Developing Classification Model on Unknown Dataset

1. Introduction

The team has been given an anonymized set of data. Our goal is to develop a machine learning model that can predict the binary target variable 'y'. This scenario of needing to develop a predictive model using data with little to no background can come up frequently in the data science field. Sources that include medical patient data or classified data would require development with very little information.

For this case study we will need to develop our binary classification model keeping one in mind, there is a cost of $35 for every false positive, and $15 for every false negative predicted. Since misclassification comes at a cost, our goal while developing this model is to minimize misclassification costs.

2. Data

The data provided included 160,000 rows with 50 columns of features labeled x0 to x49 (Table 1). The data included a response variable 'y' that was a binary value. Of the feature columns, 3 were text based and the rest were numeric values.

Table 1. The raw format of the dataset before preprocessing.

RangeIndex: 160000 entries, 0 to 159999
Data columns (total 51 columns):

#	Column	Non-Null Count	Dtype
0	x0	159974 non-null	float64
1	x1	159975 non-null	float64
2	x2	159962 non-null	float64
3	x3	159963 non-null	float64
4	x4	159974 non-null	float64
5	x5	159963 non-null	float64
...			
45	x45	159971 non-null	float64
46	x46	159969 non-null	float64
47	x47	159963 non-null	float64
48	x48	159968 non-null	float64
49	x49	159968 non-null	float64
50	y	160000 non-null	int64

dtypes: float64(45), int64(1), object(5)

The distribution of the response variable is observed and plotted below (Fig. 1). The target variable 'y' had no missing values, with 59.88% of the repsonses being 0 and 40.12% being 1.

The dataset is not significantly imbalanced enough to warrant any over or under-sampling techniques.

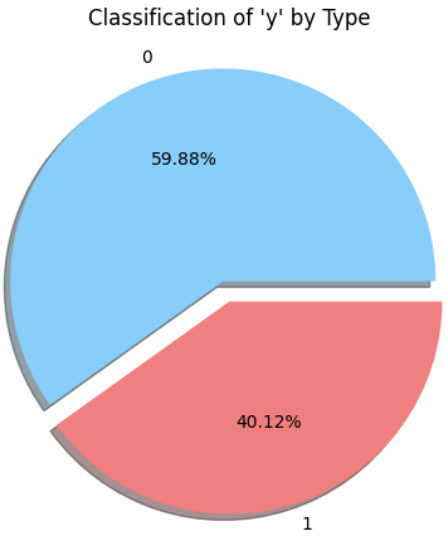

Figure 1. The distribution of 1 and 0 for target y variable.

In terms of missing data, columns x0 to x49, collectively, had 1,520 missing values or less than 1% of total records. Regardless of the relatively small amount of missing data, this will negatively affect our dense neural network developed for this project. The variable distribution via histograms can be seen in Figure 2. Aside from the last histogram representing the response variable, all numeric variables shown in Figure 2 are normally distributed. Based on their observed distribution, we imputed the missing numerical values using their median. The lack of domain knowledge may deter other teams from imputing the missing values, but we decided the consistent normal distributions and relatively small amounts of missing values would make imputation a viable option for preprocessing the data.

