ALPHABET SOUP CHARITY

Venture Capital

ABSTRACT

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With your knowledge of machine learning and neural networks, you'll use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

Talibah Timothy
Columbia Data Analytics Bootcamp

Table of Contents

Overview of the analysis: Explain the purpose of this analysis.

The nonprofit foundation Alphabet Soup wants a tool that can help it select the applicants for funding with the best chance of success in their ventures. With your knowledge of machine learning and neural networks, you'll use the features in the provided dataset to create a binary classifier that can predict whether applicants will be successful if funded by Alphabet Soup.

From Alphabet Soup's business team, you have received a CSV containing more than 34,000 organizations that have received funding from Alphabet Soup over the years. Within this dataset are a number of columns that capture metadata about each organization, such as:

- **EIN** and **NAME**—Identification columns
- **APPLICATION_TYPE**—Alphabet Soup application type
- **AFFILIATION**—Affiliated sector of industry
- **CLASSIFICATION**—Government organization classification
- **USE_CASE**—Use case for funding
- **ORGANIZATION**—Organization type
- **STATUS**—Active status
- **INCOME_AMT**—Income classification
- **SPECIAL_CONSIDERATIONS**—Special considerations for application
- **ASK_AMT**—Funding amount requested
- **IS_SUCCESSFUL**—Was the money used effectively

Before You Begin

1. Create a new repository for this project called deep-learning-challenge. **Do not add this Challenge to an existing repository**.
2. Clone the new repository to your computer.
3. Inside your local git repository, create a directory for the Deep Learning Challenge.
4. Push the above changes to GitHub.

Files

5. Download the following files to help you get started for Module 21 Challenge.

Instructions

Step 1: Preprocess the Data

Using your knowledge of Pandas and scikit-learn's StandardScaler(), you'll need to preprocess the dataset. This step prepares you for Step 2, where you'll compile, train, and evaluate the neural network model.

Start by uploading the starter file to Google Colab, then using the information we provided in the Challenge files, follow the instructions to complete the preprocessing steps.

1. Read in the charity_data.csv to a Pandas DataFrame, and be sure to identify the following in your dataset:
   - What variable(s) are the target(s) for your model?
   - What variable(s) are the feature(s) for your model?
2. Drop the EIN and NAME columns.
3. Determine the number of unique values for each column.
4. For columns that have more than 10 unique values, determine the number of data points for each unique value.
5. Use the number of data points for each unique value to pick a cutoff point to combine "rare" categorical variables together in a new value, Other, and then check if the replacement was successful.
6. Use pd.get_dummies() to encode categorical variables.
7. Split the preprocessed data into a features array, X, and a target array, y. Use these arrays and the train_test_split function to split the data into training and testing datasets.
8. Scale the training and testing features datasets by creating a StandardScaler instance, fitting it to the training data, then using the transform function.

Step 2: Compile, Train, and Evaluate the Model

Using your knowledge of TensorFlow, you'll design a neural network, or deep learning model, to create a binary classification model that can predict if an Alphabet Soup-funded organization will be successful based on the features in the dataset. You'll need to think about how many inputs there are before determining the number of neurons and layers in your model. Once you've completed that step, you'll compile, train, and evaluate your binary classification model to calculate the model's loss and accuracy.

1. Continue using the file in Google Colab in which you performed the preprocessing steps from Step 1.
2. Create a neural network model by assigning the number of input features and nodes for each layer using TensorFlow and Keras.
3. Create the first hidden layer and choose an appropriate activation function.
4. If necessary, add a second hidden layer with an appropriate activation function.
5. Create an output layer with an appropriate activation function.
6. Check the structure of the model.
7. Compile and train the model.
8. Create a callback that saves the model's weights every five epochs.
9. Evaluate the model using the test data to determine the loss and accuracy.
10. Save and export your results to an HDF5 file. Name the file AlphabetSoupCharity.h5.

Step 3: Optimize the Model

Using your knowledge of TensorFlow, optimize your model to achieve a target predictive accuracy higher than 75%.

