

A decorative graphic on the left side of the slide, consisting of white lines and circles on a dark teal background, resembling a circuit board or a stylized tree structure.

MODULE 21 CHALLENGE

CARLOS DE LA ROSA

HOME WORK 21

What is the problem?

- Which Applicants should get funded

Who has this problem?

- Alphabet Soup Foundation

Why should this problem be solved?



- Who has the best success with funding

How will I know this problem has been solved?

- Using a NN Machine Learning Model



BACKGROUND INFORMATION

- Where did you get the data ?
 - Explore CSV file containing 34k organizations that received funding over the years
- 
- 

WORKABLE SOLUTIONS

NN ML model 1

- Drop the NAME and EIN column
- Bin the Application type and Classification Column
- Perform baseline models : Linear Regression , Random Forest , Extreme GB

Compile, Train, Evaluate the NN ML model

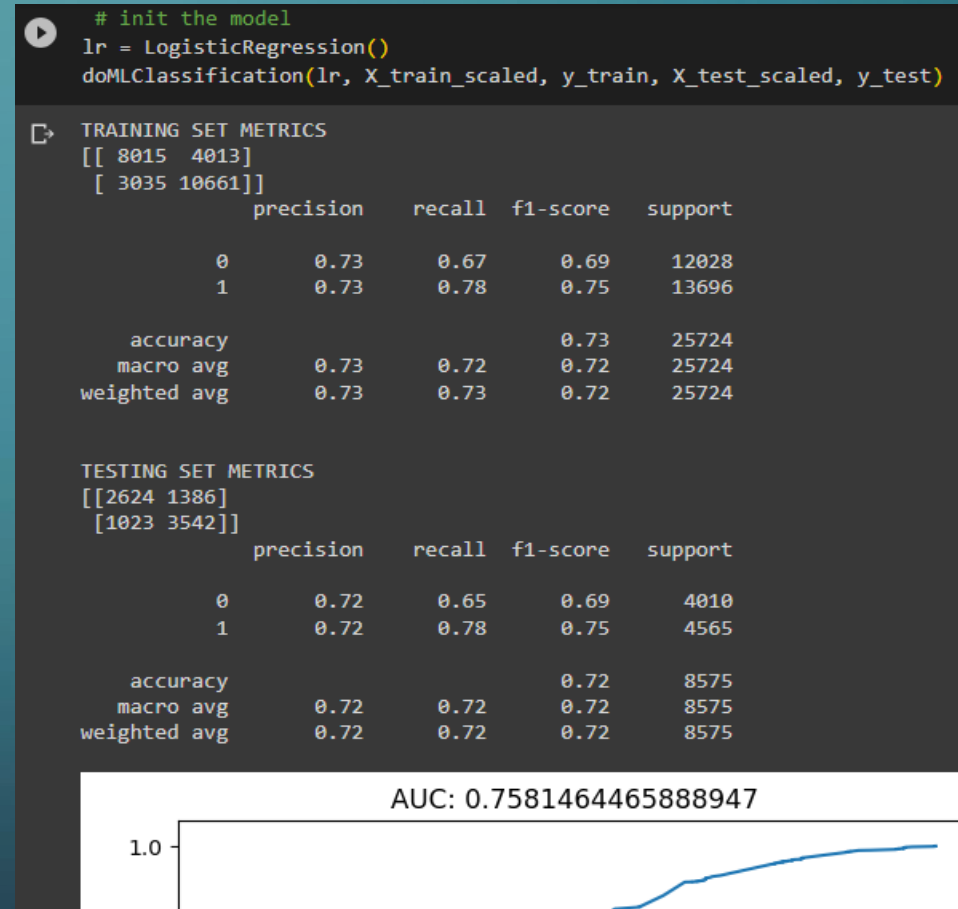
- Two hidden layers with the first layer set to 8 neurons and second layer with 5 neurons. Epoch set to 100
- Accuracy of 0.7233819365501404
- Prediction values established

Compile more models for Optimization

- Using NN ML 1 as a baseline hyper tune three more models
- Look for improvement on Area under curve , recall , F1-Score , model Accuracy
- Further exploration opportunities

