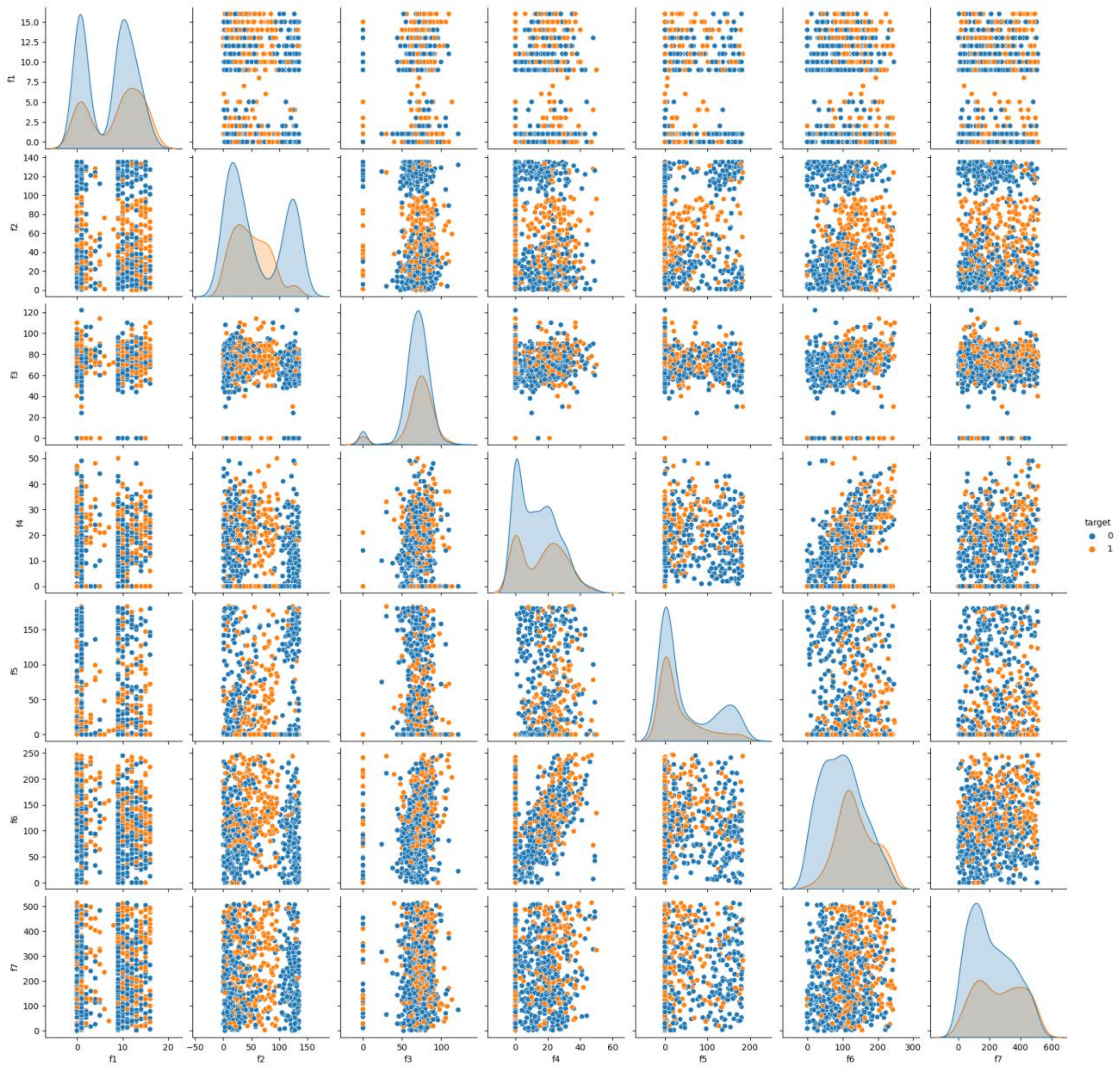
Introduction to Machine Learning

Assignment 2 Part 1 Report

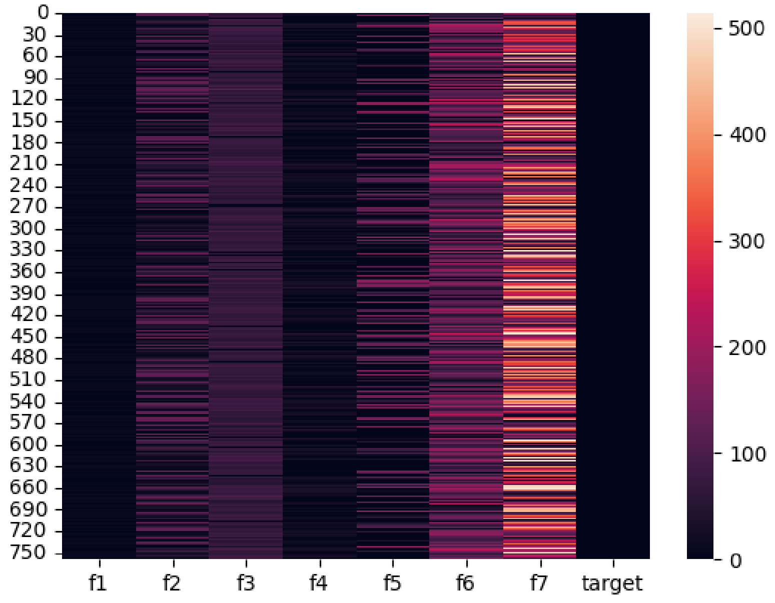
1. Provide brief details about the nature of your dataset. What type of data are we encountering? How many entries and variables does the dataset comprise? Provide the main statistics about the entries of the dataset.

- The dataset comprises 8 columns and 766 rows. It has 2 numerical columns (f3 and target), 6 category columns (f1, f2, f4, f5, f6, and f7). The collection does not include any null values, but several of its columns have values that are letters rather than integers. We eliminate these rows so that there are 760 rows and 8 columns after preprocessing.

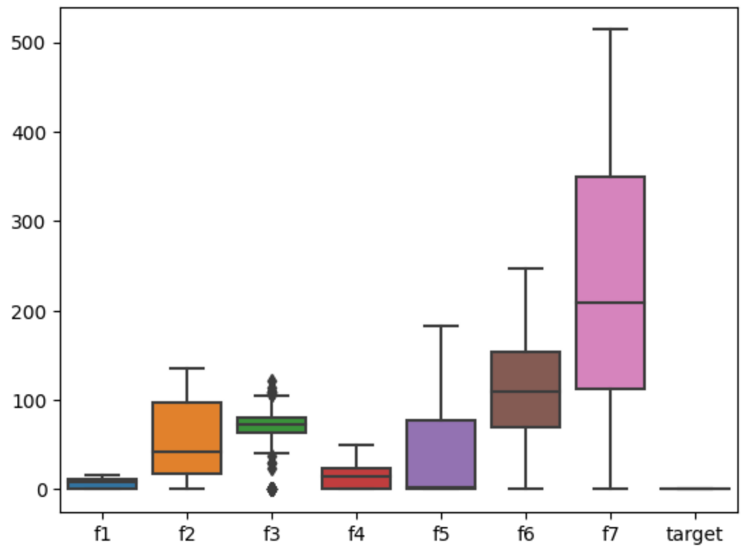
2. Provide at least 3 visualization graphs with short description for each graph.



This is a pairplot against f1, f2, f3, f4, f5, f6, f7 columns with target column. It describes how ech column is related to the target column.



This is the heatmap which used to show correlation between 2 features. The darker color indicates that the features are more corelated and lighter shades suggest less correlation.



This is the boxplot. F3 is the only column which has outliers.

3. For the preprocessing part, discuss if you use any preprocessing tools, that helps to increase the accuracy of your model.

For preprocessing we first convert categorical values to numerical values for which we used LabelEncoder. We also removed all the rows which consisted of letters (just 6 rows). Then we scaled the numerical variables to have zero mean and unit variance. Then we split the data into training and testing.