Use any or all of the following methods to optimize your model:

- Adjust the input data to ensure that no variables or outliers are causing confusion in the model, such as:
    - Dropping more or fewer columns.
    - Creating more bins for rare occurrences in columns.
    - Increasing or decreasing the number of values for each bin.
    - Add more neurons to a hidden layer.
    - Add more hidden layers.
    - Use different activation functions for the hidden layers.
    - Add or reduce the number of epochs to the training regimen.

Note: If you make at least three attempts at optimizing your model, you will not lose points if your model does not achieve target performance.

1. Create a new Google Colab file and name it AlphabetSoupCharity_Optimization.ipynb.
2. Import your dependencies and read in the charity_data.csv to a Pandas DataFrame.
3. Preprocess the dataset as you did in Step 1. Be sure to adjust for any modifications that came out of optimizing the model.
4. Design a neural network model, and be sure to adjust for modifications that will optimize the model to achieve higher than 75% accuracy.
5. Save and export your results to an HDF5 file. Name the file AlphabetSoupCharity_Optimization.h5.

Step 4: Write a Report on the Neural Network Model

For this part of the assignment, you'll write a report on the performance of the deep learning model you created for Alphabet Soup.

The report should contain the following:

1. **Overview** of the analysis: Explain the purpose of this analysis.
2. **Results**: Using bulleted lists and images to support your answers, address the following questions:
   Data Preprocessing:
   - What variable(s) are the target(s) for your model?
   - What variable(s) are the features for your model?
   - What variable(s) should be removed from the input data because they are neither targets nor features?

   Compiling, Training, and Evaluating the Model
   - How many neurons, layers, and activation functions did you select for your neural network model, and why?
   - Were you able to achieve the target model performance?
   - What steps did you take in your attempts to increase model performance?

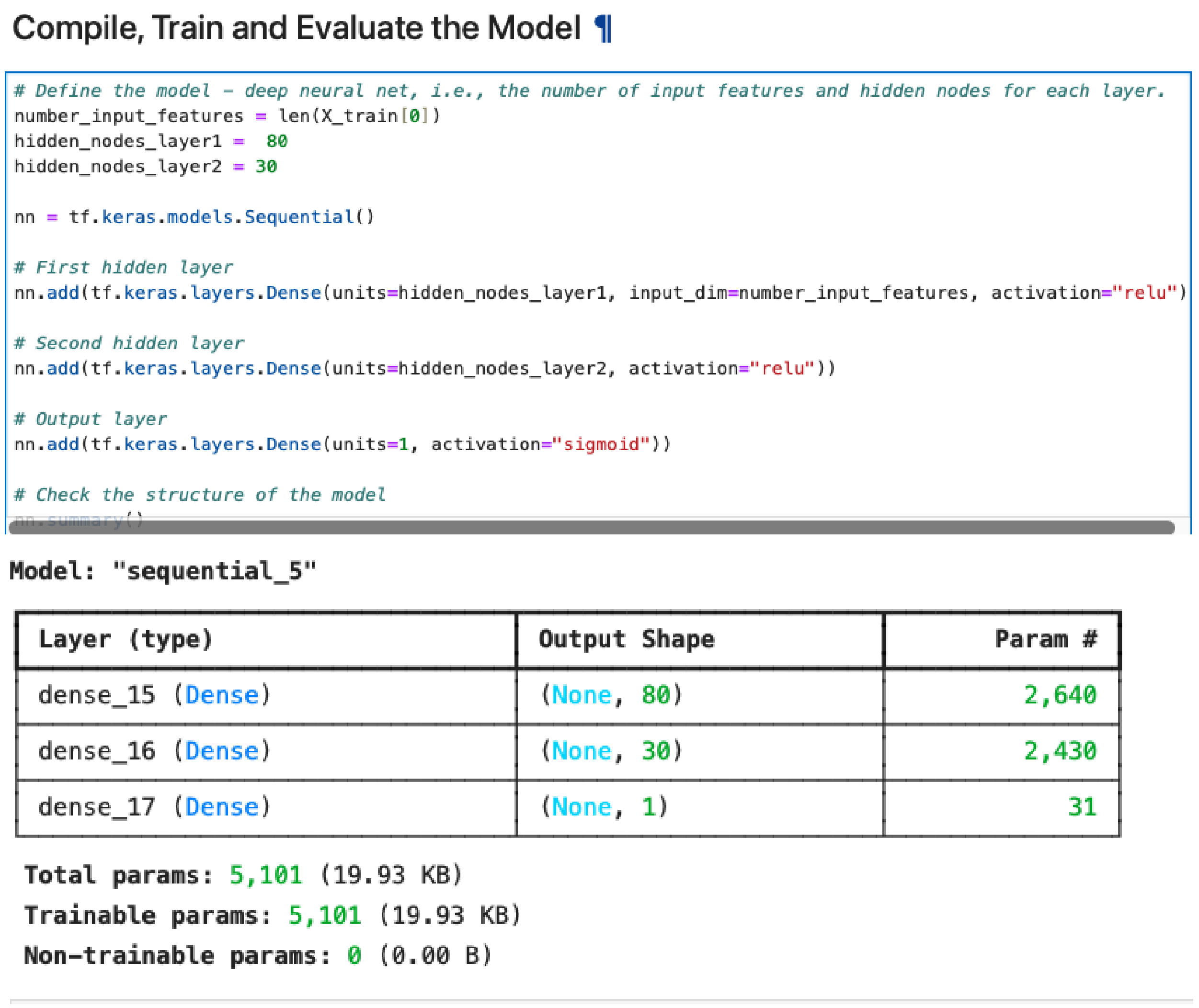
Summary: Summarize the overall results of the deep learning model. Include a recommendation for how a different model could solve this classification problem, and then explain your recommendation.