THE BASELINE MODELS

- Linear regression
- Overfit, maybe, but not significant
- Accuracy 72%
- Recall ok at .78
- F1-score .75
- AUC .7581464465888947



THE BASELINE MODEL

- Random Forest
- Overfit identified
- Accuracy 71%

```
# Random forest slightly overfits and not as good as the LR
rf = RandomForestClassifier(random_state=42)
doMLClassification(rf, X_train_scaled, y_train, X_test_scaled, y_test)
```

TRAINING SET METRICS

```
[[ 9592 2436]
 [ 2246 11450]]
```

	precision	recall	f1-score	support
0	0.81	0.80	0.80	12028
1	0.82	0.84	0.83	13696
accuracy			0.82	25724
macro avg	0.82	0.82	0.82	25724
weighted avg	0.82	0.82	0.82	25724

TESTING SET METRICS

```
[[2708 1302]
 [1198 3367]]
```

	precision	recall	f1-score	support
0	0.69	0.68	0.68	4010
1	0.72	0.74	0.73	4565
accuracy			0.71	8575
macro avg	0.71	0.71	0.71	8575
weighted avg	0.71	0.71	0.71	8575

AUC: 0.7589701540234846

THE BASELINE MODEL

- Extreme GB slight overfit
- Accuracy 72%
- Recall at .75 on testing set
- F1-score lower on testing set
- AUC .7798879854034138

```
#XGB Best AUC ,  
xgb = XGBClassifier(random_state=42)  
doMLClassification(xgb, X_train_scaled, y_train, X_test_scaled, y_test)
```

TRAINING SET METRICS

```
[[ 8515  3513]
```

```
 [ 2638 11058]]
```

	precision	recall	f1-score	support
0	0.76	0.71	0.73	12028
1	0.76	0.81	0.78	13696
accuracy			0.76	25724
macro avg	0.76	0.76	0.76	25724
weighted avg	0.76	0.76	0.76	25724

TESTING SET METRICS

```
[[2661 1349]
```

```
 [1024 3541]]
```

	precision	recall	f1-score	support
0	0.72	0.66	0.69	4010
1	0.72	0.78	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7798879854034138

TESTING THE BASELINE NEURAL NETWORK MODEL 1

- Parameters of the NN Model 1
 - The NN Model 1 used two hidden layers with the 1st layer using 8 neurons and the 2nd layer using 5 neurons. The activation function for the neurons was Relu, due to its faster learning and simplified output. The output layer used sigmoid function because of its usefulness at predicting probability. In this problem it will help predict which applicants will be successful if funded by Alphabet Soup.

THE BASELINE NN MODEL 1

- Recall aka sensitivity , .77
- No overfit identified
- Accuracy 72%
- Recall at .78 on testing set
- F1-score is not a perfect balance
- AUC .7794029165858628

```
804/804 [=====] - 1s 1ms/step
268/268 [=====] - 0s 1ms/step
268/268 [=====] - 0s 1ms/step
TRAINING SET METRICS
[[ 8215  3813]
 [ 2969 10727]]
```

	precision	recall	f1-score	support
0	0.73	0.68	0.71	12028
1	0.74	0.78	0.76	13696
accuracy			0.74	25724
macro avg	0.74	0.73	0.73	25724
weighted avg	0.74	0.74	0.74	25724

```
TESTING SET METRICS
[[2671 1339]
 [1033 3532]]
```

	precision	recall	f1-score	support
0	0.72	0.67	0.69	4010
1	0.73	0.77	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7794029165858628

THE BASELINE NN MODEL 1

- Recall aka sensitivity , .77
- No overfit identified
- Accuracy 72%
- Recall at .78 on testing set
- F1-score is not a perfect balance
- AUC .7794029165858628

```
804/804 [=====] - 1s 1ms/step
268/268 [=====] - 0s 1ms/step
268/268 [=====] - 0s 1ms/step
TRAINING SET METRICS
[[ 8215  3813]
 [ 2969 10727]]
```

	precision	recall	f1-score	support
0	0.73	0.68	0.71	12028
1	0.74	0.78	0.76	13696
accuracy			0.74	25724
macro avg	0.74	0.73	0.73	25724
weighted avg	0.74	0.74	0.74	25724

```
TESTING SET METRICS
[[2671 1339]
 [1033 3532]]
```

	precision	recall	f1-score	support
0	0.72	0.67	0.69	4010
1	0.73	0.77	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7794029165858628