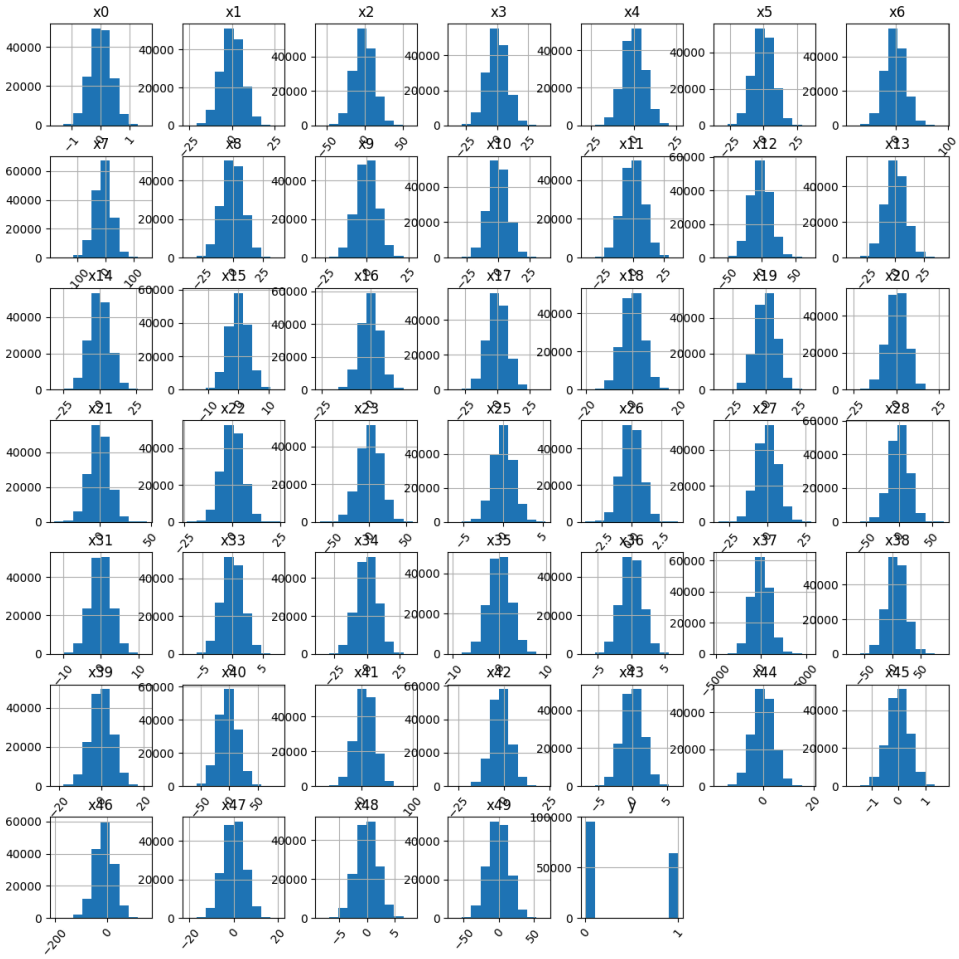

Figure 2. The distribution of each numerical variable.

In terms of missing data for the other non-numerical variables not included in Figure 2, i.e. x24, x29, x30, x32 and x37, a different approach was used. These variables all had a number of unique values, with x24 having 3 continents, x29 having 12 months, x30 having 5 days of the work week, and x32 having a specific number of percentages. X37, however, had 129,000 unique values. A closer look into x37 led to the '$' symbol being removed from all its values and the variable itself converted into "float64".

To aid the effort of training an efficient neural network, all 46 numeric variables are scaled from 0 to 1 using `MinMaxScaler(feature_range=(0, 1))`. And likewise, the 4 categorical variables x24, x29, x30 and x32 are one-hot coded, totaling in 29 variables. Ultimately, the preprocessed dataset, aside from the response variable, now consists of 75 feature variables and 160,000 entries.

In a real-world situation, the data that is validated from the model must not be seen by the model itself. Therefore, the data is split into two parts, where 80% will be trained by the model and 20% will be validated from the model.

A monetary metric was given by the client to evaluate the final model. For the misclassification cost metric, false positives result in a monetary loss of $35, while false negatives result in a monetary loss of $15.

3. Dense Neural Network

Other than the response variable, the 75 features are assumed to be sequences that can be used in the `Sequential()` model from Tensorflow's keras in Python.

3.1. Design

To conduct a preliminary assessment, a neural network with only 1 hidden layer is built. Its input layer with a shape of 75 is used prior to a dense hidden layer, using the rectified linear unit (ReLU) activation function, with 75 neurons. Additionally, dropout with a fraction of 0.2 is used after the dense layer to reduce overfitting. For the output layer, the sigmoid activation function with 1 node is used. To compile the network, the Adam optimizer is used with a 1e-3 learning rate and the rest of the arguments as defaults, along with the binary cross-entropy loss and the accuracy metric.