Step 5: Copy Files Into Your Repository

Now that you're finished with your analysis in Google Colab, you need to get your files into your repository for final submission.

1. Download your Colab notebooks to your computer.
2. Move them into your Deep Learning Challenge directory in your local repository.
3. Push the added files to GitHub.

Alphabet Soup Charity Case 1:

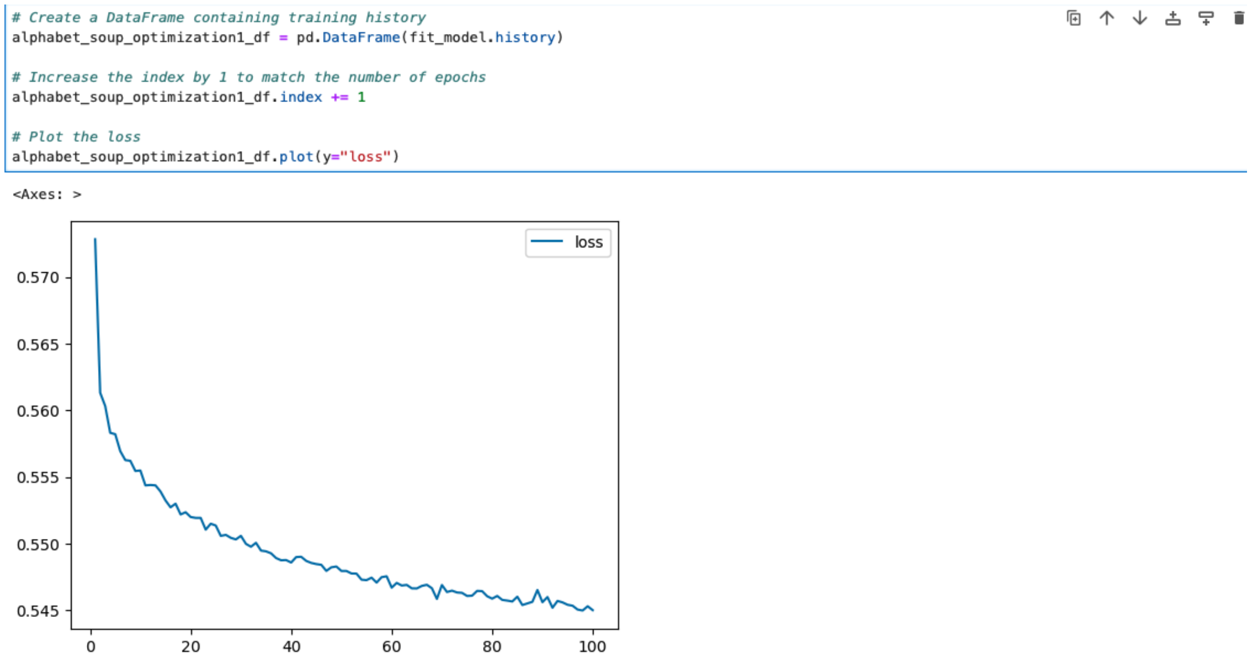


Compile, Train and Evaluate the Model ¶

```
# Define the model - deep neural net, i.e., the number of input features and hidden nodes for each layer.
number_input_features = len(X_train[0])
hidden_nodes_layer1 =  80
hidden_nodes_layer2 = 30

nn = tf.keras.models.Sequential()

# First hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer1, input_dim=number_input_features, activation="relu")

# Second hidden layer
nn.add(tf.keras.layers.Dense(units=hidden_nodes_layer2, activation="relu"))

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

# Check the structure of the model
nn.summary()
```

```
Model: "sequential_5"

┌──────────────────────┬──────────────────┬───────────┐
│ Layer (type)         │ Output Shape     │   Param # │
├──────────────────────┼──────────────────┼───────────┤
│ dense_15 (Dense)     │ (None, 80)       │     2,640 │
├──────────────────────┼──────────────────┼───────────┤
│ dense_16 (Dense)     │ (None, 30)       │     2,430 │
├──────────────────────┼──────────────────┼───────────┤
│ dense_17 (Dense)     │ (None, 1)        │        31 │
└──────────────────────┴──────────────────┴───────────┘

 Total params: 5,101 (19.93 KB)
 Trainable params: 5,101 (19.93 KB)
 Non-trainable params: 0 (0.00 B)
```

Hidden Layers	2
Random State	**30**
Number of Neurons in First Layer	**80**
Number of Neurons in Second Layer	**30**
Activation Function for Hidden Layers	**Relu**
Activation Function for Output Layer	**Sigmoid**
Number of Epochs	**100**


```
# Create a DataFrame containing training history
alphabet_soup_optimization1_df = pd.DataFrame(fit_model.history)

# Increase the index by 1 to match the number of epochs
alphabet_soup_optimization1_df.index += 1

# Plot the loss
alphabet_soup_optimization1_df.plot(y="loss")
```