OPTIMIZED MODELS

- Use the data from your testing to redesign with...
 - Feature engineering
 - Bin'ed the two features 'NAME' and 'CLASSIFICATION'
- Rerun the baseline models
- Hyper tune the neural network model in three runs of the model

OPTIMIZED MODELS DIFF. FEATURE

Linear Regression

```
# init the model
lr = LogisticRegression()
doMLClassification(lr, X_train_scaled, y_train, X_test_scaled, y_test)
```

TRAINING SET METRICS

```
[[ 8160 3868]
 [ 3211 10485]]
```

	precision	recall	f1-score	support
0	0.72	0.68	0.70	12028
1	0.73	0.77	0.75	13696
accuracy			0.72	25724
macro avg	0.72	0.72	0.72	25724
weighted avg	0.72	0.72	0.72	25724

TESTING SET METRICS

```
[[2659 1351]
 [1085 3480]]
```

	precision	recall	f1-score	support
0	0.71	0.66	0.69	4010
1	0.72	0.76	0.74	4565
accuracy			0.72	8575
macro avg	0.72	0.71	0.71	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7705262036584334

Random Forest

```
# Random forest slightly overfits and not as good as the LR
rf = RandomForestClassifier(random_state=42)
doMLClassification(rf, X_train_scaled, y_train, X_test_scaled, y_test)
```

TRAINING SET METRICS

```
[[ 9605 2423]
 [ 2245 11451]]
```

	precision	recall	f1-score	support
0	0.81	0.80	0.80	12028
1	0.83	0.84	0.83	13696
accuracy			0.82	25724
macro avg	0.82	0.82	0.82	25724
weighted avg	0.82	0.82	0.82	25724

TESTING SET METRICS

```
[[2708 1302]
 [1195 3370]]
```

	precision	recall	f1-score	support
0	0.69	0.68	0.68	4010
1	0.72	0.74	0.73	4565
accuracy			0.71	8575
macro avg	0.71	0.71	0.71	8575
weighted avg	0.71	0.71	0.71	8575

AUC: 0.764477415442773

Extreme GB

```
#LGBM Best AUC ,
xgb = XGBClassifier(random_state=42)
doMLClassification(xgb, X_train_scaled, y_train, X_test_scaled, y_test)
```

TRAINING SET METRICS

```
[[ 8422 3606]
 [ 2616 11080]]
```

	precision	recall	f1-score	support
0	0.76	0.70	0.73	12028
1	0.75	0.81	0.78	13696
accuracy			0.76	25724
macro avg	0.76	0.75	0.76	25724
weighted avg	0.76	0.76	0.76	25724

TESTING SET METRICS

```
[[2634 1376]
 [1025 3540]]
```

	precision	recall	f1-score	support
0	0.72	0.66	0.69	4010
1	0.72	0.78	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.788856555216559

- Random forest didn't do any better overfit again
- Liner still performed better
- Xgb better AUC than Linear very similar in metrics

OPTIMIZED MODELS

2nd NN Model

```
804/804 [=====] - 1s 1ms/step
268/268 [=====] - 0s 1ms/step
268/268 [=====] - 0s 1ms/step
TRAINING SET METRICS
[[ 7985  4043]
 [ 2666 11030]]
```

	precision	recall	f1-score	support
0	0.75	0.66	0.70	12028
1	0.73	0.81	0.77	13696
accuracy			0.74	25724
macro avg	0.74	0.73	0.74	25724
weighted avg	0.74	0.74	0.74	25724

```
TESTING SET METRICS
[[2576 1434]
 [ 946 3619]]
```

	precision	recall	f1-score	support
0	0.73	0.64	0.68	4010
1	0.72	0.79	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7854236806668979

- 2nd NN Model 3 hidden layers
- Neurons , lyr 1= 12, lyr2=8, lyr3=5
- Epoch = 150
- Activation function relu, output function =sigmoid