4. Provide the architecture structure of your NN.

Four fully connected layers with ReLU activation functions make up the majority of this neural network architecture, which is followed by a single output layer. It is suitable for regression tasks because it takes 7 input features and produces a single scalar value.
Using nn.Flatten(), which changes a tensor of shape (batch_size, C, H, and W) into a tensor of shape (batch_size, C * H * W), the input data is flattened.
Following the flattened input, a sequence of completely connected layers with ReLU activation functions are applied:
The nn.Linear(7, 64) call specifies that the first layer has 64 output features and 7 input features.
64 input features and 128 output features are present in the second layer.
64 output features and 128 input features make up the third layer.
According to the call to nn.Linear(64, 1), the final layer has 64 input features and 1 output feature.
The model's forecast for the target variable is the output of the last layer.

5. Provide graphs that compares test and training accuracy on the same plot, test and training loss on the same plot. Thus in total two graphs with a clear labeling.

Assignment 2 Part 2 Report

1. Include all 3 tables with different NN setups.

"NAME OF THE HYPERPARAMETER" Tuning

	Setup 1	Test Accuracy	Setup 2	Test Accuracy	Setup 3	Test Accuracy
Dropout	0.4	67.11%	0.5	77.63%	0.3	71.05%
Optimizer	optim.Adam(model.parameters(), lr=0.001)		optim.Adam(model.parameters(), lr=0.001)		optim.Adam(model.parameters(), lr=0.001)	
Activation Function Initializer	MSELoss()		MSELoss()		MSELoss()	

2. Provide graphs that compares test and training accuracy on the same plot for all your setups and add a short description for each graph.

3. Provide a detailed analysis and reasoning about the NN setups that you tried.

DROPOUT 0.4

The first model tried is a basic neural network architecture with batch normalization. It consists of four fully connected layers, each followed by a batch normalization layer and a ReLU activation function. The final layer has a single output, which is suitable for regression problems. The model is trained using the Adam optimizer and Mean Squared Error (MSE) loss function.

The second model tried is a similar neural network architecture but with dropout regularization added after each ReLU activation function. Dropout is a regularization technique that randomly sets some of the neuron outputs to zero during training, which helps to prevent overfitting. The dropout rate used is 0.4, which means that each neuron output has a 40% chance of being set to zero during training. The model is again trained using the Adam optimizer and MSE loss function. The same evaluation criteria, such as MSE, Mean Absolute Error (MAE), and accuracy, are used to compare the two models on the same test set. By thresholding the expected output at 0.7 and comparing it with the ground truth label, the accuracy is determined. It is unclear why 0.7 was selected as the threshold, and it might not be appropriate in all circumstances. On the given test set, both models seem to perform about similarly, with the second model having a little lower MAE and higher accuracy. It's crucial to keep in mind that the test set could not be representative of the complete dataset and that the models' performance may vary when applied to unobserved or additional data.

For dropout 0.4 we get the test accuracy as 67.11%

DROPOUT 0.5

In this configuration, we employed a neural network with three hidden layers that each had 64, 128, and 64 neurons. In order to avoid overfitting, we employed the ReLU activation function and a dropout layer with a probability of 0.5 between each hidden layer. One neuron with no activation function was seen in the output layer. The model was trained over the course of five epochs using the Adam optimizer and the Mean Squared Error (MSE) loss function. In the end, we assessed the model using the test set.

