Minimizing Financial Loss of Incorrect Predictions

Jason McDonald

Problem Statement:

The client in this study has provided a dataset of anonymized data in which he claims the metadata is unimportant. His only focus is to stop wasting so much money. For every false positive, the cost is $100. Each false negative costs $150.

To that end, the goal of this study is simply to maximize accuracy while increasing recall, so as to minimize the false negatives.

Provided Data and Evaluation:

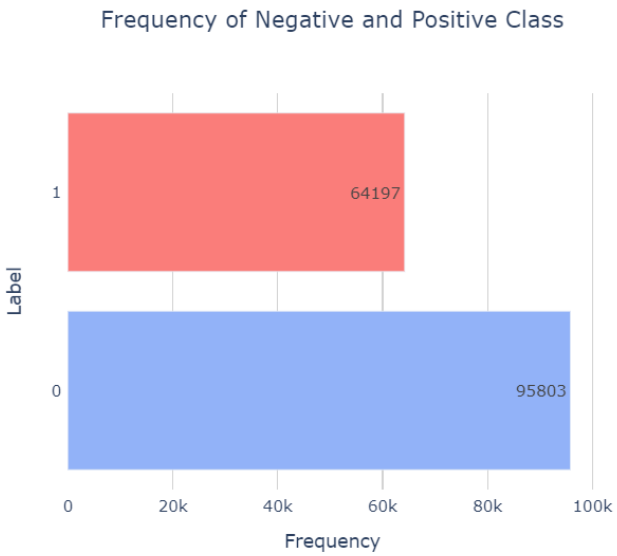
The data provided by the client consist of 50 feature columns and a single, binary classification represented by a 0 or 1.

Of the features, 45 are numeric, 2 are numeric but require preprocessing, and 3 are categorical. Of the categorical columns, 1 appears to contain a country name, a 2nd contains a month, and the 3rd contains a day of the week.

The 2 numeric columns that require preprocessing include one stored as a percentage that must be converted to a floating point number and the other is a currency value that needs to be stored a floating point number.

Quantity of Data: The dataset contains 160,000 rows of labeled data across 51 columns.

Response Variable: The response variable is a binary category representing the classification as a 1 for positive or 0 for negative.



The dataset has a mild class imbalance but at this time, this study will proceed with exploratory data analysis and preparing the data for model building without addressing the mild class imbalance.

Preprocessing Numeric Features: The 2 features that contain numeric data stored as a string will be handled as such:

Percentage: The '%' will be replaced with an empty string, the column will be converted to a numeric value, and finally that value will be divided by 100 to get a decimal representation of the percentage.

Currency: The '$' will be replaced by an empty string and the column will be converted to a numeric value.

Missing Data: The data contains a small number of missing data in each feature column and no missing data in the response.

The missing data in the numeric column will be addressed with a standard scaler imputing the mean of that column into the missing values.

The missing data in the categorical columns will be imputed by replacing the missing values with the most common value in that column.

Categorical Features: The categorical features will be one hot encoded to represent each possible category as a column containing either a 1 when that category is represented in the row, or 0 when not.

Scaling Data: The last step before exploring model building is to scale the data using a Standard Scaler from the SciKit Learn library.

Scoring Method to Evaluate All Models:

To be able to obtain a cost of a model on the provided data, out of fold predictions must be iteratively used until all values have been used out of fold exactly once.

For this study, this is accomplished by using the KFold functionality in the SciKit Learn Model Selection Library. A number of folds were created with a hold out set.

A loop was then created to loop through each KFold split of the data to where a model was trained on a training and validation set before predictions were made on the hold out set.

Those predictions were stored into a list to where they could be scored once all hold out sets were completed.

This same process was used for all model types explored.

XGBoost for Predicting Class:

The first model type to be explored by this study is an XGBoost Classifier. XGBoost, or Extreme Gradient Boosting can excel at many machine learning tasks by building decision trees in parallel rather than sequentially as in standard gradient boosting. This allows for XGBoost to benefit from the learning of numerous decision trees while being able to perform the exploration of the trees in much less time.

An extensive search of Hyperparameters was explored with the Hyperopt library. Tree of Parsen Estimators is used in the search function which performs a smart Bayesian optimization grid search.

The Hyperparameters explored are shown in table 1 below.

Table 1: Hyperparameter Search Space for XGBoost for 100 Evaluations

Hyperparameter	Hyperopt Type	Range
learning_rate	loguniform	-7 to 0

max_depth	int(uniform)	1 to 100
min_child_weight	loguniform	-2 to 3
subsample	uniform	.5 to 1
colsample_bytree	uniform	.5 to 1
gamma	loguniform	-10 to 10
alpha	loguniform	-10 to 10
lambda	loguniform	-10 to 10
Objective		Binary:Logistic
Early Stopping Rounds		10
Evaluation Metric		Error

Table 1 contains the hyperparameter search space for the XGBoost algorithm employed in the case study.

