# Neural Network Model Report

### Introduction

In this report, we detail the development and evaluation of a neural network model for classifying charity applications based on a dataset. The goal of this analysis is to create a model that accurately predicts the success of applications using various features provided in the dataset.

### Data Preprocessing

### Data Cleaning

- **Removed Non-Beneficial Columns:** The columns EIN and NAME were dropped as they were not useful for the prediction task.
- **Categorical Variable Encoding:** Categorical variables such as APPLICATION_TYPE and CLASSIFICATION were converted to numeric format using one-hot encoding. Rare categories were grouped into "Other" to reduce dimensionality and improve model performance.

### Feature Scaling

- **StandardScaler** was used to normalize feature values, ensuring that each feature contributes equally to the model training process. This step is crucial for the neural network to perform optimally.

### Model Architecture

### Defining the Neural Network

- **Input Features:** The model accepts features corresponding to the number of columns after preprocessing.
- **Hidden Layers:**
  - **First Hidden Layer:** 7 neurons with relu activation.
  - **Second Hidden Layer:** 14 neurons with relu activation.
- **Output Layer:** 1 neuron with sigmoid activation for binary classification.

```python
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
input_features = X_train_scaled.shape[1]
hidden_nodes1=7
hidden_nodes2=14

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes1, input_dim=input_features, activation='relu'))

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes2, activation='relu'))

# Output layer
nn.add(tf.keras.layers.Dense(units=1, activation="sigmoid"))

# Check the structure of the model
nn.summary()
```
✓ 0.1s Python

**Training the Model**

The model was trained using the preprocessed data with the following parameters:

- **Epochs:** 200
- **Validation Split:** 15% of the data was used for validation.

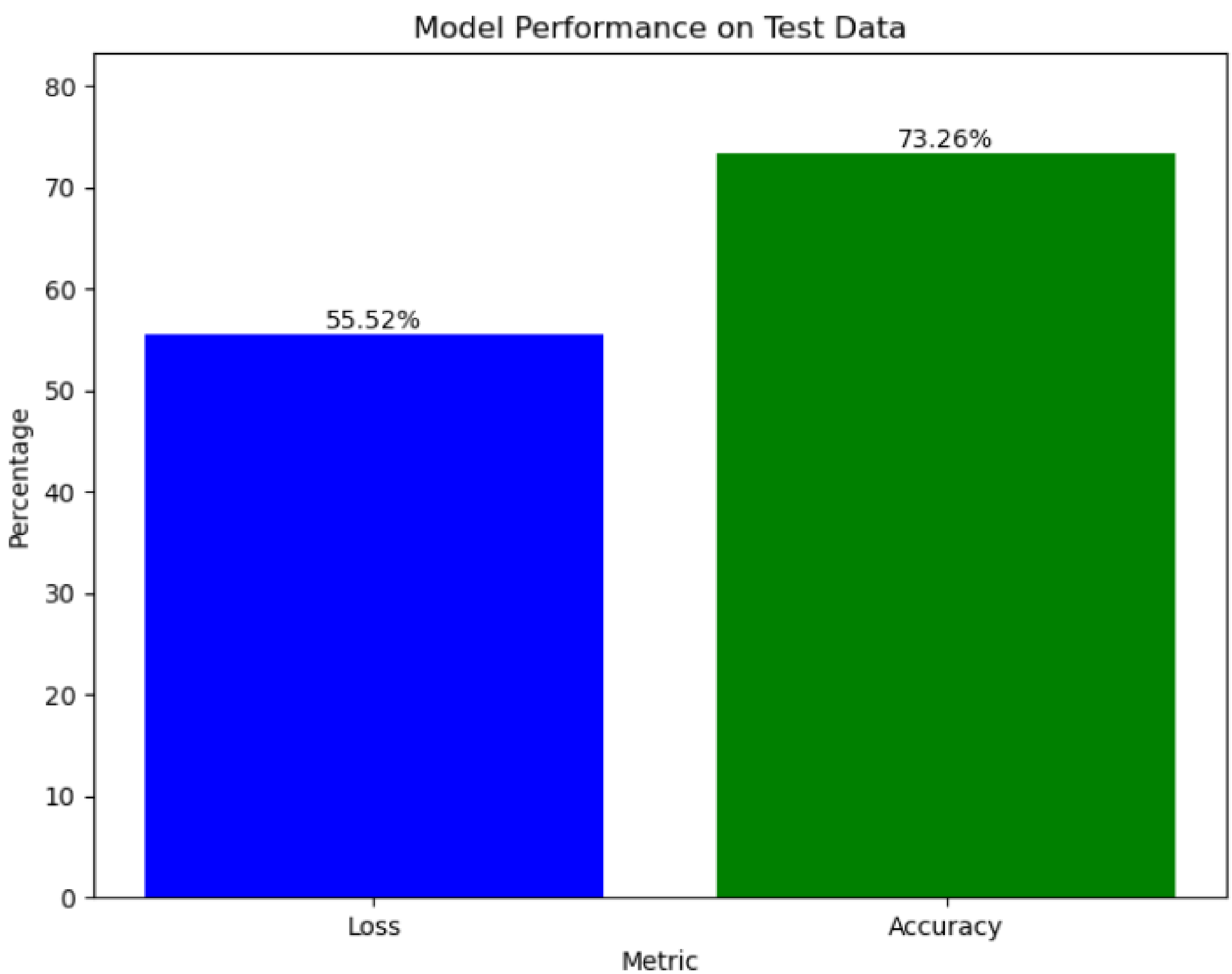
**Training History**

The above chart displays the loss and accuracy metrics for both training and validation sets over the epochs. It provides insight into how well the model is learning and generalizing to unseen data.

**Results**

**Model Performance**

- **Loss:** 55.52%
- **Accuracy:** 73.26%



**Questions Answered**

1. **What is the purpose of the model?**
   - The model aims to classify charity applications based on various features to predict their success. This helps in making data-driven decisions regarding funding.
2. **What preprocessing steps were applied to the data?**
   - Non-beneficial columns were dropped, categorical variables were encoded, and features were scaled.
3. **How was the model structured?**
   - The model consists of an input layer, two hidden layers with ReLU activation, and an output layer with sigmoid activation.
4. **What were the key metrics for model evaluation?**
   - Key metrics include loss and accuracy, with the model achieving a loss of 55.52% and an accuracy of 73.26%.
5. **What do the training and validation curves indicate?**
   - The training and validation curves help identify if the model is overfitting or underfitting. A well-balanced curve suggests good model performance.
6. **What improvements could be made to the model?**
   - Consider tuning hyperparameters, adding more hidden layers, or using different activation functions to enhance model performance.

**Summary**

The neural network model successfully classifies charity applications with an accuracy of 73.26% and a loss of 55.52% The training and validation metrics suggest that the model is reasonably well-tuned, though there is room for improvement in accuracy and loss.

**Alternative Approaches**

**Using a Different Model**

**Random Forest Classifier**

- **Why Use It?**
  - **Versatility:** Random Forests are robust to overfitting and handle a mix of categorical and numerical data effectively.
  - **Feature Importance:** They provide insights into feature importance, which can be valuable for understanding which features most influence the predictions.
  - **Performance:** Often performs well on structured data with fewer hyperparameters to tune compared to neural networks.

By comparing different models, you can choose the one that best fits the problem and provides the most reliable results.