Classification Using Neural Networks

Mental Health Data Analysis

Paula Lozano Gonzalo

February 25, 2025

1 Introduction

This report shows the process of developing and evaluating neural network models for predicting mental health outcomes using the Kaggle Playground Series - Season 4 - Episode 11 Exploring Mental Health Data dataset. The goal was to train multiple neural network architectures, experiment with various hyperparameters, and submit predictions to the Kaggle competition.

2 Data Understanding

The dataset contains 18 features that can be used to predict depression status. After initial data exploration, I determined that all features seemed potentially useful for predicting the mental health outcome, with the possible exception of the "name" feature, which I wondered whether it would have predictive value for depression status.

3 Data Preparation

I used preprocessing methods from scikit-learn rather than Keras or TensorFlow, as these offer more robust preprocessing capabilities (Just like we said in class). The preprocessing steps were implemented in a separate `preprocess.py` file. To ensure data integrity, I verified that the file sizes remained consistent after preprocessing:

```
wc -l data/*.csv
   93801 data/preprocessed-test.csv
  140701 data/preprocessed-train.csv
   93801 data/test.csv
  140701 data/train.csv
```

4 Model Development and Evaluation

All models used mean squared error (MSE) as the loss function, R^2 Score as the evaluation metric, and Stochastic Gradient Descent (SGD) as the optimizer. The following models were developed and evaluated:

4.1 First Model

Configuration:

- Model Architecture: Sequential
- Activation Function: ReLU
- Layers: 1 hidden layer

- Neurons per Layer: 100
- Output Activation: linear
- Batch Size: 32
- Learning Rate: 0.1

Results:

- Best Epoch: 8/100 (12/100 total)
- Training R^2 Score: 0.7000
- Training Loss: 0.0444
- Validation R^2 Score: 0.6934
- Validation Loss: 0.0461
- Kaggle Score (Accuracy): 0.93823

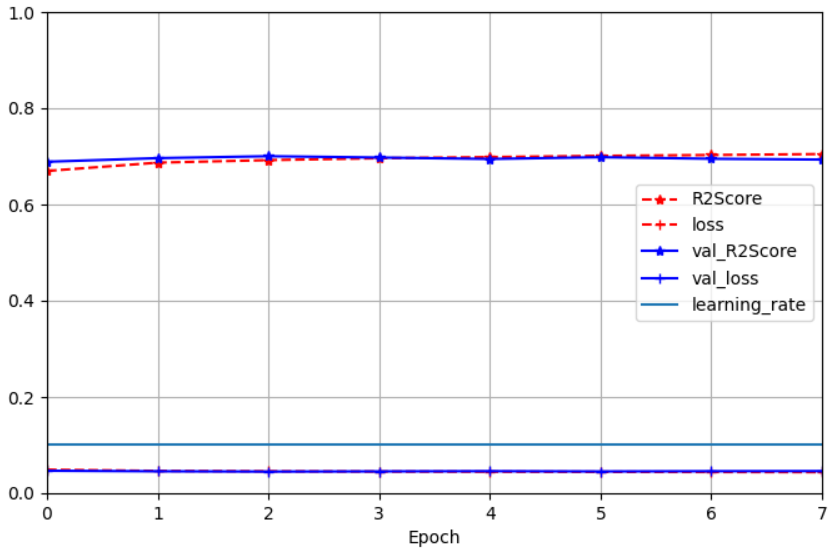

Figure 1: Learning curve for Model 1

Figure 2: Kaggle score for Model 1

Analysis: This baseline model performed surprisingly well without any optimization, achieving a Kaggle accuracy score above 0.93. The model converged quickly after only 8 epochs, indicating that the problem might be relatively straightforward for a neural network to learn.

4.2 Second Model

Configuration:

- Model Architecture: Sequential
- Hidden Layer Activation: ReLU
- Output Activation: Sigmoid
- Weight Initializer: HeNormal()
- Layers: 2 hidden layers
- Neurons per Layer: 128
- Batch Size: 32 (default)
- Learning Rate: 0.1

Results:

- Best Epoch: 9/100 (14/100 total)
- Training R^2 Score: 0.7229
- Training Loss: 0.0408
- Validation R^2 Score: 0.7044
- Validation Loss: 0.0444
- Kaggle Score (Accuracy): 0.93980