3rd NN Model

```
804/804 [=====] - 2s 2ms/step
268/268 [=====] - 1s 2ms/step
268/268 [=====] - 1s 2ms/step
TRAINING SET METRICS
[[ 7974  4054]
 [ 2627 11069]]
```

	precision	recall	f1-score	support
0	0.75	0.66	0.70	12028
1	0.73	0.81	0.77	13696
accuracy			0.74	25724
macro avg	0.74	0.74	0.74	25724
weighted avg	0.74	0.74	0.74	25724

```
TESTING SET METRICS
[[2576 1434]
 [ 920 3645]]
```

	precision	recall	f1-score	support
0	0.74	0.64	0.69	4010
1	0.72	0.80	0.76	4565
accuracy			0.73	8575
macro avg	0.73	0.72	0.72	8575
weighted avg	0.73	0.73	0.72	8575

AUC: 0.7785550636005822

- 3rd NN Model 3 hidden layers
- Neurons , lyr 1= 10, lyr2=6, lyr3=3
- Epoch = 100
- Activation function relu, output function =sigmoid

4th NN Model

```
804/804 [=====] - 1s 1ms/step
268/268 [=====] - 0s 1ms/step
268/268 [=====] - 0s 1ms/step
TRAINING SET METRICS
[[ 7953  4075]
 [ 2667 11029]]
```

	precision	recall	f1-score	support
0	0.75	0.66	0.70	12028
1	0.73	0.81	0.77	13696
accuracy			0.74	25724
macro avg	0.74	0.73	0.73	25724
weighted avg	0.74	0.74	0.74	25724

```
TESTING SET METRICS
[[2570 1440]
 [ 937 3628]]
```

	precision	recall	f1-score	support
0	0.73	0.64	0.68	4010
1	0.72	0.79	0.75	4565
accuracy			0.72	8575
macro avg	0.72	0.72	0.72	8575
weighted avg	0.72	0.72	0.72	8575

AUC: 0.7846020217801607

- 4th NN Model 4 hidden layers
- Neurons , lyr 1= 8, lyr2=6, lyr3=4, lyr=2
- Epoch = 50
- Activation function relu, output function =sigmoid

FINAL RESULT

- The problem for which applicant should be selected for funding by nonprofit foundation Alphabet Soup had a big data set to use for modeling. The target for the problem is what would predict a successful model for funding of applicants. The specific target is `Is_successful` and the variable features used for the model were **classification**, **name** and **application type**. The first model dropped the **name** and **EIN** because of the difficulty using these string values that would not be beneficial to the first run on the data set. The neurons, layers and activation functions selected for the first model, were selected as a baseline with the understanding that `relu` as a activation function due to its speed and `sigmoid` doing well at predicting. The neurons selected for the first model were randomly selected and the hidden layers were selected because I wanted to begin with an even number to discover if odd number of neurons or more even numbered neurons would produce results that would help identify which had a better accuracy prediction. After the first neural network model was ran, my intention was to get the accuracy metric to improve with adjustments to features. This is the action taken on the optimized models with the binning of the **NAME feature**. The model remained using `CLASSIFICATION` and after scaling the data I again used the baseline algorithms on the original model. Three additional neural network models were made, and the 2nd model utilized four neurons and epoch value of 150. I kept the activation function for the hidden layers as `reuls` for speed purposes and the `sigmoid` because of the usefulness in predicting. The accuracy score remained at 72%, and a similar hyper tune was done on the 4th model using four hidden layers, this time, with the first layer using 8 neurons, layer 2 six neurons and layer 3 using 4 neurons and layer 4 using 2 neurons. The accuracy score did not move by much and remained at 72%. The improvement came with the third model, this model utilized three hidden layers with the first layer using 10 neurons and the second hidden layer using 6 neurons and the third layer using 3 neurons. This model, had the best accuracy of 73%. It did not have a better area under curve than models 2 and 4, but precision and recall were slightly better than model 2 and 4. The improvement made with model 3 leads me to believe with additional feature engineering and further exploration on the neurons assigned to each layer and experimenting with odd number of hidden layers may have even better results. The data set is large, and the use of a neural network model has its advantages, but the linear regression model may be the best choice for this problem. It is less complex which would lead to easier interpretability to stakeholders. The neural network models had comparable results but required more tuning and testing to gain slightly better results.