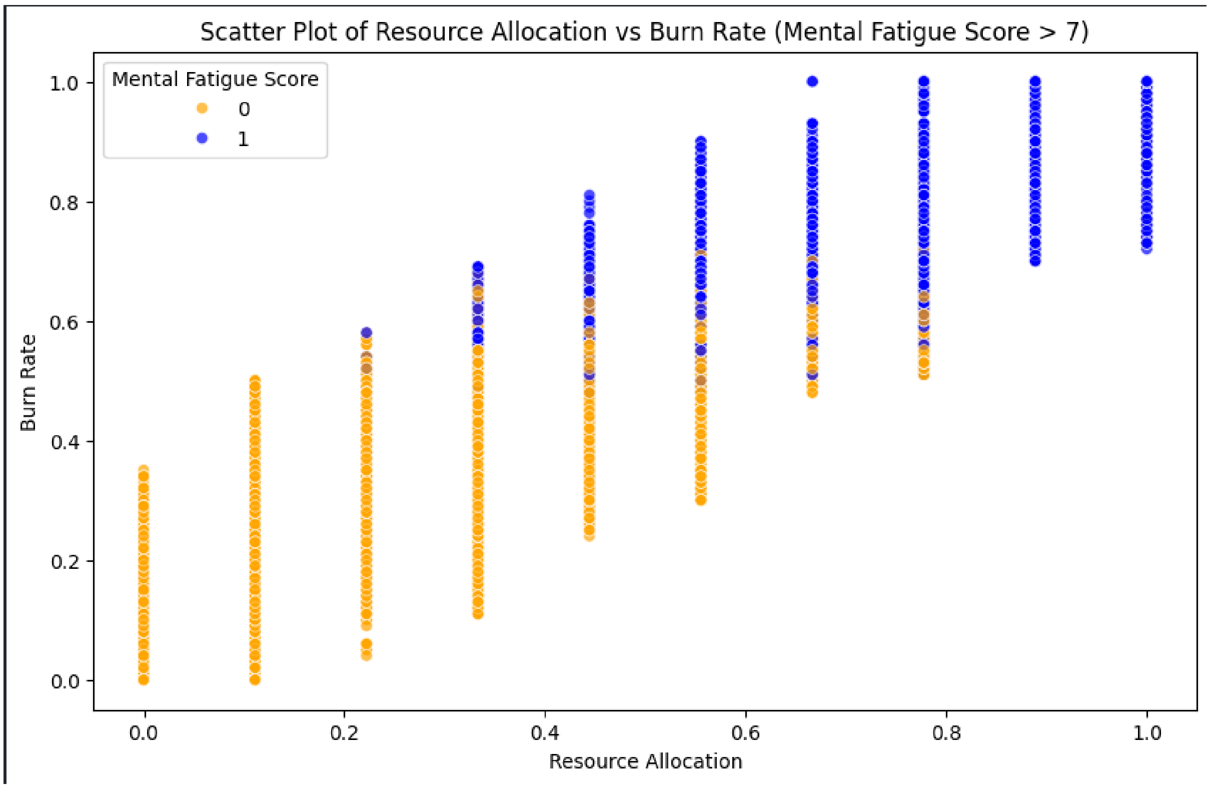


Fig 1.0: Scatter Plot showing data is not linearly separable.

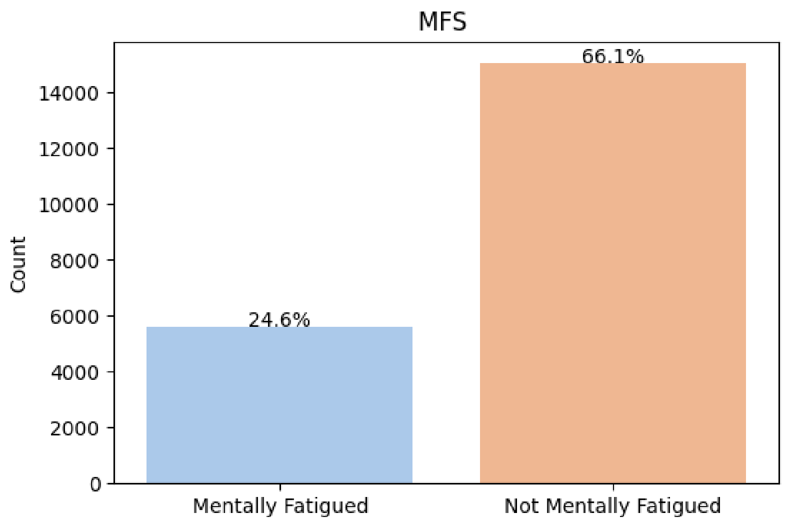


Fig 1.1: Shown here is the proportion of people in the dataset mentally fatigued (>=7) to not mentally fatigued < 7. 24.6% of people in the dataset were mentally fatigued, 66.1% were not mentally fatigued. The other 9.3% didn't have a value in the dataset so we chose to remove these employee's which left us with 20,498 employees