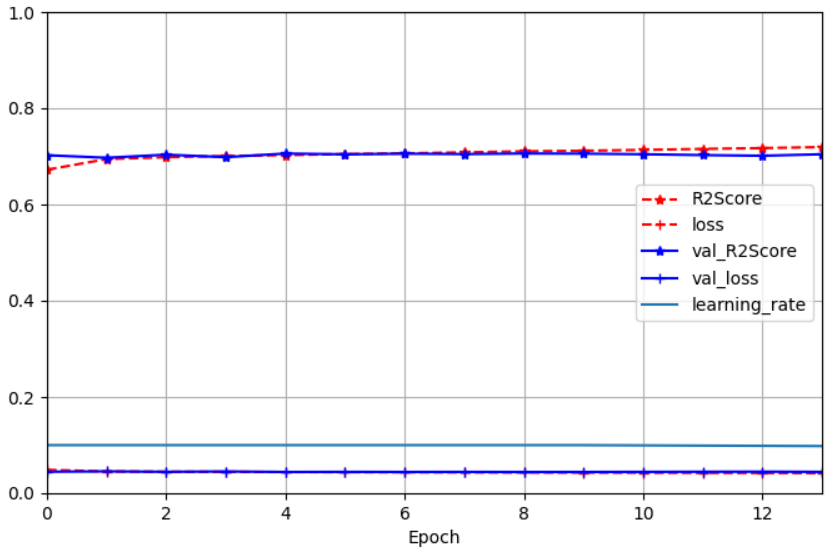

Figure 3: Learning curve for Model 2

Analysis: The second model showed a little improvement over the first one, with increases in both training and validation R^2 scores. The addition of a second hidden layer, increased neuron count, and improved weight initialization strategy likely contributed to the performance gain. The model required more epochs to converge, suggesting it was learning a more complex representation of the data.

Figure 4: Kaggle score for Model 2

4.3 Third Model (Name Feature Removed)

Configuration:

- Model Architecture: Sequential
- Hidden Layer Activation: ReLU
- Output Activation: Sigmoid
- Weight Initializer: HeNormal()
- Layers: 2 hidden layers
- Neurons per Layer: 128
- Batch Size: Initially 32, attempted 64
- Learning Rate: 0.1 (initially, with decay)

Results:

- Best Epoch: 14/100 (with batch size 32)
- Training R^2 Score: 0.7102
- Training Loss: 0.0432
- Validation R^2 Score: 0.7023
- Validation Loss: 0.0447
- Note: Encountered dimensional mismatch during prediction

Here is the error I got and couldn't figure out:

```
./neural_net_predict.py
Model: "sequential"
 Total params: 62,211 (243.02 KB)
 Trainable params: 62,209 (243.00 KB)
 Non-trainable params: 0 (0.00 B)
 Optimizer params: 2 (12.00 B)
None
Traceback (most recent call last):
  File "/Users/paulalozanogonzalo/spring2025/CS-4320/neural_network_classification/./neural_net_predict
    y_hat = model.predict(X)
            ^^^^^^^^^^^^^^^^
  File "/Users/paulalozanogonzalo/spring2025/CS-4320/.venv11/lib/python3.11/site-packages/keras/src/util
    raise e.with_traceback(filtered_tb) from None
  File "/Users/paulalozanogonzalo/spring2025/CS-4320/.venv11/lib/python3.11/site-packages/keras/src/laye
    raise ValueError(
ValueError: Exception encountered when calling Sequential.call().
Input 0 of layer "dense" is incompatible with the layer: expected axis -1 of input shape to have value 3
Arguments received by Sequential.call():
  • inputs=tf.Tensor(shape=(32, 777), dtype=float32)
  • training=False
  • mask=None
```

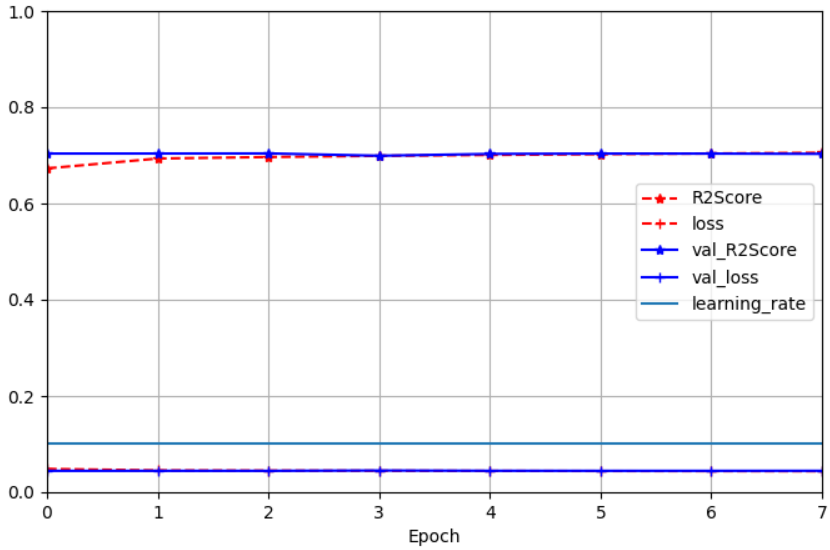

Figure 5: Learning curve for Model 3

Analysis: Removing the "name" feature had minimal impact on the model's performance metrics, suggesting this feature might not have been critical for the prediction task. However, the preprocessing change led to compatibility issues when attempting to generate predictions, possibly due to inconsistent feature dimensionality between training and test datasets.