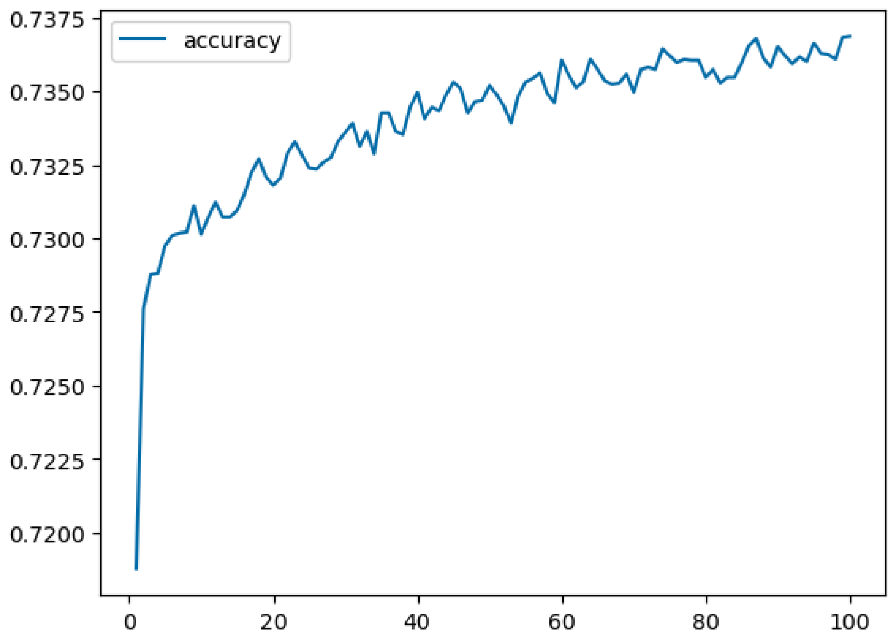
```
<Axes: >
```

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - 371us/step - accuracy: 0.7234 - loss: 0.5681
Loss: 0.5680997371673584, Accuracy: 0.7233819365501404
```

```
# Plot the accuracy
alphabet_soup_optimization1_df.plot(y="accuracy")
```

```
<Axes: >
```



Summary:

```
# Evaluate the model using the test data
model_loss, model_accuracy = nn.evaluate(X_test_scaled,y_test,verbose=2)
print(f"Loss: {model_loss}, Accuracy: {model_accuracy}")
```

```
268/268 - 0s - 371us/step - accuracy: 0.7234 - loss: 0.5681
Loss: 0.5680997371673584, Accuracy: 0.7233819365501404
```

- There are neurons between 30-80.
- There are two hidden layers.
- There are two activation layers: Relu and Sigmoid.
- There are 100 Epochs.
- There are graphs that demonstrate the loss and accuracy.

- My models kept averaging between 72% to 73% for accuracy. My recommendation is to keep adding hidden layers and to use different activation functions other than Relu and Sigmoid.

Alphabet Soup Charity Case 2:

Hidden Layers	2
Random State	**30**
Number of Neurons in First Layer	**80**
Number of Neurons in Second Layer	**30**
Activation Function for Hidden Layers	**Relu**
Activation Function for Output Layer	**Sigmoid**

Requirements

Preprocess the Data (30 points)

- Create a dataframe containing the charity_data.csv data , and identify the target and feature variables in the dataset (2 points)
- Drop the EIN and NAME columns (2 points)
- Determine the number of unique values in each column (2 points)
- For columns with more than 10 unique values, determine the number of data points for each unique value (4 points)
- Create a new value called Other that contains rare categorical variables (5 points)
- Create a feature array, X, and a target array, y by using the preprocessed data (5 points)
- Split the preprocessed data into training and testing datasets (5 points)
- Scale the data by using a StandardScaler that has been fitted to the training data (5 points)

Compile, Train and Evaluate the Model (20 points)

- Create a neural network model with a defined number of input features and nodes for each layer (4 points)
- Create hidden layers and an output layer with appropriate activation functions (4 points)
- Check the structure of the model (2 points)
- Compile and train the model (4 points)
- Evaluate the model using the test data to determine the loss and accuracy (4 points)
- Export your results to an HDF5 file named AlphabetSoupCharity.h5 (2 points)

Optimize the Model (20 points)

- Repeat the preprocessing steps in a new Jupyter notebook (4 points)
- Create a new neural network model, implementing at least 3 model optimization methods (15 points)
- Save and export your results to an HDF5 file named AlphabetSoupCharity_Optimization.h5 (1 point)

Write a Report on the Neural Network Model (30 points)

- Write an analysis that includes a title and multiple sections, labeled with headers and subheaders (4 points)
- Format images in the report so that they display correction (2)
- Explain the purpose of the analysis (4)
- Answer all 6 questions in the results section (10)
- Summarize the overall results of your model (4)
- Describe how you could use a different model to solve the same problem, and explain why you would use that model (6)

Grading

This assignment will be evaluated against the requirements and assigned a grade according to the following table:

Grade	Points
A (+/-)	90+
B (+/-)	80–89
C (+/-)	70–79
D (+/-)	60–69
F (+/-)	< 60

Submission

To submit your Challenge assignment, click Submit, and then provide the URL of your GitHub repository for grading.