Final Project – Minimize the Monetary Loss for Company

Kevin Boyd
Laura Ahumada
Shikha Pandey

Abstract

The seventh and final case study for Quantifying the World is to predict a class being 0 or 1 with no information on the data other than incorrectly predicting class 1 cost the company 100 dollars and incorrectly predicting class 0 costs the company 40 dollars. This makes recall the priority metric, making sure false negatives, incorrectly predicting 1 as 0, are as low as possible while still making sure F1 score in general is still high since categorizing incorrectly 0 as 1 still hurts the business as well, with the goal of costing the company as little as possible. This data included a total of 51 features encompassed by 45 continuous, 5 categorical and the target itself with class 0 or 1. There were missing variables for all features. All NA columns were dropped for the 3 categorical features, 2 of the other categorical features were correctly changed to their numeric form and then all the missing values for the continuous variables were imputed by using KNN imputer to keep as much data as possible. The categorical features were turned into one-hot encoding and various models were run to minimize the misclassification resulting in lower cost for the company. Random Forest, XG Boost, and various Neural Networks combination were run where we found that the neural network with 33 epochs found by early stop turned out to be the best model costing only $110, 460 when tested on 20% of the data and having a recall and F1 score of 96%.

1 Introduction

1.1 Background

This case study was to build a model to predict a binary classification with no information on the data other than the fact that an incorrect classification of one class costs substantially more money than the other class. Our main goal is to cost the company as little as possible by lowering the number of incorrect classifications. Due to this fact our model must be as accurate as possible, especially when predicting the high-cost class. Below we will discuss some models that we believe were best suited to this task and gave us the highest chance of success for our client.

1.2 Random Forest Model

Random Forest models are often the first model to try when making predictions as they are robust to overfitting and suited to many different applications. These models are made from a large number of decision trees, allowing them to utilize a multitude of individual predictors into a versatile model and is what makes them so well suited for many different applications. Single decision trees on their own are prone to overfitting but this ensemble approach of using a 'forest' of decision trees mitigates the concern for overfitting. As seen in figure 1, each of the decision trees within the random forests have their own set of predictions and take a majority rule for the final output of the model reducing the variance.

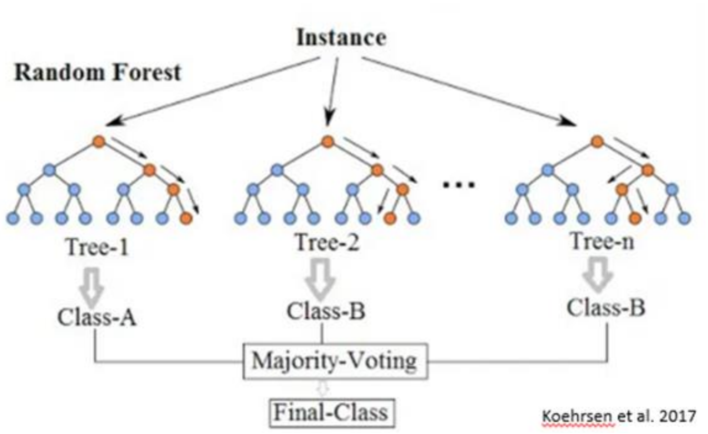


Figure 1. Random Forest diagram with multiple decision trees.

1.3 XG Boost

XG Boost is a powerful machine learning method that has gained popularity for its excellent predictive performance across a multitude of tasks including regression, ranking, and classification. This method was created to address limitations of traditional gradient boosting and combines the strength of decision trees. A key innovation of this method is the importance of regularization and parallel processing which improves both speed and performance. It also employs a technique called 'tree-pruning' during the building of decision trees eliminating branches that do not contribute substantially to the model's overall predictive power. Below in figure 2, this is an example of a plot of a single XG Boost decision tree from left to right.