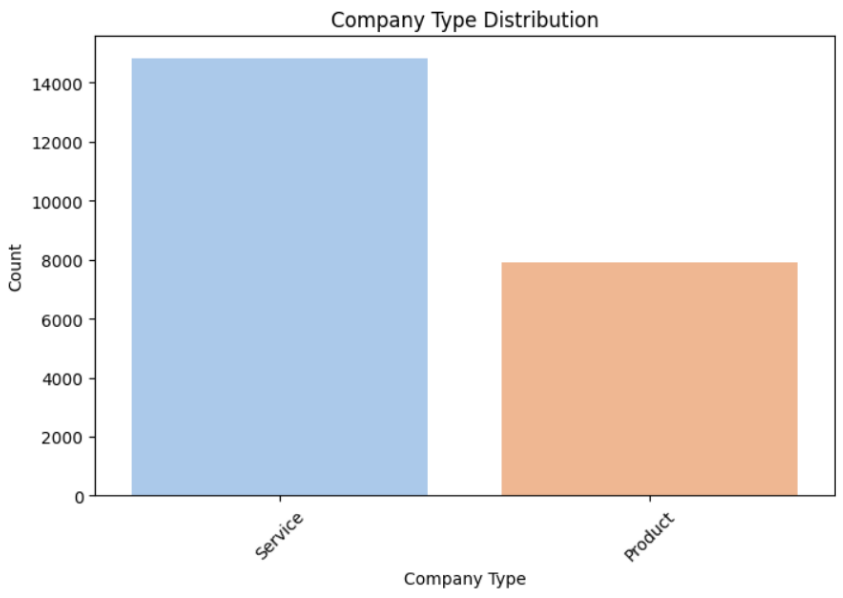


Fig 1.1: Shown here is the portion of service to product companies in the dataset. We believe that due to the large dataset we should be able to accurately judge both company types effect on Mental Fatigue

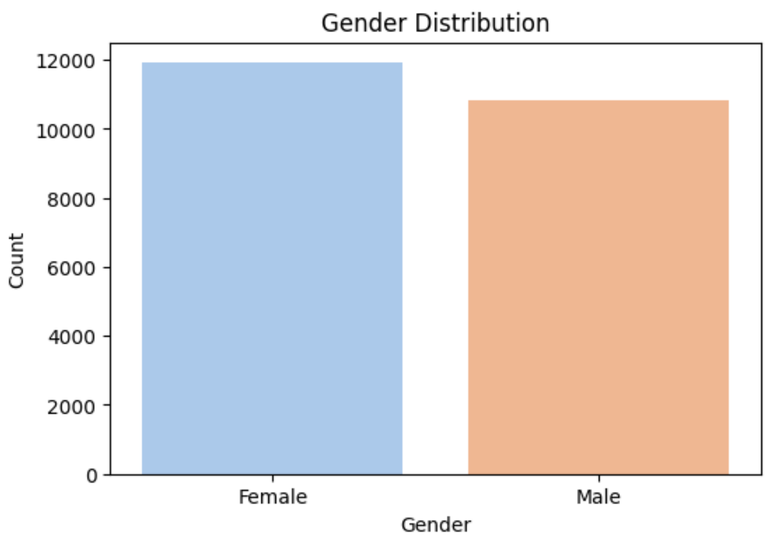


Fig 1.2: Gender is distributed close to even so it is safe to say that we will get an accurate picture of Mental Fatigue on both Genders.