To create another model for comparison, a neural network with 5 hidden layers is created, with their number of neurons in 5 decrements of multiples of 75, starting with 375 neurons for the first dense layer, i.e., the second dense layer has 300 neurons, and so on. A dropout is added after every dense layer as well, and both the input and output layers are similar to the first preliminary neural network.

3.2. Hypertuning

The built-in early stopping function is used to monitor the validation loss with patience set to 3 epochs. In other words, training will be stopped when there are no improvements to the validation loss after 3 epochs. To fit this model, both the train and the test set are fitted using 100 epochs and a batch size of 100.

4. Results

The result of 1 hidden layer neural network is sub-optimal, which led to a total budget loss of $360,000, when validated against the entire 160,000 out-of-fold entries. However, the neural network with 5 dense layers managed to achieve a budget loss of $200,000. The team decided that this was a good stopping point in terms of increasing the complexity of this neural work. The classification report and confusion matrix of this final model can be seen in Table 2 and Figure 3 respectively.

Table 2. The classification report for the dense neural network.

	precision	recall	f1-score	support
0	0.97	0.95	0.96	19180
1	0.93	0.96	0.94	12820
accuracy			0.95	32000
macro avg	0.95	0.95	0.95	32000
weighted avg	0.95	0.95	0.95	32000

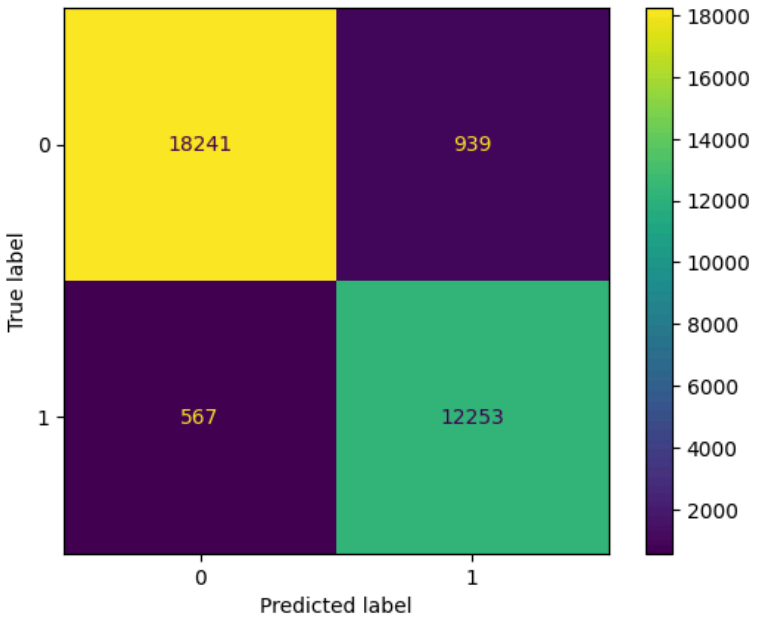

Figure 3. The confusion matrix for the dense neural network.

5. Conclusion

For this case study we were provided with an anonymized dataset of 160,000 observations containing 50 features and were tasked with modeling the target binary variable. Regarding the data, we were given no background information to help us understand the context of this study. The team was able to gain experience working with anonymized data and provided us with the autonomy to develop a model utilizing any method learned throughout this course.

The team developed a deep neural network model with a classifier to predict the target variable 'y'. After scaling the data, splitting the data into an 80/20 training/test split, the team utilized a rectified linear unit (ReLU) activation function and dense layers using the sigmoid activation function to produce a binary output. Additionally, the final model produced an accuracy of 0.95.

Based on our final model's confusion matrix, our misclassification cost totaled $199,520.

6. Appendix