Neural Network for Titanic Survival Prediction

A Machine Learning Tutorial with Theoretical Insights and Implementation

Gopi Erla

Course: MSc Data Science with Placement

Date: 13-12-2024

Contents

1 Introduction

The Titanic disaster of 1912 remains a historically significant event and serves as a popular problem in machine learning. The dataset, provided by Kaggle's Titanic competition, includes passenger details such as age, gender, ticket class, and fare paid to predict survival outcomes [5].

The task is a classic **binary classification problem** where the target variable has two classes:

- **1 (Survived):** The passenger survived the Titanic disaster.
- **0 (Not Survived):** The passenger did not survive.

In this tutorial, we solve this problem using a **Feedforward Neural Network (FNN)** implemented in TensorFlow/Keras. Neural Networks excel in learning non-linear relationships in data and can handle complex decision boundaries [1].

2 Neural Networks: Theoretical Foundations

2.1 What is a Neural Network?

A neural network is a computational model inspired by the human brain. It consists of multiple layers of interconnected nodes called neurons:

- **Input Layer:** Accepts the input features (e.g., age, fare, gender).
- **Hidden Layers:** Perform non-linear transformations using weighted connections and activation functions [2].
- **Output Layer:** Produces the final prediction.

2.2 Mathematical Representation of a Single Neuron

A single neuron in the network computes:

$$z = \sum_{i=1}^{n} w_i x_i + b$$

where:

- w_i: Weight of the input x_i,
- b: Bias term,
- z: Linear combination of inputs.

The activation function f introduces non-linearity:

$$a = f(z)$$

2.3 Activation Functions

1. ReLU (Rectified Linear Unit) [3]:

$$f(z) = \max(0, z)$$

- Effective for hidden layers due to reduced vanishing gradient problems.

2. Sigmoid [1]:

$$f(z) = \frac{1}{1 + e^{-z}}$$

- Maps the output to the range $(0, 1)$, commonly used for binary classification.

2.4 Forward Propagation

Forward propagation calculates the output of a network layer:

$$a^{[l]} = f(W^{[l]} a^{[l-1]} + b^{[l]})$$

where:

- $W^{[l]}$: Weights of layer l,
- $b^{[l]}$: Bias of layer l,
- f: Activation function,
- $a^{[l-1]}$: Outputs of the previous layer [4].

2.5 Loss Function

The Binary Cross-Entropy Loss is used for binary classification:

$$L = -\frac{1}{m} \sum_{i=1}^{m} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$

where:

- y_i: True label,
- $\hat{y}_i$: Predicted probability.

2.6 Backpropagation and Optimization

Backpropagation calculates gradients using the chain rule. The weights are updated using optimizers such as Adam:

$$W = W - \eta \nabla L$$

where:

- η: Learning rate,
- ∇L: Gradient of the loss function.

3 Data Preprocessing

3.1 Steps

The dataset preprocessing involves:

1. Handling missing values for **Age** and **Embarked**.
2. Encoding categorical features using One-Hot Encoding.
3. Scaling numerical features (**Age**, **Fare**) using StandardScaler.

Code Snippet:

```
df['Age'].fillna(df['Age'].median(), inplace=True)
df['Embarked'].fillna(df['Embarked'].mode()[0], inplace=True)
df = pd.get_dummies(df, columns=['Sex', 'Embarked'], drop_first=True)
scaler = StandardScaler()
df[['Age', 'Fare']] = scaler.fit_transform(df[['Age', 'Fare']])
```

4 Model Implementation

The Feedforward Neural Network consists of:

- Input Layer: 64 neurons (ReLU activation),
- Dropout: 30% for regularization,
- Hidden Layer: 32 neurons (ReLU activation),
- Output Layer: 1 neuron (Sigmoid activation).

Code Snippet:

```
model = Sequential([
    Dense(64, input_dim=X_train.shape[1], activation='relu'),
    Dropout(0.3),
    Dense(32, activation='relu'),
    Dense(1, activation='sigmoid')
])
model.compile(optimizer=Adam(learning_rate=0.001), loss='binary_crossentropy', metric
```

5 Results and Evaluation

5.1 Performance Metrics

6 Conclusion

This tutorial demonstrated a practical implementation of a neural network for predicting survival on the Titanic dataset. Key takeaways:

- Neural Networks can effectively model non-linear relationships.