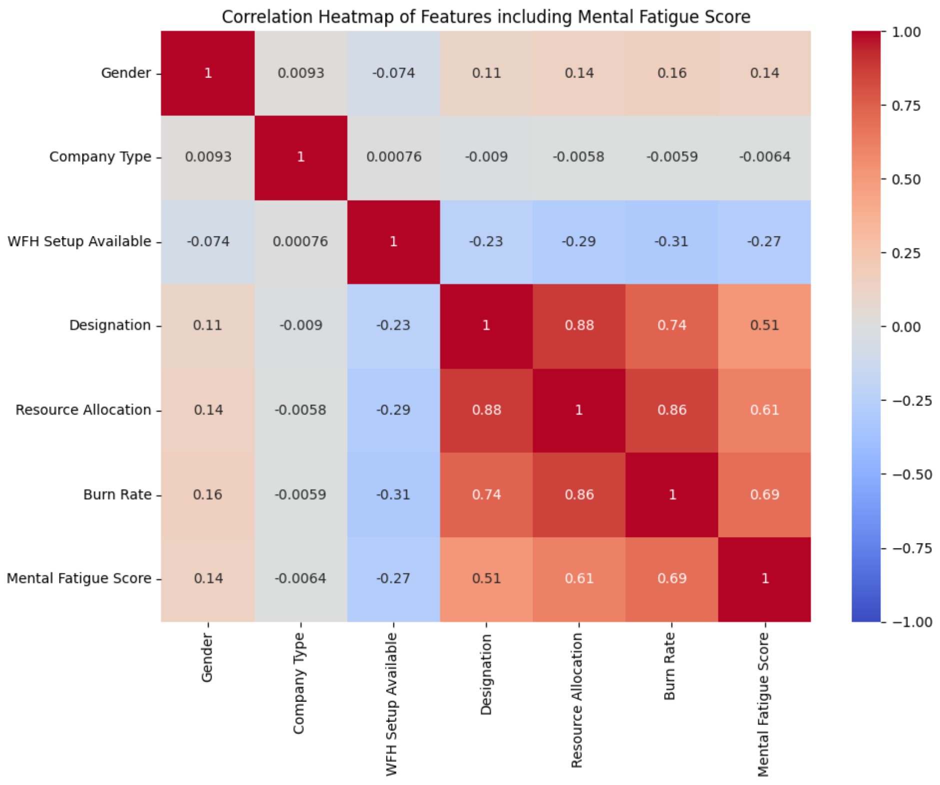


Fig 1.3: Shown here is a Heatmap denoting the correlation. As can be seen in the graph, Burn Rate has a great effect on the mental fatigue score whereas Gender has little to no correlation.

Normalisation

We chose to use a MinMaxScaler as it scales the features to a fixed range, usually [0, 1]. For our dataset, the categorical features had to be converted into binary values (e.g., Gender, Company Type, WFH Setup Available), which are 0 or 1. Since MinMaxScaler maintains these

0 and 1 values, it integrates smoothly with this binary encoding and keeps the categorical features in the proper format.

$$X_{\text{scaled}} = \frac{X - X_{\text{min}}}{X_{\text{max}} - X_{\text{min}}}$$

Fig 1.4: MinMaxScaler Formula

Network Structure and other hyperparameters

The model is composed of three layers:

Input Layer: The input dimension is set to match the number of features in the training dataset (input_dim = X_train.shape[1]). The input layer consists of 128 neurons with a ReLU Rectified Linear Unit) activation function. We chose ReLU to introduce non-linearity and allow the network to learn complex relationships.

Hidden Layer: A single hidden layer with 64 neurons also uses the ReLU activation function. This layer processes the intermediate representations of the data, allowing the network to extract higher-level features relevant to the classification task.

Dropout Layer: To mitigate overfitting, a dropout layer is incorporated after the input layer, with a dropout rate of 0.5, meaning 50% of the neurons are randomly dropped during training. This prevents the model from becoming overly reliant on specific neurons and helps generalise better on unseen data.

Output Layer: The output layer has a single neuron with a sigmoid activation function, which outputs a value between 0 and 1, corresponding to the probability of the sample belonging to the positive class (high mental fatigue). This is suitable for binary classification tasks.

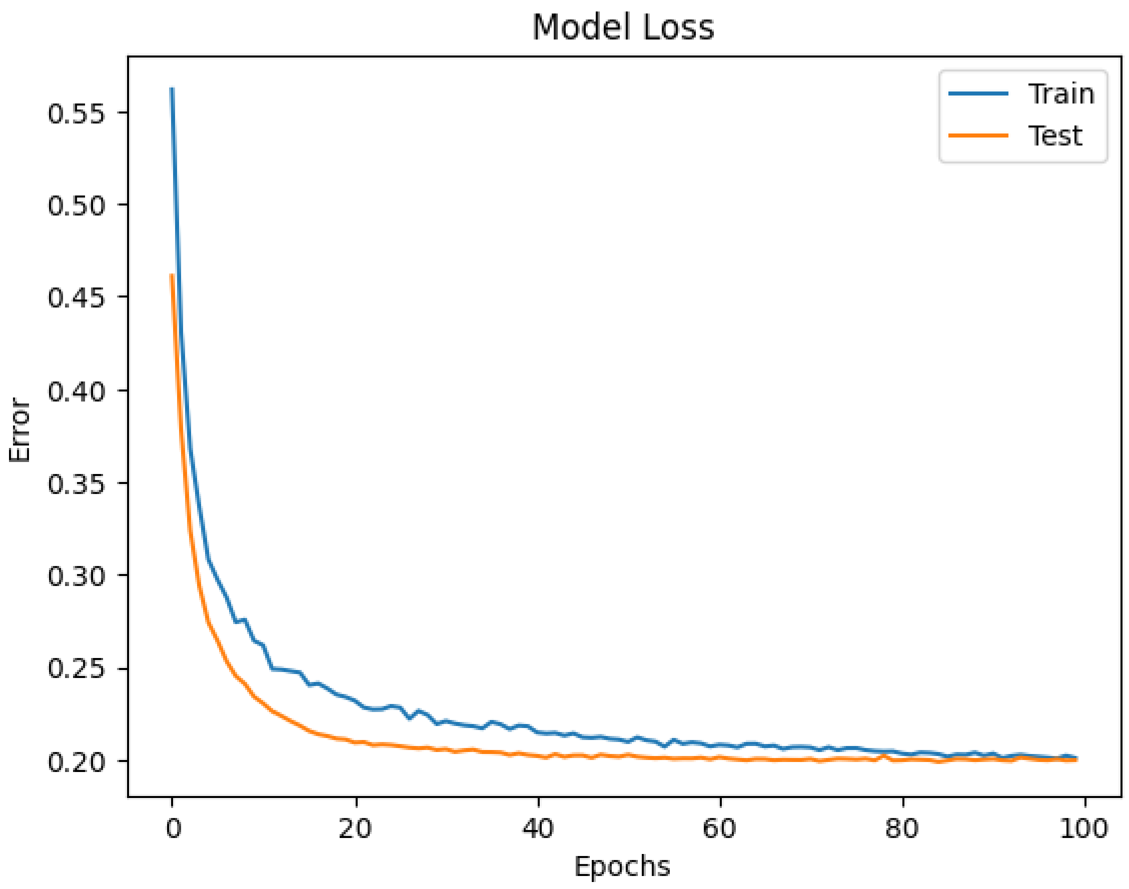
Hyperparameters:

Loss Function: The model is trained using the binary cross-entropy as the loss function.

Epochs and Batch Size: The model is trained for 100 epochs with a batch size of 16, balancing the computational efficiency and the rate at which the model learns. We noticed that while training, the model gained no extra learning so we decided to reduce it to 100 to save the amount of time it had to run for. We originally had it set to 200 epochs.

Validation Split: During training, 20% of the training data is reserved for validation (validation_split=0.2), allowing the model to be evaluated on unseen data at each epoch, ensuring that the model is not overfitting to the training data.

Cost / Loss / Error / Objective function



This loss curve shows that the model performs well without severe overfitting/underfitting problems. The validation loss closely follows the training loss, which means the model can generalise to unseen data. This is healthy training.. The data set is large and it adapts quickly to data and the errors drop rapidly.

As mentioned with the hyperparameters, we used binary cross-entropy as the loss function. We used this as binary cross entropy is specifically designed for binary classification problems, which works with us deciding if someone is A. Mentally Fatigued or B. Not Mentally Fatigued. The formula for this is

$$\text{Loss} = -\frac{1}{N}\sum_{i=1}^{N}[y_{\text{true}}\log(y_{\text{pred}}) + (1 - y_{\text{true}})\log(1 - y_{\text{pred}})]$$

Where:

N = Samples in dataset

Y_true = Actual label for each sample (1 for mentally fatigued, 0 for not mentally fatigued)

Y_pred = Predicted probability from the model for each sample

Log = Natural log function

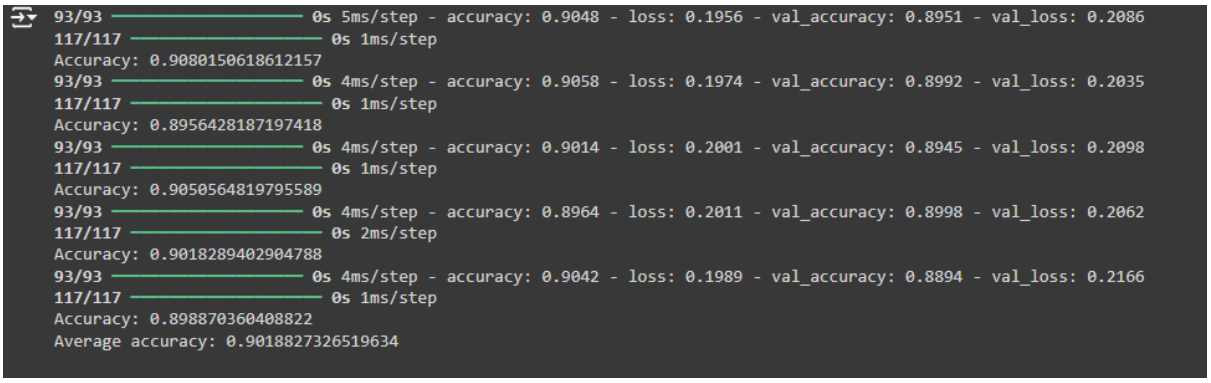
Compared to a loss function such as mean squared error (MSE) (which we originally had used), which is more appropriate for regression tasks, the binary cross-entropy is designed to handle probabilistic predictions, which our model purpose is for. It was also a better choice than say categorical cross-entropy which we used in Lab 3, as we were only dealing with two possible classes. Categorical cross-entropy is used for multi-class classification tasks such as classifying an animal into horse, dog, rabbit.

