CASE STUDY 6:
SUPERCONDUCTORS DENSE NEURAL NETWORK

BY: ANDREEA CAROLINA CRAUS

1 INTRODUCTION

In the particle physics, the identification of new superconductor particles can be a challenge. Superconductors are materials that give little or no resistance to electrical currents without an onset or buildup of heat, creating a magnetic field that generates a constant flow of electricity. In this case study, neural networks and classifiers are applied to an extensive superconductor dataset with the primary objective of predicting the existence of a new particle with a high level of accuracy. Binary detection is employed with the target variable having two different responses, “detection” (denoted by 1) and “non-detection” (denoted by 0).

2 METHODS

2.1 MODEL PREPARATION

The first step in preparing the data for modeling is scaling the data in order to improve the convergence speed and stability of the training process. Scaling ensures that the weights of the neural network are updated uniformly across all features during training so that features with larger magnitudes will not dominate the learning process, which can lead to suboptimal solutions or unstable training behaviors. The MinMaxScaler was used for this task which scales and translates each feature individually so it is in the range [0,1] by subtracting the minimum value of the feature and then dividing the difference between the maximum and minimum values, effectively mapping the original feature values to the range. This still preserves the shape of the original distribution which ensures that the relative relationships between feature values are maintained.

2.2 DATA EXPLORATION & PREPROCESSING

The dataset provided consisted of 7 million instances and 27 features of different properties associated with detecting new particles. An initial look at the target variable distribution shows an equal balance of “detection” and “non-detection” labels, shown in the histogram in Figure 1. The data contained no missing values and all features are numerical.