For dropout 0.5 we get the test accuracy as 77.63%

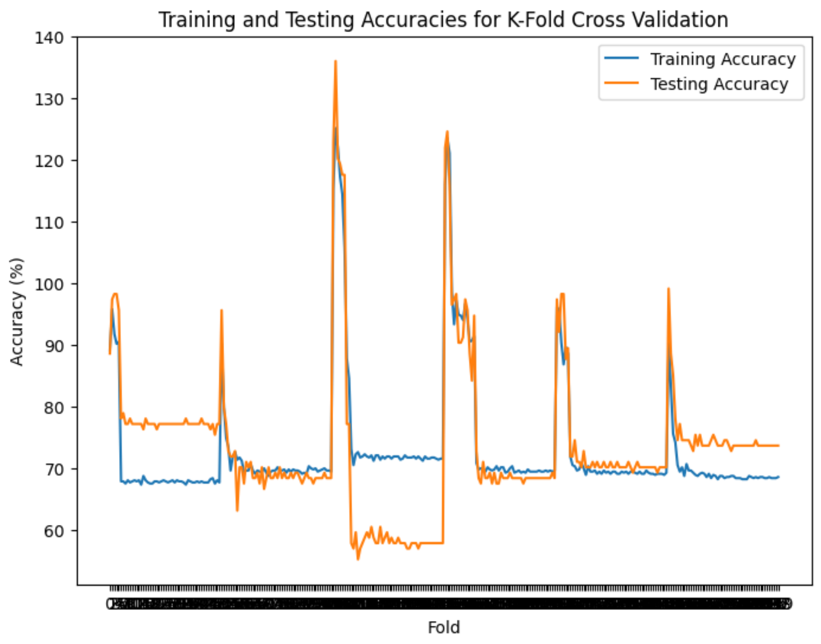
DROPUT 0.3

This code creates a neural network model for binary classification on input data with seven characteristics using PyTorch's nn.Module class. The model architecture consists of four fully connected layers with dropout regularization applied with a probability of 0.3 after each ReLU layer. A custom function called "train" is used to train the model across 5 iterations using a given optimizer and loss criterion. Finally, accuracy, mean squared error, and mean absolute error metrics are used to assess the trained model on a test set.

For dropout 0.3 we get the test accuracy as 76.32%

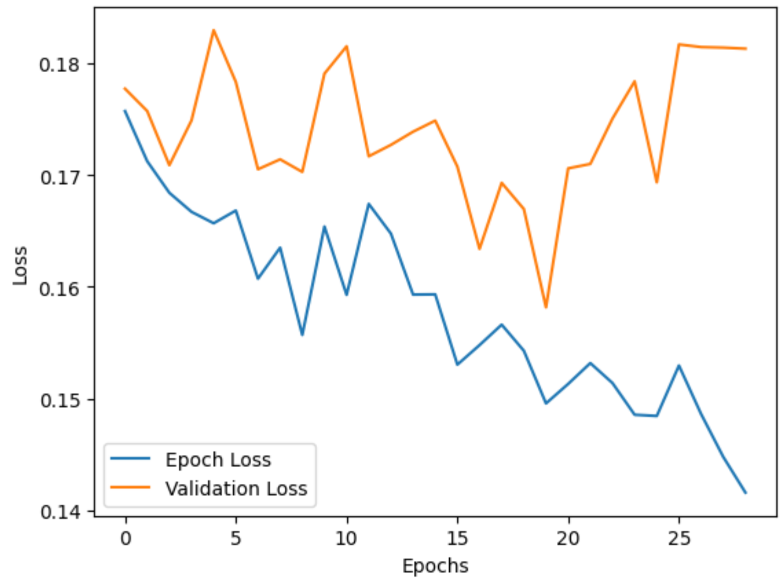
4. Briefly discuss all the methods you used that help to improve the accuracy or training time. Provide accuracy graphs and your short descriptions.

The first method we use is kfold. In order to solve a binary classification problem, the algorithm applies a K-fold cross-validation. The data is divided into training and testing sets, a neural network model, optimizer, and loss function are initialized, the model is trained for a predetermined number of epochs, and accuracy on both the training and testing sets is assessed for each fold. It then figures out the average accuracy across all folds and prints it.