Optimiser

We used Adam as our optimiser. Adam builds upon two popular techniques, AdaGrad and RMSprop. It is an adaptive learning rate algorithm. We initially set to a learning rate of 0.001. Why Adam is useful is because it adapts the learning rate for each parameter dynamically based on the estimated of the first and second moments of gradients. We felt we should use Adam compared to other optimisers because of this adaptive learning rate as it can ensure efficient learning. It also is computationally efficient and requires little memory, which was good for us as our dataset contains many rows of data

We could have used AdaGRad or RMSprop but Adam combines the strength of both techniques. Stochastic Gradient Descent (SGD) could have been used but we would have had to carefully tune the learning rate which could have cost a lot of time. Adam also incorporates bias correction which is good for improving gradient estimations early in training.

Cross Fold Validation



```
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 5ms/step - accuracy: 0.9048 - loss: 0.1956 - val_accuracy: 0.8951 - val_loss: 0.2086
117/117 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Accuracy: 0.9080150618612157
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9058 - loss: 0.1974 - val_accuracy: 0.8992 - val_loss: 0.2035
117/117 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Accuracy: 0.8956428187197418
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9014 - loss: 0.2001 - val_accuracy: 0.8945 - val_loss: 0.2098
117/117 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Accuracy: 0.9050564819795589
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.8964 - loss: 0.2011 - val_accuracy: 0.8998 - val_loss: 0.2062
117/117 ━━━━━━━━━━━━━━━━━━━━ 0s 2ms/step
Accuracy: 0.9018289402904788
93/93 ━━━━━━━━━━━━━━━━━━━━ 0s 4ms/step - accuracy: 0.9042 - loss: 0.1989 - val_accuracy: 0.8894 - val_loss: 0.2166
117/117 ━━━━━━━━━━━━━━━━━━━━ 0s 1ms/step
Accuracy: 0.898870360408822
Average accuracy: 0.9018827326519634
```

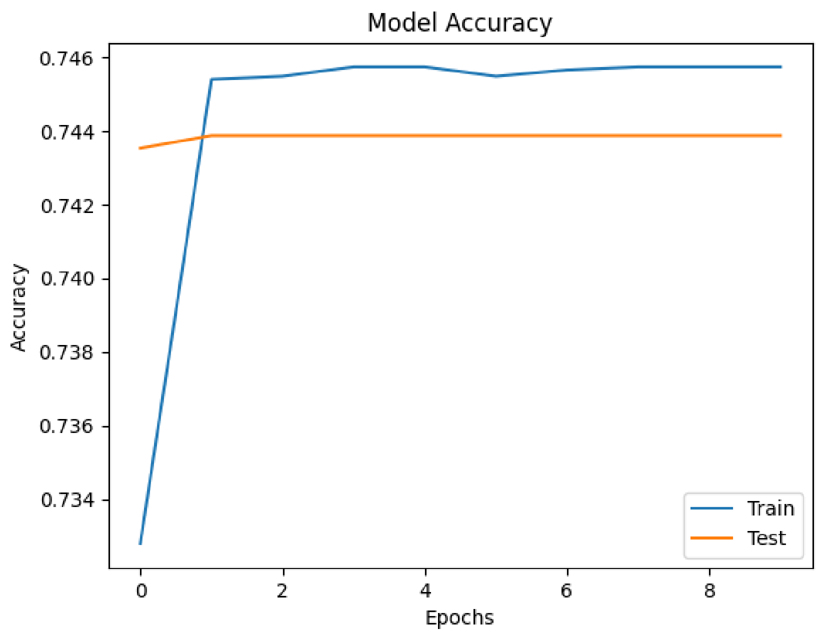
We split our data into 5 folds using K-fold Cross Validation. Across the 5 folds, the training accuracy of our model was consistent in the ranges between 0.89 to 0.908. This shows that our model is learning well from the training data. Our validation accuracy is in a similar range and is showing good consistency across the folds. This indicates that our model is generalising well to unseen data and is not overfitting. Our validation loss is in an around the 0.20 mark. This suggests that our models' predictions are reasonably close to the true values. Overall these results suggest our model is well tuned and can make accurate predictions on if someone is mentally fatigued or not.