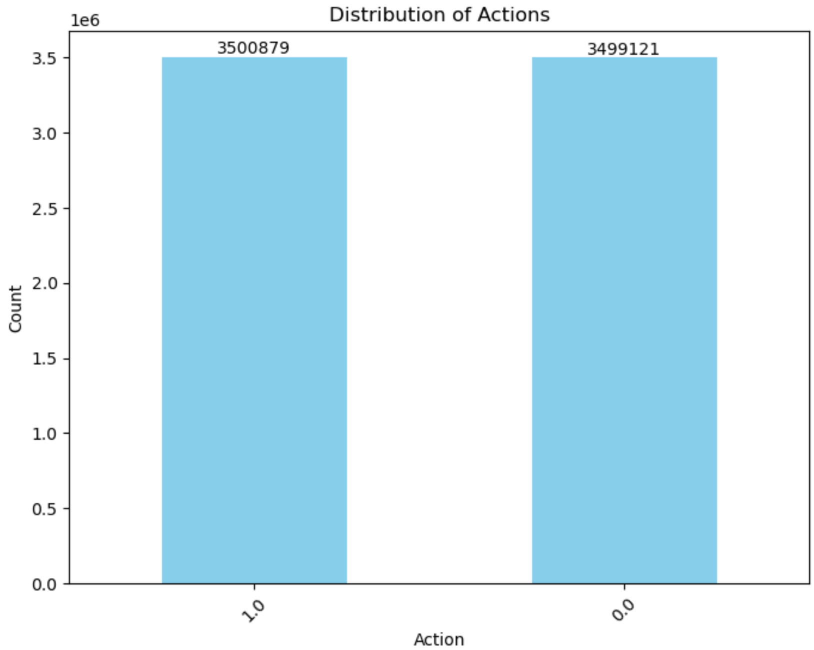

Figure 1: Histogram of Target Variable Distribution of Labels

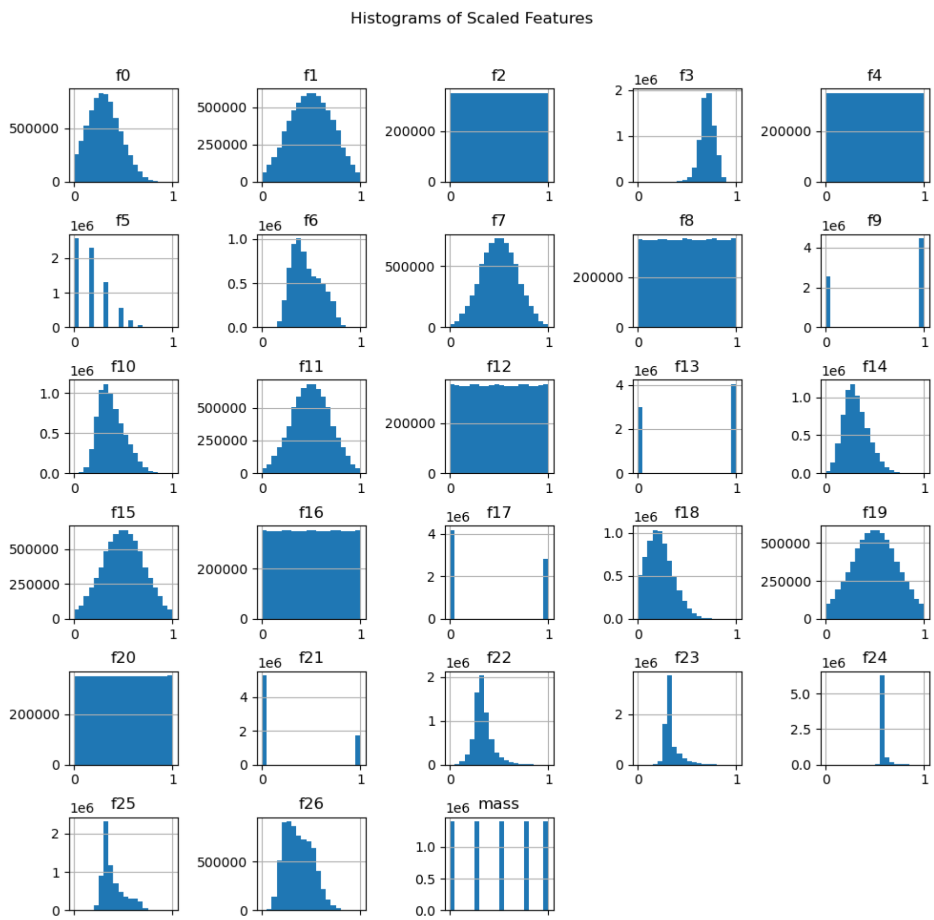

Figure 2: Histograms of Distributions for all Scaled Features

To understand the features better, the histograms of the distribution of each scaled feature is shown in Figure 2. Certain features such as “f2”, “f12”, “f16” and “f20” seem to not have very much variability, therefore these features will be further considered and perhaps removed from the analysis to minimize computational time and increase accuracy. Next, the data is split into training and testing sets which is a crucial step for model evaluation, to prevent overfitting, parameter tuning, and generalization.

2.3 NEURAL NETWORKS (NNs)

Neural networks are mathematically designed to mimic biological neurons, which are interconnected nodes organized in layers. These neurons have thresholds at which signals are allowed to continue through the neuron, essentially making these an ensemble of regressions. Each neuron receives input signals, processes them, and then generates an output signal which in some cases will serve as input for the neurons in the subsequent layers. These connections between the neurons are represented by weights, which is what determines the strength of the influence each neuron has on each other. Different threshold functions can provide different step function behavior such as the sigmoid function, which are called activation functions.

This case study utilizes a type of neural network architecture called dense neural networks, or fully connected neural networks, where each neuron in a certain layer is connected to every neuron in adjacent layers. These connections form a dense matrix structure and capture complex relationships that are learned between input features and output predictions.