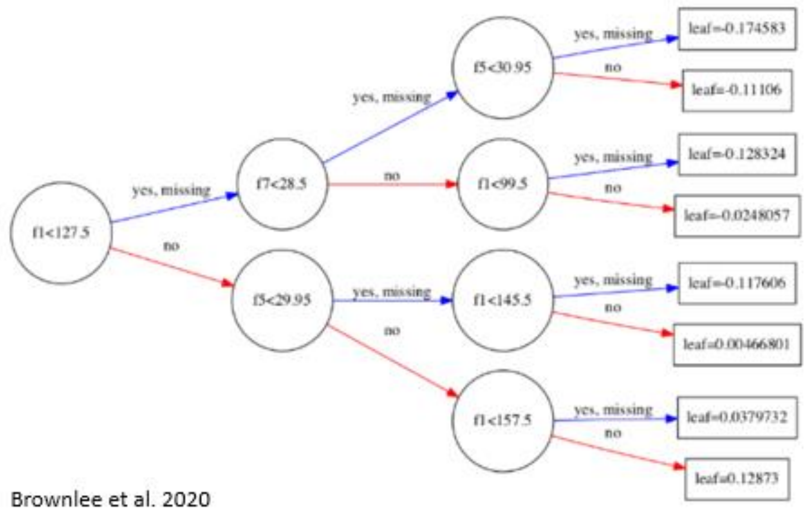


Figure 2. XG Boost plot of Single Decision Tree Left to Right

1.4 Dense Neural Networks

Neural networks are a specific type of machine learning that was inspired by the structure and function of the human brain. They consist of interconnected nodes organized into layers that include an input layer, hidden layers, output layers, weights, biases, and activation functions. Specifically dense neural networks are also known as fully connected neural networks because each node in a layer is connected to each node in the next layer. Every node receives input from all nodes in the previous layer which contributes to the overall output of the model. Dense neural networks are powerful for capturing intricate patterns in the data and have seen a lot of success when applied to image recognition, natural language processing, regression tasks, generative tasks, recommendation systems, time series forecasting, feature learning, and transfer learning. It is important to also mention when employing neural networks that each of these tasks have optimal architectures, hyperparameters, and training strategies. Below in figure 3 is an example of a dense neural network with three inputs in the input layer, two hidden layers with four neurons in each hidden layer, and a single output layer.

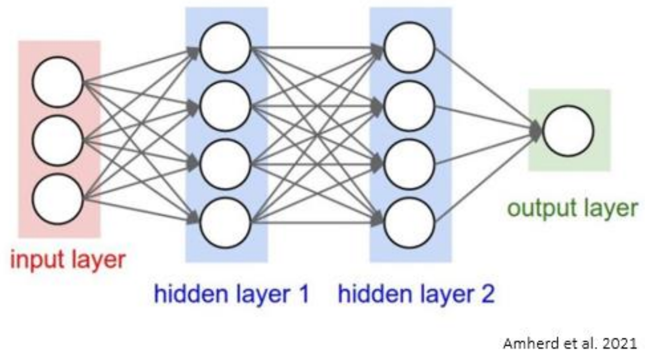


Figure 3. Dense 3-layer neural network example

2 Methods/ EDA

When observing the data, we see that it had 160,000 records and 50 columns with dummy column names except for the target labeled as y. The features encompassed 47 continuous variables, 3 categorical features and the target variable. There were no duplicates found, but all features had .01%-.02% missing variables as observed in Figure 4.