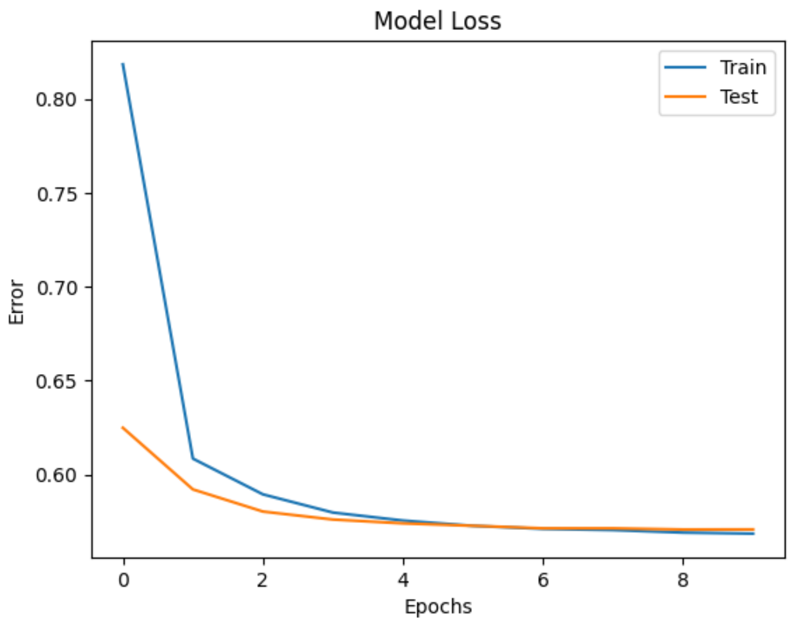
Impact of varying a hyperparameter

Changing the Epochs from 100 to 10

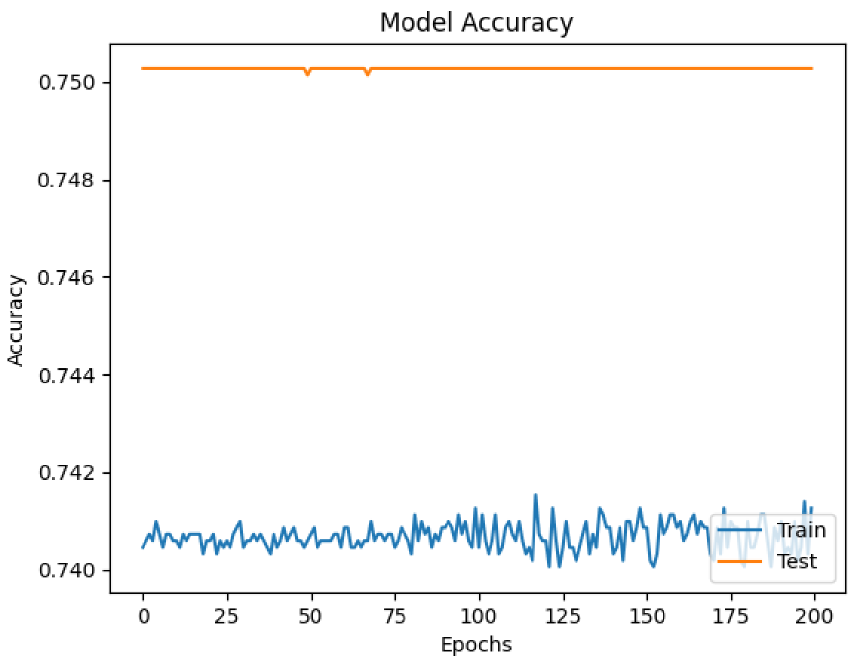
With only 10 epochs, our model does not have enough time to adequately learn the underlying patterns in the training data. As a result, our model is likely to exhibit underfitting, where it fails to capture the complexities of the data, leading to poor performance on both the training and validation datasets.

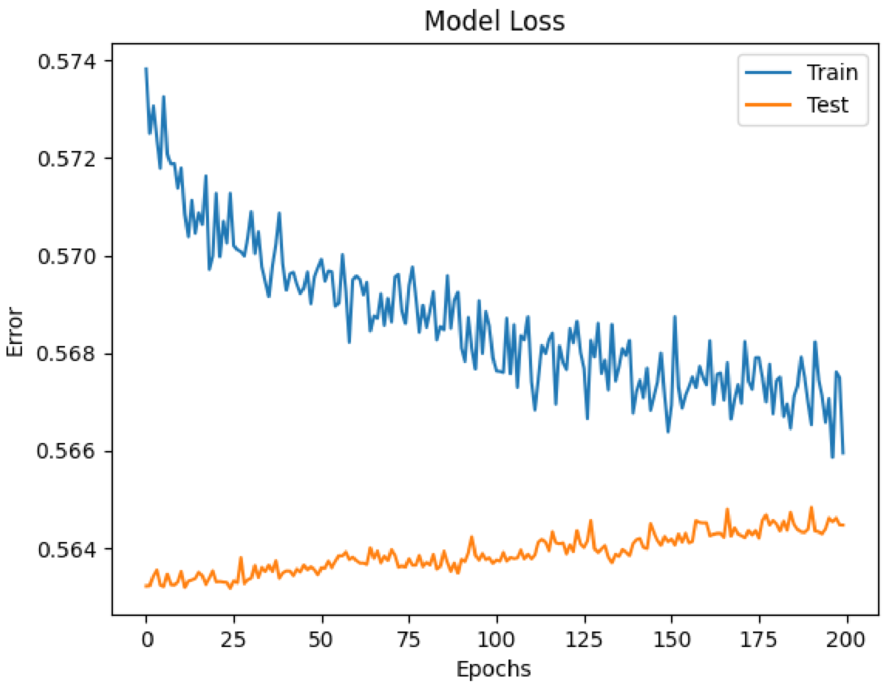
You can also see a higher training loss at the end of training compared to when using 100 epochs. The loss function, which measures how well the model's predictions match the actual outcomes, does not decrease significantly, indicating that our model hasn't effectively learned from the data.