4.4 Fourth Model (Complex Architecture)

Configuration:

- Model Architecture: Sequential
- Activation Function: SELU
- Weight Initializer: LecunNormal()
- Layers: 3 hidden layers with additional components:
  - Batch Normalization after each dense layer
  - Dropout (0.3) after each batch normalization
- Output Activation: Sigmoid
- Batch Size: 32
- Learning Rate: 0.1

Results:

- Best Epoch: 9/100
- Training R^2 Score: 0.6649
- Training Loss: 0.0496
- Validation R^2 Score: 0.7042

- Validation Loss: 0.0444
- Kaggle Score (Accuracy): 0.93937

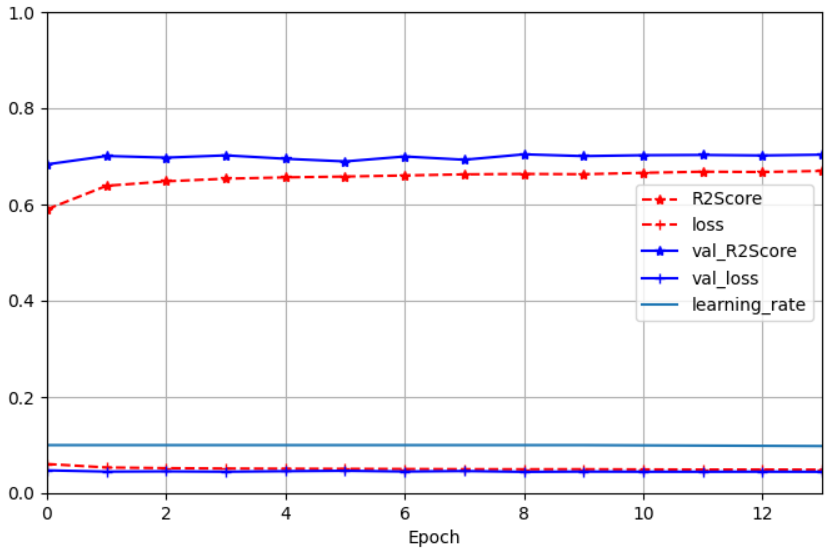

Figure 6: Learning curve for Model 4

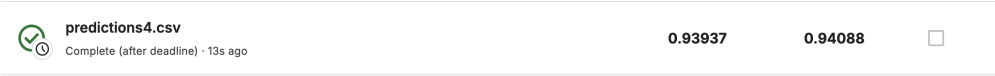

Figure 7: Kaggle score for Model 4

Analysis: Despite having lower training R^2 scores, this more complex model achieved the second-best Kaggle performance. The inclusion of regularization techniques (batch normalization and dropout) likely helped prevent overfitting, making the model more generalizable to the test data. The SELU activation function, which is self-normalizing, may have contributed to more stable training dynamics.

5 Model Comparison

Model	Architecture	Activation	Layers	Neurons	Best Epoch	Train R^2	Val R^2	Kaggle Score
1	Sequential	ReLU	1	100	8/100	0.7000	0.6934	0.93823
2	Sequential	ReLU+Sigmoid	2	128	14/100	0.7229	0.7044	0.93980
3*	Sequential	ReLU+Sigmoid	2	128	14/100	0.7102	0.7023	N/A
4	Sequential+	SELU+Sigmoid	3+BN+DO	100	9/100	0.6649	0.7042	0.93937

Table 1: Model Comparison

*Model 3 encountered prediction issues and could not be properly evaluated on Kaggle.
+BN: Batch Normalization, DO: Dropout (0.3)

6 Conclusions

The second model achieved the best performance with a Kaggle score of 0.93980, surpassing the target threshold of 0.9 accuracy (I don't know if this still holds from last assignment). Key findings from the experiments include:

1. Adding a second hidden layer and increasing the neuron count from 100 to 128 improved performance.
2. The "name" feature had minimal impact on model performance, confirming the initial hypothesis.
3. Complex regularization techniques (batch normalization and dropout) produced competitive results despite lower training metrics, highlighting the importance of generalization over training performance.
4. All models performed remarkably well with minimal tuning, suggesting the classification task may be relatively straightforward for neural networks.

For future work, I would consider:

- More extensive hyperparameter tuning, particularly exploring different learning rates and optimizers
- Addressing the dimensional mismatch issue encountered in Model 3 to enable proper evaluation

All models successfully exceeded the required 0.9 accuracy threshold on the Kaggle competition, demonstrating the effectiveness of neural networks for this classification task.