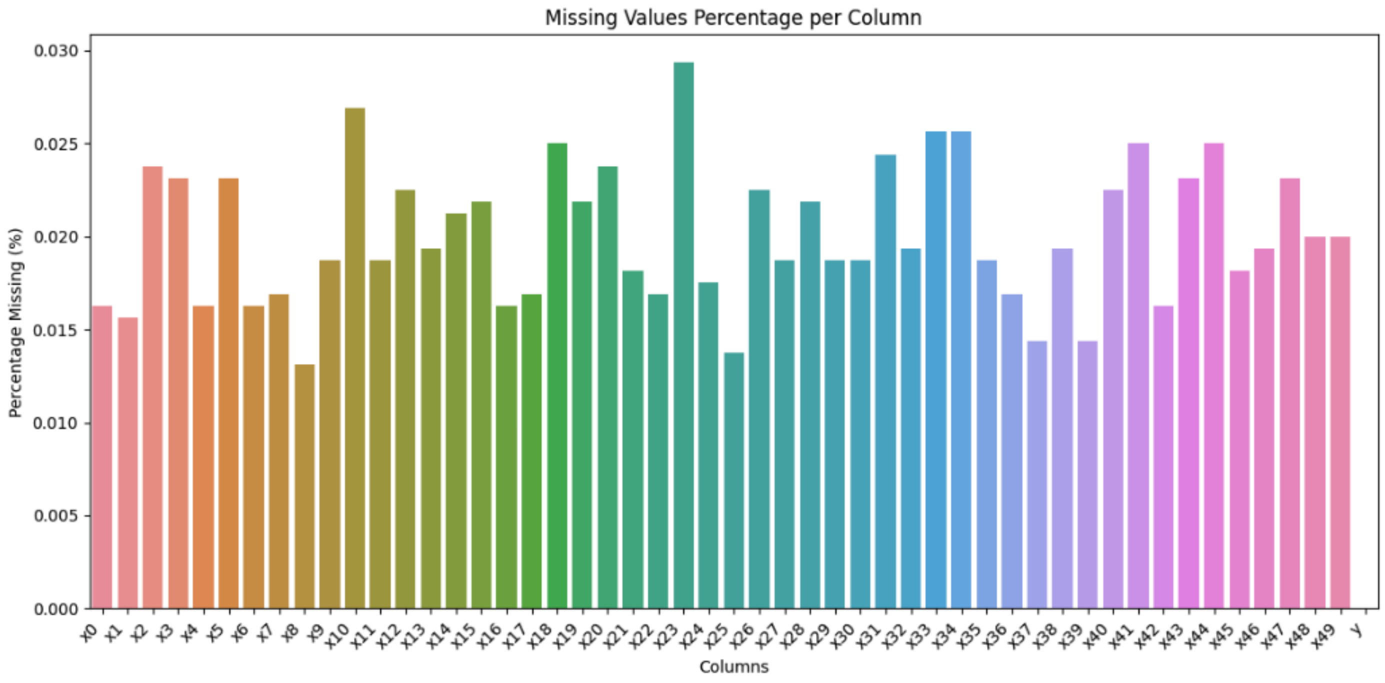


Figure 4. Missing variables

Even though the missing features were small enough to drop, the aim was to keep all the data given that predicting as accurately as possible was a priority; every miss was a cost for the company. Therefore, we wanted to look through each of the feature types in detail as much as possible to find patterns or see what could be done to impute missing variables.

The 5 categorical variables were the first variables that we addressed. 3 of those features were actual categorical variables representing region, month, and day of the week while the other 2 were numeric variables, one representing a percent and the other money. We changed the percent column to a continuous variable by removing the percent symbol and dividing it by 100 to turn it in ratio. For the money column, the dollar sign was removed, and the type was changed to continuous. We briefly looked at the distribution of the 3 categorical variables where we found that the region variable was largely unbalanced, having most of its data (80%) coming from Asia which could lead to some biases not being able to predict for the other regions. For the month's distribution, summer was the most condensed. For the days of the week, most of the data was on Wednesday and the middle of the week having very little data for Friday and Monday. Having an idea of these 3 categories we thought there could be some relationship that could be leveraged to help impute the missing variables. Different combinations between Country, Month and Day were tested using Chi–Square and there was not enough evidence to support the hypothesis that there was a significant association between any of the pairs. That meant we could not use either of the variables to fill in missing variables which is why we decided to drop those missing categorical variables. While doing so we found that there were some misspellings therefore looked for consistency to avoid having duplicates thought of as 2 different types: 'euorpe' and 'europe', but the misspellings were consistent.

We continued with the numeric features where we surprisingly found that all of them had normal distributions. Due to the anonymity of the features and lack of a subject matter expert (SME) we didn't want to assume relationships but instead used KNN imputer to fill in the missing variables. At that point we had examined both data types within the data set, addressed the missing variables and proceeded to assess the target feature. There we found it was little imbalanced where ~60% of the data was target 0 while ~40% of the data was target 1 as observed in Figure 5. This was very unfortunate because that meant the model would have more information to predict correctly 0 more so than 1 when in reality classifying 1 is actually more important. To account for the imbalance of the targets in each of the models we made sure to adjust the targets' weights.