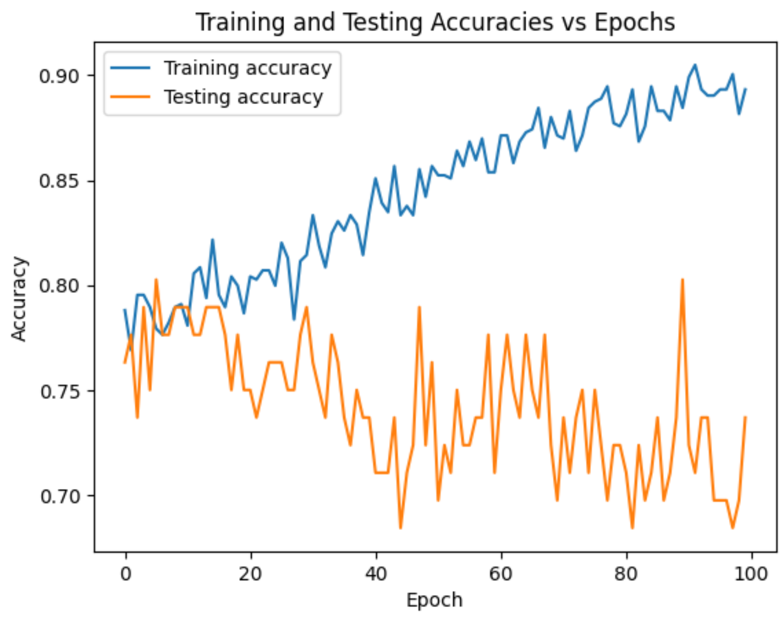
This is the graph for training and testing accuracy using kfold method

The second method we use is early stopping. To train a PyTorch neural network model using MSE loss and Adam optimizer, the code defines a training function with early stopping. The function computes the validation loss after each epoch of training the model for a predetermined number of epochs. The training is terminated early to avoid overfitting if the validation loss does not improve after a predetermined number of epochs (patience). The model is then trained using the defined function, and its performance is then assessed using the test set to determine the test loss and accuracy. The user can print the output to see the test accuracy and loss.



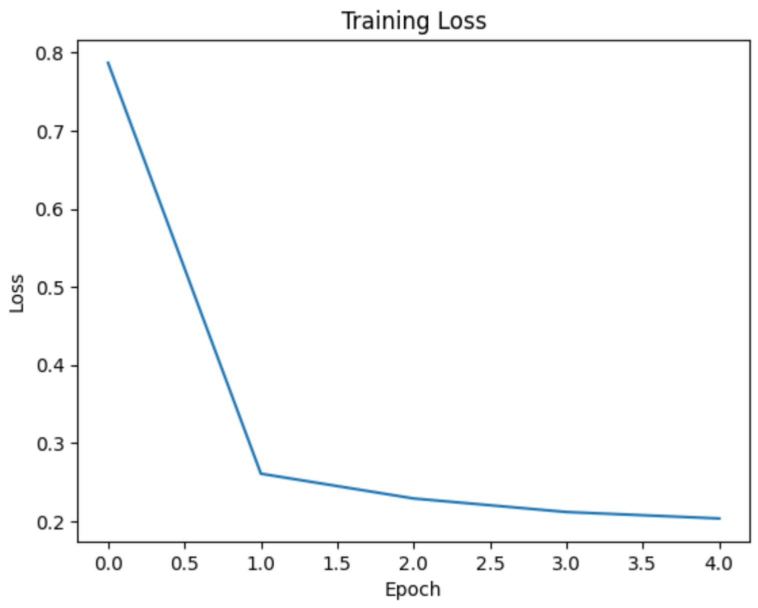
This graph shows the epoch loss and validation loss from early stopping

The **third method we use is binary classification**. This Python tool uses a binary classification dataset to train a PyTorch neural network model. An input layer, two hidden layers with ReLU activation, and an output layer with a sigmoid activation function make up the model's architecture. The programming language specifies a training function that iteratively goes through the training data, computes the loss, and backpropagates to update the model parameters using the Adam optimizer. On the training and testing datasets at each epoch, the training function also calculates the model's accuracy. The model is then assessed using the test set, the loss and accuracy are calculated, and the results are printed. The epoch number with the best training and testing accuracies is then printed, along with the corresponding accuracy numbers.



This graph shows training and testing accuracies vs epochs

The **fourth method is batch normalization.** This code defines a neural network model with batch normalization layers, trains the model using the mean squared error loss function and Adam optimizer, and evaluates the model on a test set using mean squared error, mean absolute error, and accuracy metrics. The model is trained on a training set for a specified number of epochs, with the loss for each epoch recorded. The code also includes a print statement for the loss for each epoch during training.



This graph shows the training loss.