**Methodology**

**Data Preprocessing**

1. **Feature Selection and Transformation**:

• All categorical variables were one-hot encoded.

• Continuous variables were standardized using a StandardScaler to ensure consistent input for the neural network.

2. **Data Split**:

• The dataset was split into training and testing sets using an 80/20 split.

• The training set was further split into a validation set during model training for tuning purposes.

**Model Architecture**

The ANN model consisted of the following layers:

1. **Input Layer**: Directly connected to the standardized input features (tenure, monthly charges, and categorical variables).

2. **Hidden Layers**:

• Two hidden layers were used:

• The first layer consisted of 64 neurons, with a ReLU activation function and an L2 regularization to prevent overfitting.

• The second hidden layer consisted of 32 neurons, also using ReLU and L2 regularization.

• **Dropout Layers** were included after each hidden layer to randomly drop 30% of the neurons during training to further prevent overfitting.

3. **Output Layer**:

• A single neuron with a sigmoid activation function to output a probability of churn (binary classification).

**Training Process**

• The model was compiled using the **Adam optimizer** with a tunable learning rate of 0.001.

• **Binary cross-entropy** was used as the loss function since the problem was a binary classification.

• The model was trained for 30 epochs with a batch size of 32.

• **Early stopping** was implemented to halt training if the validation loss did not improve for 3 consecutive epochs, restoring the best weights observed during training.

**Hyperparameter Tuning**

To further optimize model performance, hyperparameter tuning was performed using **Keras Tuner**. The following parameters were tuned:

• **Number of neurons** in the hidden layers.

• **Dropout rate** for regularization.

• **Learning rate** for the Adam optimizer.

Despite tuning, performance remained similar across different trials, indicating that the model was well-configured from the start and no significant improvements could be gained from further adjustments.

**Results and Performance**

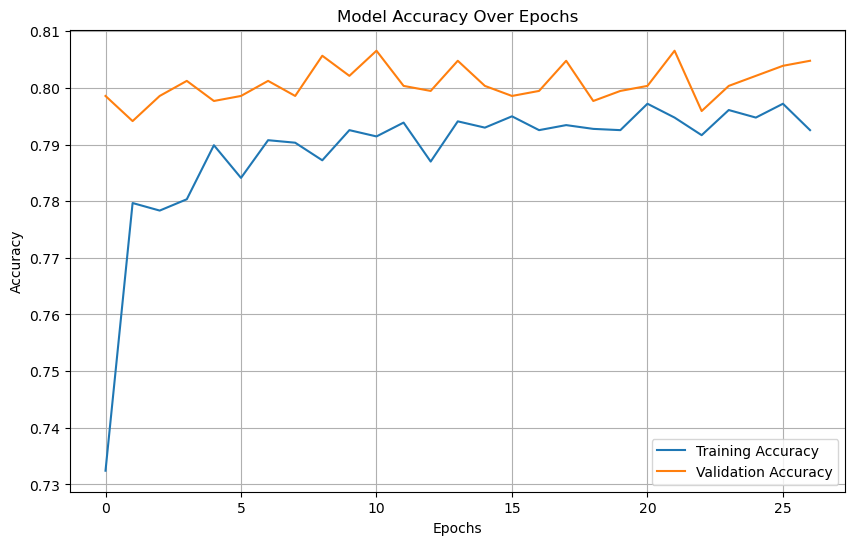
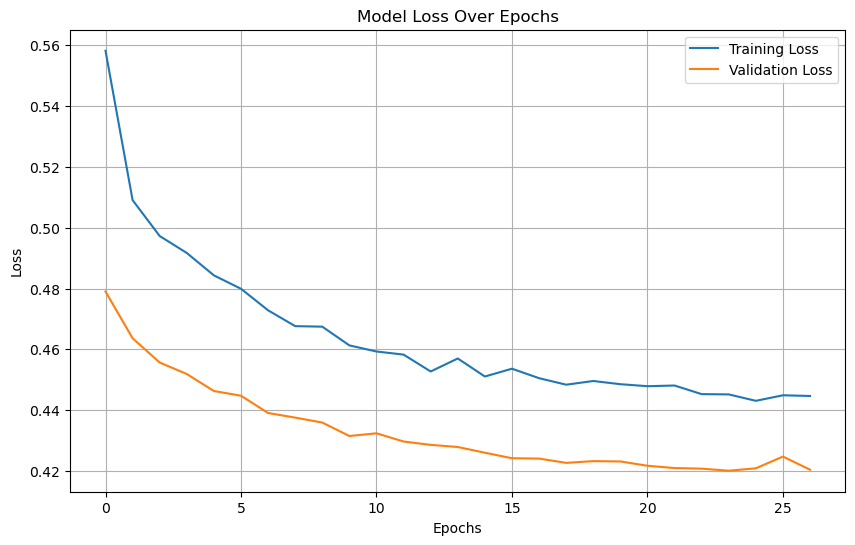
**Test Performance**