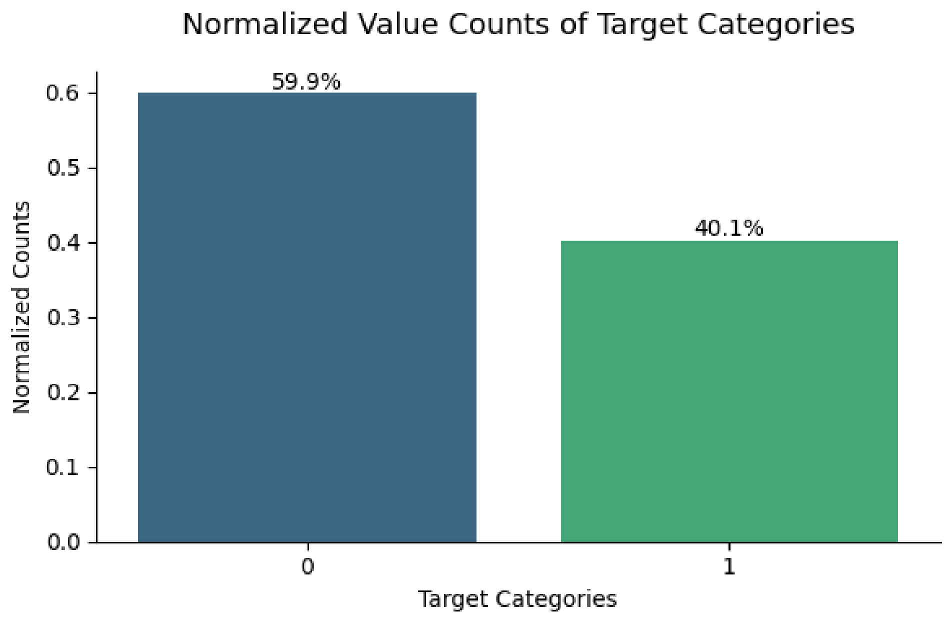
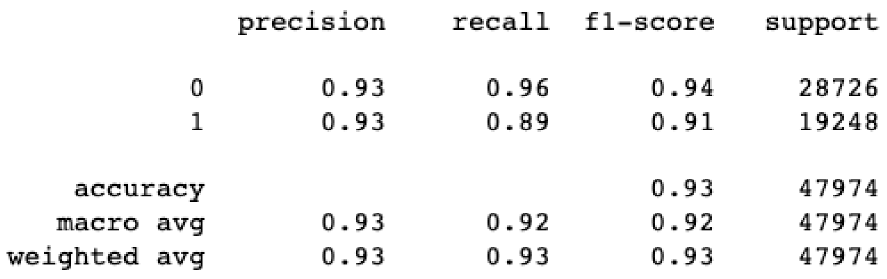


Figure 5. Target distribution

3 Results

Data was split into 70% Training and 30% testing. We then began the model process. Given that the main purpose of the project was to minimize the cost and there was no importance to do any model interpretation we decided to start with a simple yet effective model that captures complex relationships, Random Forest. A random search was performed to obtain the best parameters without spending too much time on it. We made sure to retain these parameters 100 trees and balance class weights. These were added to the parameters derived from the random search, encompassing a maximum depth of 30, maximum features of 40, minimum samples per leaf of 3, and a minimum sample split of 12. It led to good scores with a recall of 92% and an f1 score of 93% as observed in Figure 6. On the test data made up of 30% of the total data provided as mentioned earlier, it meant a total cost to the company of $247,780. This calculation was used utilizing the confusion matrix in Figure 7, adding false positive and false negatives with their corresponding cost.



```
              precision    recall  f1-score   support

           0       0.93      0.96      0.94     28726
           1       0.93      0.89      0.91     19248

    accuracy                           0.93     47974
   macro avg       0.93      0.92      0.92     47974
weighted avg       0.93      0.93      0.93     47974
```