3 | RESULTS

First, a sequential model was created as a baseline with three dense layers, which can be seen in Figure 3. The first two layers used sigmoid activation functions and the third layer used a linear activation function. This model was trained on 10 epochs with which performed with a validation accuracy of 49.94%, which not ideal.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 100)               2900

 dense_1 (Dense)             (None, 50)                5050

 dense_2 (Dense)             (None, 1)                 51

=================================================================
Total params: 8001 (31.25 KB)
Trainable params: 8001 (31.25 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 3: Model 1 Architecture - Sequential Dense Neural Network with 3 Dense Layers

To improve the accuracy, a second sequential model was created with a dense layer using the tanh activation function, a dropout layer, and another dense layer using the softmax activation function, which can be seen in Figure 4. This model performed with a validation accuracy of 85.11%, which is a big improvement from the previous model. The dropout layer and different activation functions appear to make a big different in the training of the model. Optimizing these parameters is crucial in training neural network models.

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (100, 100)                2900

 dropout_1 (Dropout)         (100, 100)                0

 dense_3 (Dense)             (100, 3)                  303

=================================================================
Total params: 3203 (12.51 KB)
Trainable params: 3203 (12.51 KB)
Non-trainable params: 0 (0.00 Byte)
```

Figure 4: Model 2 Architecture – Sequential Dense Neural Network with 2 Dense Layers and Dropout Layer

In Figure 5, a plot showing the training loss (blue) and validation loss (orange) across epochs during training. As can be seen, both training and validation loss decreases as the epochs increase which means the model is learning from the previous runs.

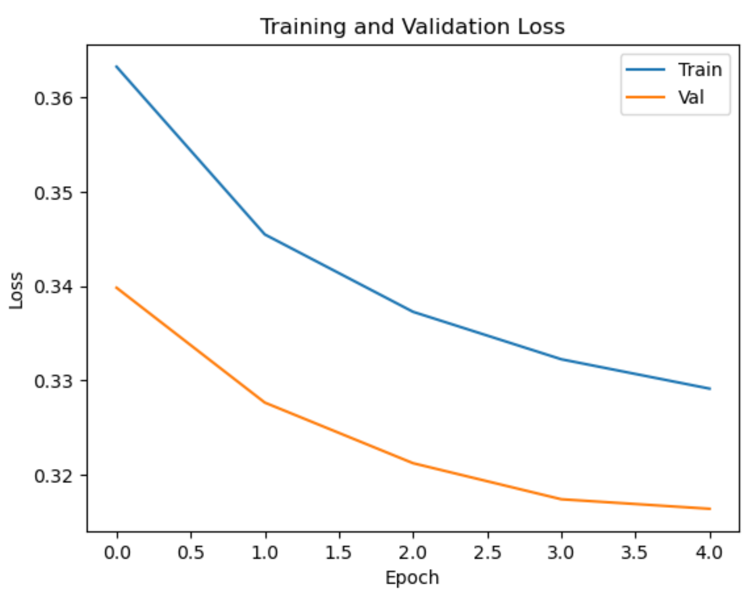

Figure 5: Training Loss (blue) & Validation Loss (orange) over Epochs during Training