After training and tuning, the model achieved the following results on the test set:

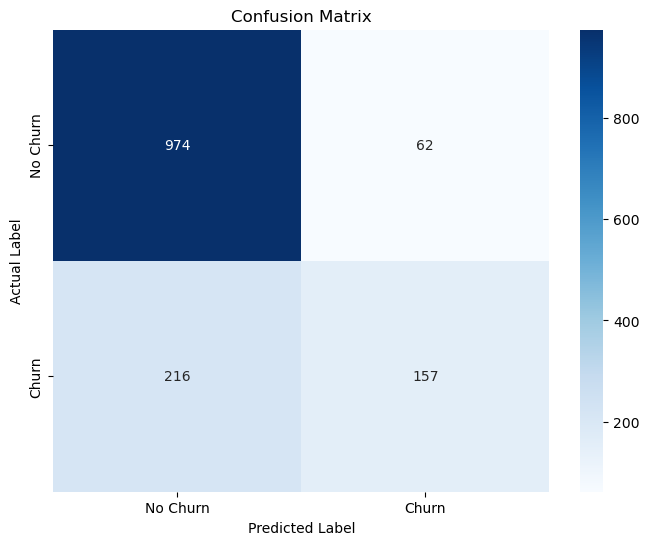
• **Test Accuracy**: 80.48%

• **Test Loss**: 0.4148

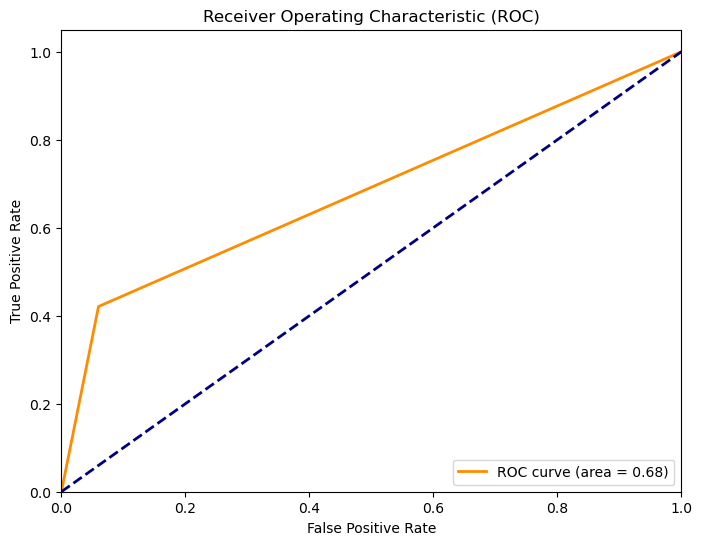
**Model Loss and Accuracy Over Epochs:**



**Confusion Matrix:**



**ROC Curve and AUC:**



The model demonstrated consistent performance across multiple runs, indicating a stable solution for the classification problem. While the test accuracy is reasonable, further optimization steps such as feature engineering or advanced ensembling could potentially improve predictive power.

**Training vs. Validation Accuracy**

During training, both the training and validation accuracy improved steadily, with early stopping preventing overfitting by halting training when further epochs did not result in improvements in the validation set. The following performance was observed:

• **Training Accuracy**: ~79-80%

• **Validation Accuracy**: ~79-80%

**Hyperparameter Tuning Results**

Hyperparameter tuning explored several configurations of the model, but the final test accuracy remained around 80%, suggesting that additional complexity in terms of more neurons or layers did not provide significant benefits. The following hyperparameters were selected as optimal:

• **First hidden layer neurons**: 64

• **Second hidden layer neurons**: 32

• **Dropout rates**: 30% for both hidden layers.

• **Learning rate**: 0.001

**Conclusion**

The ANN model developed for customer churn prediction achieved an accuracy of approximately 80.48% on the test set. Through techniques like regularization, dropout, and early stopping, the model demonstrated good generalization without overfitting.

Although hyperparameter tuning did not significantly improve performance, the model is well-optimized for the current data and task. Future improvements could focus on feature engineering, alternative machine learning models (e.g., ensemble methods), or exploring more complex network architectures.