Figure 6. Classification report of Random Forest

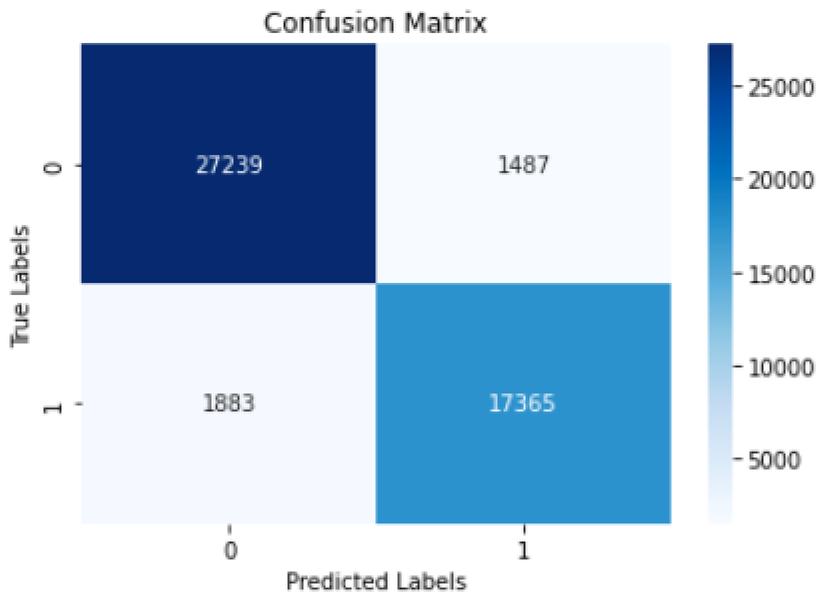


Figure 7. Confusion Matrix of Random Forest results

We then proceeded with XG Boost to boost the scores. Due to the imbalanced labels and the fact that predicting the minority class (target 1) was a priority we accounted for the weights again using scale pos weight, number of rounds to 600, included a max depth of 10, lowered the learning rate to 0.1 and utilized log loss as the evaluation metric since we were doing a binary classification. The number of rounds was not initially set to 600, it had been set to 300. It was through testing different number of rounds using cross validation results that we ended up choosing 600. At that point we saw the train and test results flatten out as observed in Figure 8, squeezing as much information as possible. When compared to the Random Forest results it improved the macro average recall by 2% (94%) and macro average f1 score by 1% (93%) as seen in figure 9. It especially improved the recall of target 1 by 0.4

which was observed in the total cost now being $202,620 (Figure 10), which was a $45,160 decrease from that observed in Random Forest.

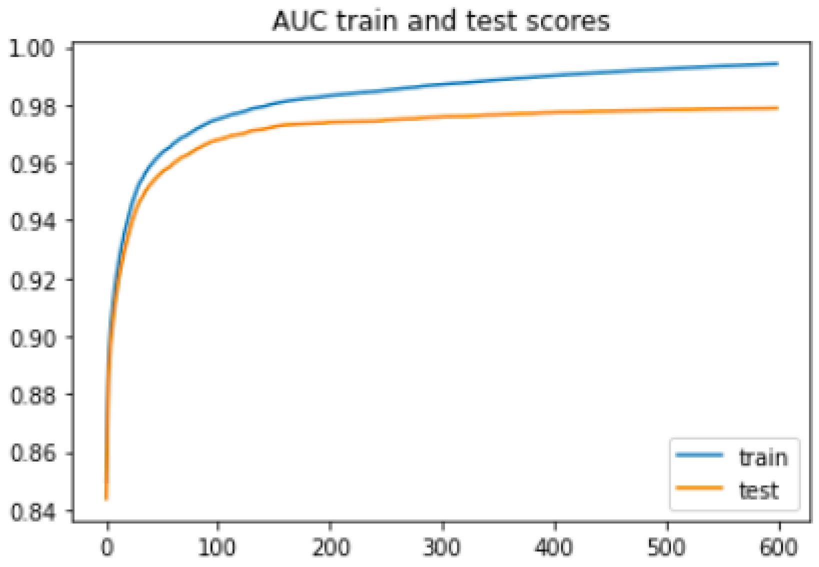
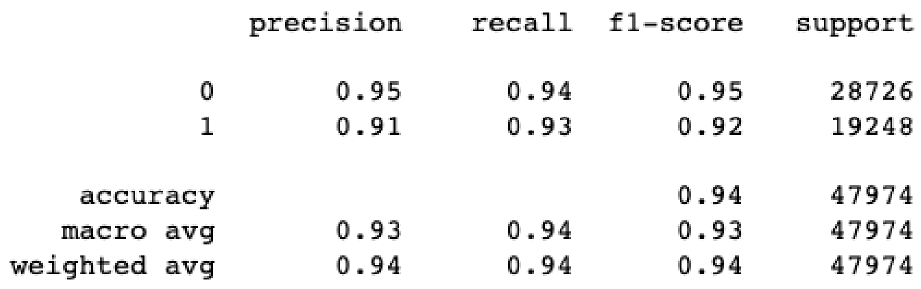