To visualize the accuracy of the model, the training accuracy (blue) and validation accuracy (orange) across epochs during training is shown in Figure 6. This shows the training accuracy consistently increasing across epochs, while the validation accuracy is less consistent, it is also overall increasing.

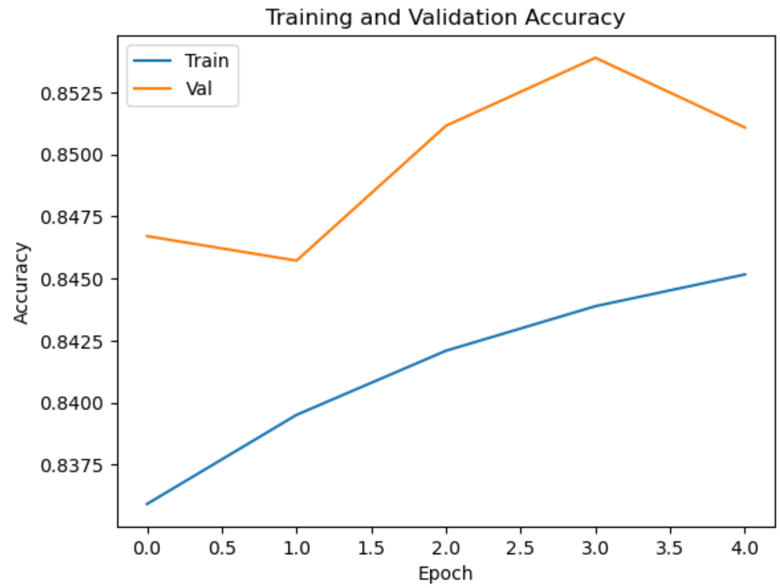

Figure 6: Training Accuracy (blue) & Validation Accuracy (orange) over Epochs during Training

In the confusion matrix shown in Figure 7, the performance of the model's classification for accurately predicting "Detection" or "Non-Detection" can be visualized.

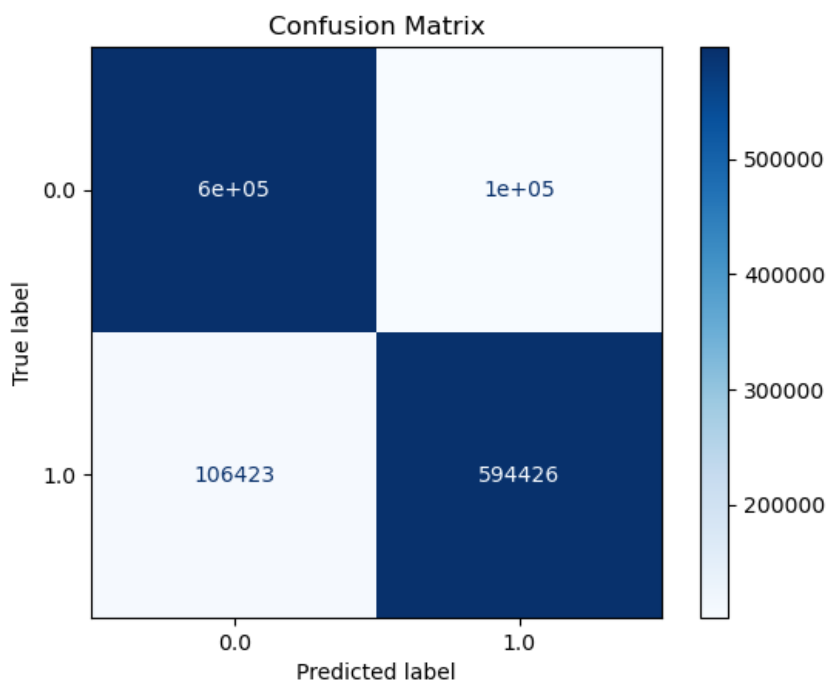

Figure 7: Confusion Matrix of Predicted Labels vs. Actual Labels

The correctly predicted “Non-Detection” labels are in the top-right quadrant while the incorrectly predicted “Non-Detection” labels are in the bottom-left quadrant. On the other hand, the correctly predicted “Detection” labels are in the top-left quadrant while the incorrectly predicted “Detection” labels are in the bottom-left quadrant.

4 | CONCLUSION

In conclusion, dense neural networks offer a powerful and flexible approach for tackling binary classification tasks. By leveraging multiple layers of interconnected neurons, these type of neural networks can efficiently learn complex patterns and relationships in data, and are especially suited for working with large datasets such as this. By utilizing activation functions such as the sigmoid function or reLU function, the dense layers allows for the modeling of non-linear relationships between input features and the target variable. In addition, optimization algorithms such as SGD or Adam are useful for hyper-parameter tuning. While these neural networks are efficient, they require careful tuning of hyperparameters, such as number of layers, neurons per layer, learning rate, and others, to achieve optimal performance. The downfall is these processes can take a lot of time and computational resources. Further hyperparameter tuning and optimization could further improve the overall accuracy of the model.

5 | CODE

Relevant code is attached in CarolinaCraus_CS6.ipynb