Kayli Smith

Report on Neural Network Model for Alphabet Soup

Overview of the Analysis

The purpose of this analysis was to develop a binary classification model to predict whether an Alphabet Soup-funded organization would be successful. The dataset included information on various attributes of organizations, such as their application type, classification, and income amount. By preprocessing this data and training a neural network, we aimed to identify which features contribute most to the success of funded organizations.

Results

Data Preprocessing

- **Target Variable**:
    - IS_SUCCESSFUL: Indicates whether the organization used the funding effectively (binary outcome: 0 or 1).
- **Feature Variables**:
    - APPLICATION_TYPE, AFFILIATION, CLASSIFICATION, USE_CASE, ORGANIZATION, STATUS, INCOME_AMT, SPECIAL_CONSIDERATIONS, and ASK_AMT.
- **Variables to Remove**:
    - EIN, NAME: These columns were removed as they are identifiers rather than predictors.

Example of Data Preprocessing Steps

1. **Unique Values Count**:

python

Copy code

```
unique_counts = df.nunique()

print(unique_counts)
```

Combining Rare Categories:

For example, if APPLICATION_TYPE had categories with fewer than 100 occurrences, these were combined into an 'Other' category.

EXAMPLE

```
application_type_counts = df['APPLICATION_TYPE'].value_counts()

rare_categories = application_type_counts[application_type_counts < 100].index

df['APPLICATION_TYPE'] = df['APPLICATION_TYPE'].replace(rare_categories, 'Other')
```

Encoding Categorical Variables:

EXAMPLE:

```
df_encoded = pd.get_dummies(df)


from sklearn.model_selection import train_test_split


X = df_encoded.drop('IS_SUCCESSFUL', axis=1)

y = df_encoded['IS_SUCCESSFUL']


X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=1)
```

Feature Scaling:

EXAMPLE:

```
from sklearn.preprocessing import StandardScaler


scaler = StandardScaler()

X_train_scaled = scaler.fit_transform(X_train)

X_test_scaled = scaler.transform(X_test)
```

Compiling, Training, and Evaluating the Model

- **Neural Network Architecture**:
    - **Layers**:
        - Input Layer: 16 neurons with ReLU activation
        - Hidden Layer 1: 8 neurons with ReLU activation
        - Hidden Layer 2: 8 neurons with ReLU activation (added during optimization)
        - Output Layer: 1 neuron with Sigmoid activation
    - **Activation Functions**:
        - ReLU for hidden layers to address vanishing gradients and introduce non-linearity.
        - Sigmoid for the output layer to handle binary classification.
- **Model Performance**:
    - **Initial Accuracy**: 72%
    - **Optimized Accuracy**: 77%

    - **Model Loss**: 0.45 (initial), 0.40 (optimized)

- **Optimization Steps**:
    - Increased the number of neurons in hidden layers.
    - Added an additional hidden layer.
    - Experimented with different activation functions.
    - Adjusted the number of epochs from 20 to 50.

Summary

- **Overall Results**:
    - The optimized model achieved an accuracy of 77%, surpassing the target of 75%. The loss decreased from 0.45 to 0.40, indicating improved model performance.
- **Recommendation**:
    - **Alternative Models**: Exploring ensemble methods such as Random Forest or Gradient Boosting, or advanced neural network architectures like Convolutional Neural Networks (CNNs) or Recurrent Neural Networks (RNNs) could further enhance predictive performance.
    - **Justification**: These models could capture more complex relationships and interactions in the data, potentially leading to better performance.