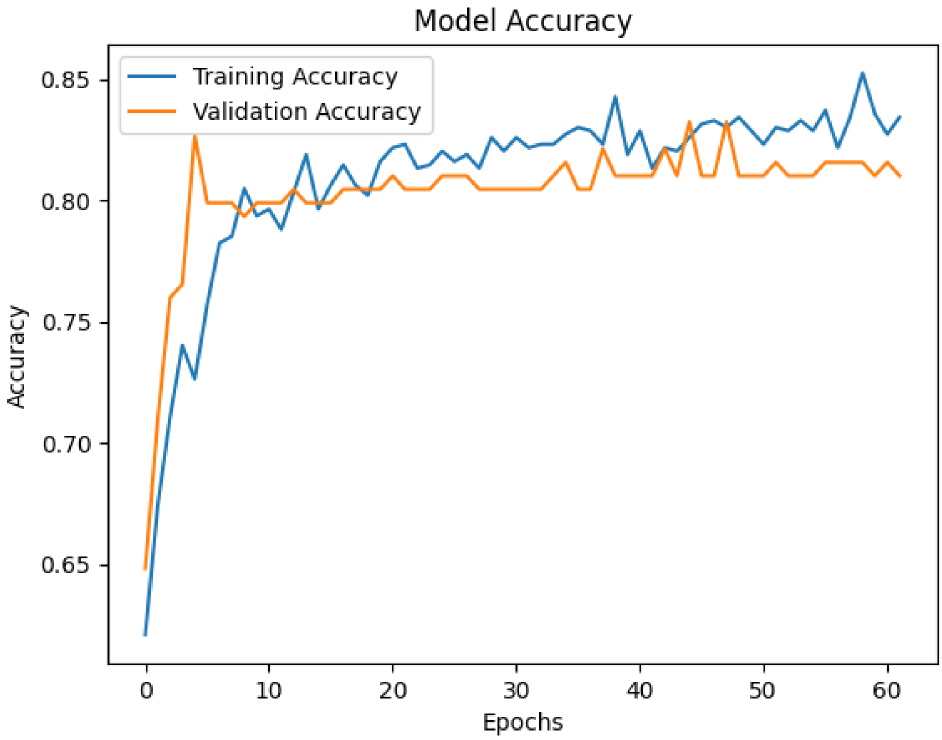

Figure 1: Training vs. Validation Accuracy for Neural Network

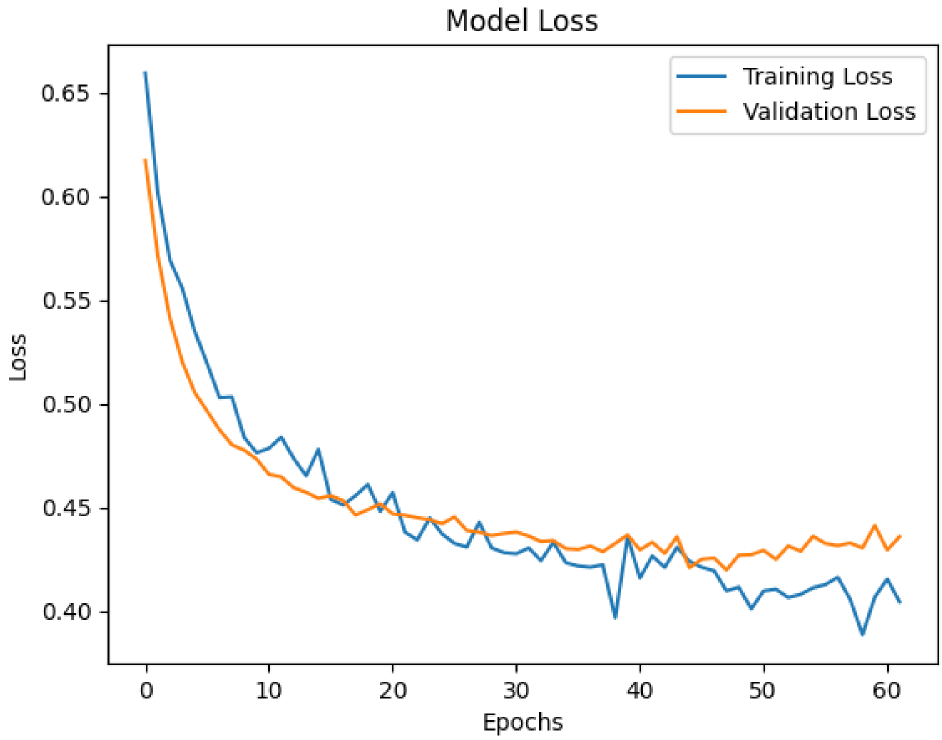

Figure 2: Training vs. Validation Loss for Neural Network

Metric	**Class 0**	**Class 1**
Precision	0.82	0.79
Recall	0.87	0.73
F1-Score	0.84	0.76

Table 1: Classification Report

- Preprocessing steps such as scaling and encoding improve accuracy.

Future Work:

- Hyperparameter tuning using GridSearchCV.
- Experimenting with deeper architectures.

7 References

References

[1] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016.

[2] Christopher Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[3] Xavier Glorot and Yoshua Bengio. Deep sparse rectifier neural networks. *AISTATS*, 2011.

[4] Andrew Ng. *Machine Learning Course Notes*. Stanford University, 2011.

[5] Kaggle. Titanic Dataset. `https://www.kaggle.com/c/titanic/data`.