Figure 8. XG Boost cross validation AUC results for the train and test by round



```
              precision    recall  f1-score   support

           0       0.95      0.94      0.95     28726
           1       0.91      0.93      0.92     19248

    accuracy                           0.94     47974
   macro avg       0.93      0.94      0.93     47974
weighted avg       0.94      0.94      0.94     47974
```

Figure 9. Classification Report of XG Boost

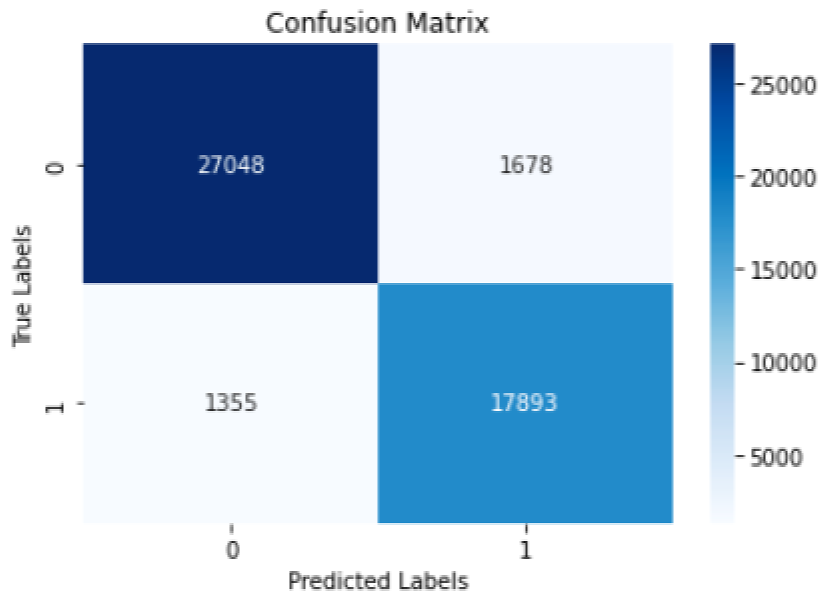


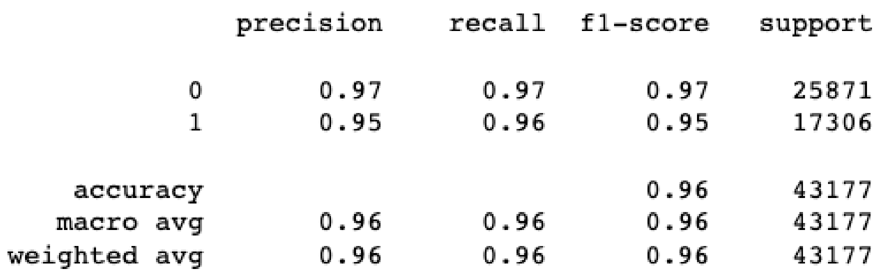
Figure 10. Confusion Matrix of the XG Boost results

Even though the cost did decrease, we decided to try an even more complex model, a Dense Neural Network, to reduce the cost even further. Knowing that the costumer did not care about the interpretation of the model and only cared about the predictions made this a perfect model candidate. The first neural network architecture that led to lower results than the XG boost comprised various layers: initially, a Dense layer with 100 neurons using ReLU activation, followed by another Dense layer consisting of 50 neurons with ReLU activation, a dropout layer of 0.3, another dense layer with 30 neurons utilizing ReLU activation, and a classification sigmoid layer. For optimization, an Adam optimizer was utilized, incorporating a batch size of 1,000, and class weights were assigned to accentuate the importance of the minority class 1 just like the previous models. The model was set to run for 1,000 epochs, employing early stopping criteria to halt training after five consecutive instances of minimal loss reduction (0.0001). The model ran for 112 epochs, yielding a macro recall and f1 score of 93%, resulting in a misclassification cost of $193,520.

That lead us to assume that 1000 batch size could have been the issue, so we re-ran it again with batch size of 100 instead. The model was stopped at 34 epochs by early stop. The macro average recall and f1 score did improve to 95%, and saw the cost largely decrease to $135,000 which was a $58,520 decrease.