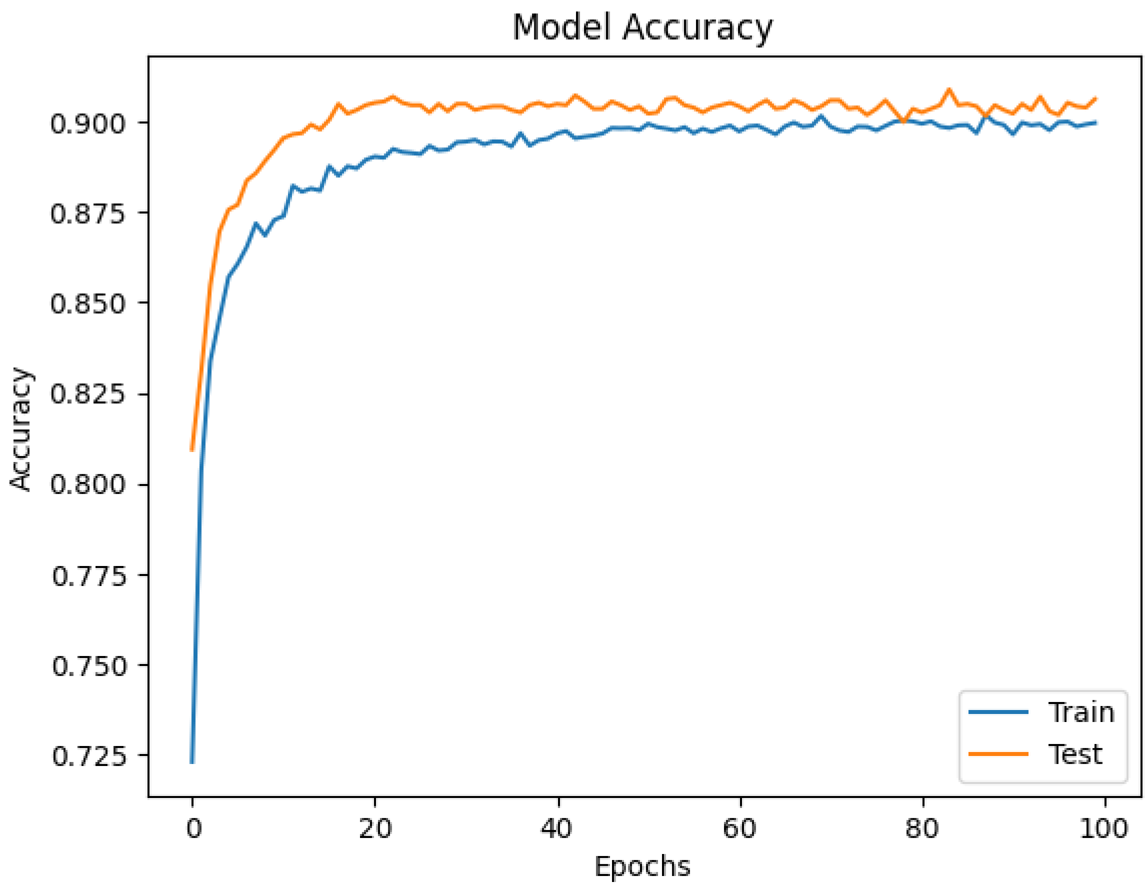


We then tried changing the validation split from 80/20 to 50/50. Changing the validation split to 50/50 while maintaining 100 epochs has several implications for your neural network model performance. Firstly, this adjustment increases the validation dataset, so we can get a more complete and reliable assessment of our model's generalisation capability based on unseen data. However, this change reduces the training dataset by half as well, which can lead to underfitting if our model fails to learn from the smaller subset. A larger validation set provides more informative feedback during hyperparameter tuning and allows for better decision making regarding model architecture and regularisation techniques. The accuracy only remains around 75% as opposed to up to 90% when split 80/20.