Table 2 contains the hyperparameters identified as the best by the Hyperopt tuning library in this case.

Table 2: Best Hyperparameters per Hyperopt for XGBoost after 100 Evaluations

Hyperparameter	Value
learning_rate	.0896
max_depth	91
min_child_weight	2
subsample	.8214
colsample_bytree	.9273
lambda	.0166
gamma	.0732
alpha	.0000467

Table 2 contains the hyperparameters identified by the Hyperopt library for the XGBoost algorithm employed in the case study.

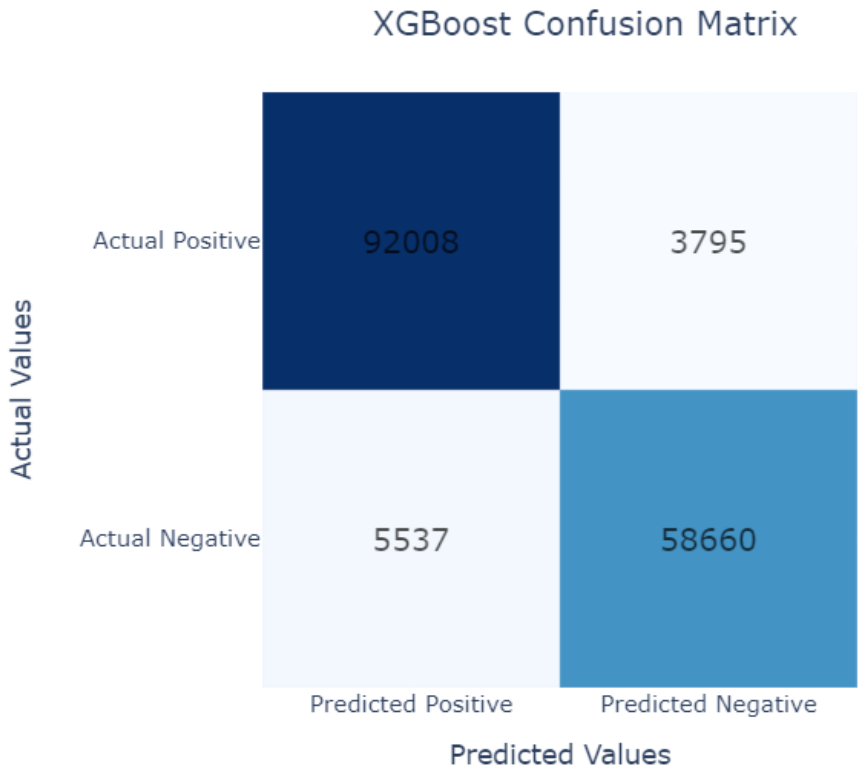
Once the best hyperparameters were chosen and stored, the process described in 'Scoring Method to Evaluate All Models' was performed so as to allow every row of data to be predicted and scored as an out-of-fold prediction where the training had not seen the predicted rows. This was to allow a true representation of the performance of the model.

Table 3 shows the performance of the XGBoost model.

Table 3: XGBoost Performance

Metric	Value
Accuracy	94.16%
Recall	0.9137
Cost	$1,122,950.00

Table 3 contains the performance of the XGBoost model



Loss of XGBoost Model: $1,122,950.00

Dense Neural Network for Predicting Class:

The last model this study will explore is a neural network model. By training the neural network on a GPU, an extensive window of hyperparameters can be explored. The exploration will take place using the Keras Tuner library and a Hyperband tuner.

The Hyperband tuner sets up a bracket of possible models and pits them against one another in a tournament. Each model is allowed to train for a short number of epochs and the winner of the 'game' moves on.

In the next round, the models are able to pick up where they left off due to the model state being saved to disk. The next round trains for additional epochs and again the best models from each 'game' move on to the next round. This process continues until the best model explored is identified.

The tuning is able to use early stopping to reduce training time for models which exhibit behavior indicating they have reached their optimal training period.

Table 4 shows the hyperparameter space explored by the Hyperband algorithm.

Table 4: Hyperparameters Explored by the Hyperband Algorithm

Default Values Underlined

Hyperparameter	Value
Number Layers	2 to 6 (excluding final layer)
Layer Size	32 to 768, by 32
Activation Function	Relu and Tanh
Regularization - Weight Decay	0 to .01
Dropout	False/True, .005 to .055 by .01
Learning Rate	.01, .001, .0001

Table 4 contains the hyperparameters explored by the Hyperband Algorithm in identifying an optimal Neural Network for the study.