Observing the large cost saving we decided to run it again with an even smaller batch size of 20 batch. That led to a very small cost decrease of $1,820, a total cost of $133,180. To check again if assigning the weights was indeed helpful it was run with a batch size of 20 without the weight adjustments which yielded to cost of $151,540, an increase cost of $18,360 showing how assigning class weight was necessary and useful.

Thinking we had lowered the batch size too much we decided to run it last time with a batch size of 50 which was stopped at 33 epochs by early stop. It led to best results with a misclassification cost of $110,460, a decrease cost of $22,720 from that ran with a batch size 20. The classification report of that model can be observed in Figure 10 showing 96% scores for all precision, recall and f1 score. Same steps were applied to another neural network with larger and different iteration of number of neuron sizes from 500-1000, changing the batch sizes, but all attempts lead to higher cost than our best observed $110,460, the lowest it got with more neurons was $128,700. We also provided the confusion matrix showing a summary of the predictions below in figure 11 below.



```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97     25871
           1       0.95      0.96      0.95     17306

    accuracy                           0.96     43177
   macro avg       0.96      0.96      0.96     43177
weighted avg       0.96      0.96      0.96     43177
```

Figure 10. Classification report of Neural Network.

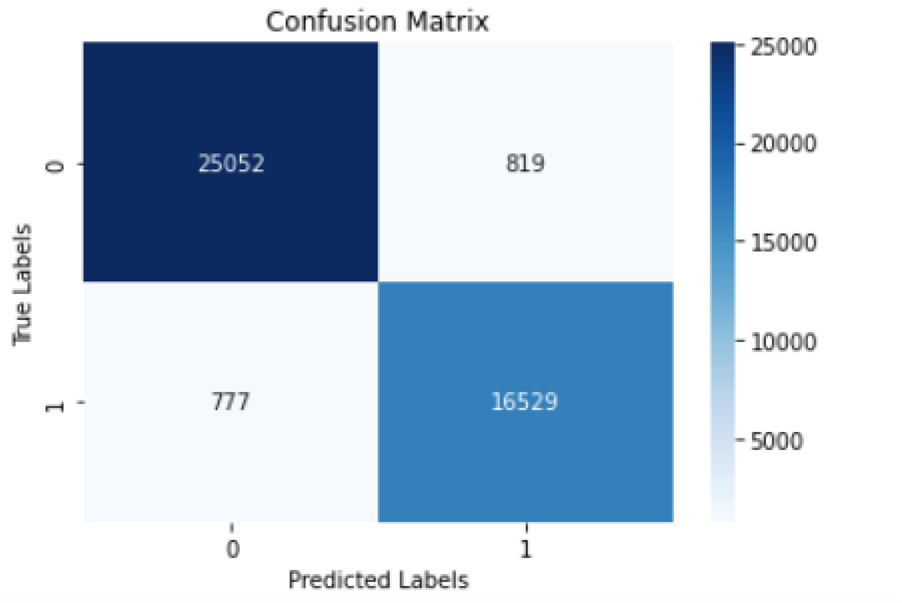


Figure 11. Confusion Matrix of Neural Network (50 Batches, 33 Epochs, class weight Adj, 5 layers)

After identifying the optimal model, our next step was to conduct cross-validation to assess its trade-off between bias and variance while validating the model's performance. Employing a five-fold validation approach, we achieved recall scores of 95% with a narrow standard deviation of 0.004. These results, when consolidated, align with the specific scores outlined in Figure 11. This validation confirmed the model's lack of bias, its balanced nature, and its reliability. The neural network architecture comprised five dense layers: the initial layer featured 100 neurons with a ReLU activation function, followed by another dense layer consisting of 50 neurons with ReLU activation, a dropout layer of 0.3, an additional dense layer with 30 neurons utilizing ReLU activation, and finally, a classification sigmoid layer. The model utilized the Adam optimizer, employed 50 batches, ran for 33 epochs, and underwent weight adjustments. Remarkably, this neural network successfully reduced the cost from the initial random forest results of $247,780 down to $110,460 minimizing the cost by more than half, with a recall and f1 score of 96%,

4 Code

Link to the Jupyter notebook with code assisting in the case study below:
https://github.com/lauraah10/QW-Projects/blob/main/QW_FinalProject_NN.ipynb