Model Accuracy
Accuracy
0.750
0.748
0.746
0.744
0.742
0.740
0
25
50
75
100
125
150
175
200
Epochs
Train
Test

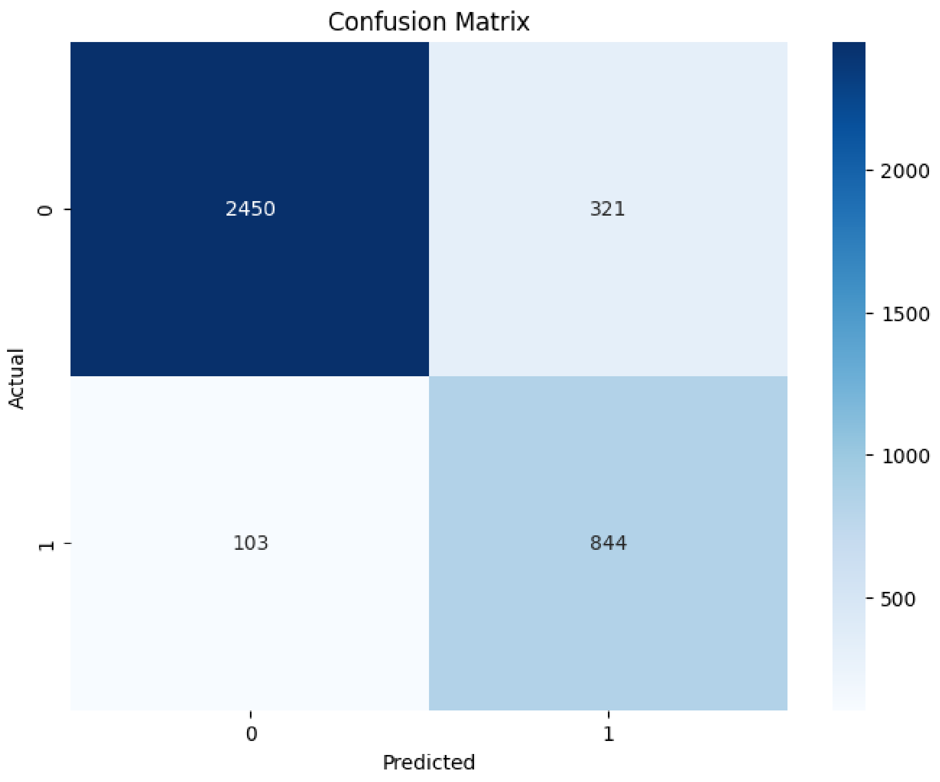

Model Loss
Error
0.574
0.572
0.570
0.568
0.566
0.564
0
25
50
75
100
125
150
175
200
Epochs
Train
Test

Results



This graph reports training and test data accuracy over 100 epochs. In the initial epochs, the model accuracy improves rapidly to about 0.89 for both training and test sets. Test data accuracy starts slightly higher than training data accuracy (in blue) and remains above it throughout the epochs - a positive sign of no overfitting. Both curves stabilise around 0.90, suggesting that the model learned from the data without significant variance between training and testing performance. Small fluctuations in accuracy are to be expected but do not represent major instability. The overall trend shows strong generalisation, as performance on unseen data (test) roughly matches that on the training set.

[0,0]	True Negative	2450	Predicted an employee was not mentally fatigued, and they weren’t
[0,1]	False Positive	321	Predicted an employee was mentally fatigued, but they weren’t
[1,0]	False Negative	103	Predicted an employee was not mentally fatigued, but they were
[1,1]	True Positive	844	Predicted an employee was mentally fatigued, and they were



Evaluation of Results

Precision: This measures how many of the positive predictions made by our model were actually correct. So how many mentally fatigued employees were actually predicted to be mentally fatigued

Formula: True positive / (True Positive + False Positive)

844 / (844 + 321) = 0.72

So out of all the employees, our model predicted out of all our employees that 82% were actually mentally fatigued. This is good and shows our model is fairly reliable when it comes to predicting mentally fatigued employees and minimises the rate of false positives

Recall: This measures the true positivity rate, so how many mentally fatigued employees were actually correctly identified by our model

Formula: True positive / (True Positive + false negative)

844 / (844 + 103) = 0.89

So this means our model correctly identified 78.6% of our employees who are actually mentally fatigued. This is also good and shows our model is good at capturing most of the mentally fatigued employees and minimises cases where they were missed.

References and Notes

Kaggle Dataset: https://www.kaggle.com/datasets/blurredmachine/are-your-employees-burning-out/data

Seqential Model: The Sequential model (keras.io)

Binary Cross Entropy tf.keras.losses.BinaryCrossentropy | TensorFlow v2.16.1

K-Fold Cross Validation Cross Validation in Machine Learning - GeeksforGeeks

MiniMaxScaler MinMaxScaler — scikit-learn 1.5.2 documentation

Adam Optimiser Adam (keras.io)

We used the sample submission as a guide