In addition to the hyperparameters listed above, an initial flatten layer was added, along with a final dense layer with a sigmoid activation function so as to produce a binary classification output.

The model was built with an Adam optimizer and the loss function used was binary cross entropy.

After completion of the Hyperband tuning algorithm, the hyperparameters identified in table 5 were stored to be used by the KFold scoring process and final model build.

Table 5: Neural Network Structure as Identified as 'Best' by the Hyperband Tuning Algorithm

Layer	Units	Activation	Decay	Dropout
Flatten	784			
Dense Layer 1	672	Tanh	0.007	None

Dense Layer 2	128	Relu	0.006	0.045
Dense Layer 3	384	Relu	0.0005	0.015
Dense Layer 4	32	Relu	0.004	0.055
Dense Layer 5	128	Relu	0.0003	0.015
Dense Layer 6	1	Sigmoid	None	None

Table 5 contains the structure of the neural network identified by the hyperband tuning algorithm.

With the best hyperparameters chosen and stored, the process described in 'Scoring Method to Evaluate All Models' was followed as in the XGBoost model building to allow for true cost to be calculated of predictions on all rows of data without bleeding over from training data.

Early stopping was implemented to train the models within the out of fold training and scoring loop to optimize the training of the models for scoring purposes. A patience of 5 was used on the val_accuracy metric.

After making predictions on out-of-fold values for all rows in the dataset, an initial cost was calculated to be $604,600.

Because the Keras Fit function outputs probabilities, the probability cutoff of a positive or negative prediction can be tuned or optimized for best performance. By knowing a full set of predicted vs actual values, a simple loop mechanism was developed to find an optimal cutoff for the probability of a positive or negative class.

The cutoff that achieved the lowest cost and highest value returned was a cutoff of .63. Probabilities equal to or below .63 would be considered the negative class and those above would be considered the positive class.

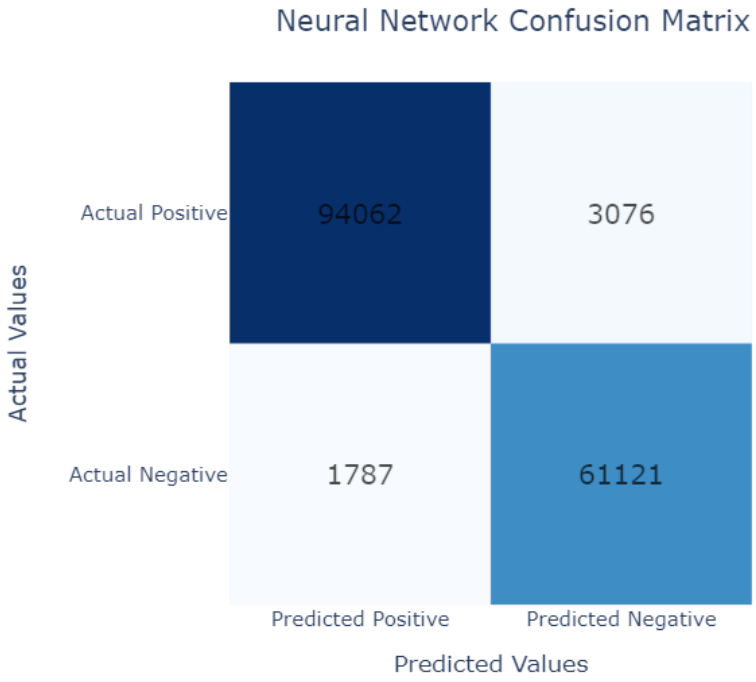
Final scoring of the predictions achieved a best case cost on the provided data of $575,550.00.

Table 6 shows the performance of the Neural Network model.

Table 6: Neural Network Performance

Metric	Value
Accuracy	96.96%
Recall	0.9521
Cost	$575,550.00

Table 6 contains the performance of the Neural Network model



Loss of Neural Network Model: $575,550.00

Conclusion: In comparing the XGBoost model to that of the Neural Network, it can be seen that the cost of the latter is nearly 50% less than that of the former. For this reason, the study proposes to proceed with productionalizing the Neural Network.

Given the conclusion of this study, a final neural network model with a first dense layer using a hyperbolic tangent function, followed by 4 layers of varying size using a rectified linear unit, before being passed through a layer with a sigmoid function for the binary classification step, can be delivered which minimizes cost by achieving nearly 97% accuracy and maximizing recall to effectively reduce the cost of an inaccurate prediction.