Banking Campaign Output Prediction

Henok Birru

Introduction

Thousands of distinct sectors and products are represented in the marketing industry. According to 2020 surveys, enterprises and organisations in the banking and financial industry spends an average of 13% of their total budget on marketing.[1]

The Financial Brand on their blog “People Think Their Bank Should Be Calling Them More Often”, mention that people want to a strong connection with their bank, but on their terms and branded calling or telemarketing is a way to meet those terms and increase engagement.

The aim of this project is to build a machine learning particularly neural network classifier model that predicts whether a customer will agree to subscribe to a product(term deposit).

The model uses different customer’s features such as age, job, education, previous contact and others, to predict their response towards a phone call campaign.

This report explains the dataset we are using to train the model and the preprocessing steps taken before feed it to the network. The modeling section of the report describes what modeling approach is taken and what techniques have been applied to improve the performance of the model.

Data

Data Description

The data is related with direct marketing campaigns of a banking institution. The marketing campaigns were based on phone calls. In order to determine if the product (bank term deposit) would be subscribed ('yes') or not ('no'), it was sometimes necessary to make more than one contact with the same client.

The dataset contains 41188 rows and 21 columns. The 20 columns are the inputs and the last column contains the response of a user for the campaign.

#	Column	Non-Null Count	Dtype
0	age	41188 non-null	int64
1	job	41188 non-null	object
2	marital	41188 non-null	object
3	education	41188 non-null	object
4	default	41188 non-null	object
5	housing	41188 non-null	object
6	loan	41188 non-null	object
7	contact	41188 non-null	object
8	month	41188 non-null	object
9	day_of_week	41188 non-null	object
10	duration	41188 non-null	int64
11	campaign	41188 non-null	int64
12	pdays	41188 non-null	int64
13	previous	41188 non-null	int64
14	poutcome	41188 non-null	object
15	emp.var.rate	41188 non-null	float64
16	cons.price.idx	41188 non-null	float64
17	cons.conf.idx	41188 non-null	float64
18	euribor3m	41188 non-null	float64
19	nr.employed	41188 non-null	float64
20	y	41188 non-null	object

Figure - 1 Dataset Description

As shown in figure-1, all the 21 columns don’t have missing values and the dataset contains a mixture of both numerical and categorial data types. The last column ‘y’ is the output variable which has a binary values of ‘yes’ or ‘no’.

Data Preprocessing

Preparing the data and making it suitable for a model is a crucial step while creating a machine-learning model. The raw data might contain missing information or inconvenient data format for the desired model, hence to build a reliable model and improve its performance, it is recommended to do data cleaning.

The dataset of this project is described in the previous section briefly. Duration is one of the feature of the dataset which contains the value of last contact duration in seconds. As given in the project description document, this column is highly correlated to the class variable and it is suggested to exclude it from the dataset for the reliability of the predictive model thus, it is dropped.

Next, as mentioned before the features of this dataset is composed of both qualitative and quantitative types, therefore the preprocessing step should be handled separately.

A pandas dataframe with categorical features and another dataframe with numerical features are created.

Although, we didn’t find any explicit missing values when we check using the pandas ‘isnull’ function, there were some examples with a value of ‘**unknown**’ which can be considered as missing values. There are different techniques to handle missing data in machine learning, we can either drop all the examples with unknown values or fill them with generated values. For this scenario, since there are many features which contains unknow value, we decided to replace them with the most frequent value of a given feature.

```
job             330
marital          80
education      1731
default        8597
housing         990
loan            990
contact           0
month             0
day_of_week       0
poutcome          0
dtype: int64
```

Figure - 2- ‘Unknown’ values count

Following to handling missing values, encoding the categorical features to numerical representation comes. Machine-learning algorithms require numerical data, thus the qualitative data must be converted to numerical features before using them to train a model.

The most common technique to perform categorical data encoding is one hot encoding. It creates a new column for each unique value contained in a given feature. Although, while encoding it expands the feature set, since we have enough examples in our dataset, we decided to use this technique. After performing the encoding process the number of feature columns increase to 56.

Feature scaling is the data preprocessing technique we used for the numeric variables. In a training set usually numeric features have very different scale, that might make our machine learning model to interpret incorrectly by giving more attention to the larger values.

Finally, as we did for the feature vectors, the output binary class is also represented as 0 and 1, instead of ‘yes’ and ‘no’.

Modeling

The modeling step is started by dividing the dataset into training and testing dataset. 80 percent (32950 rows) for training and and 20 percent(8238 rows) for testing are assigned.

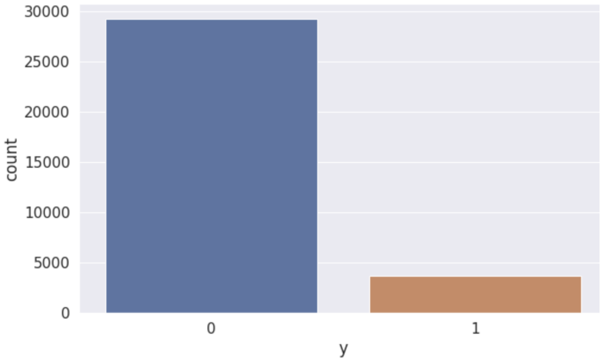

Figure- 3- Output class distribution

Figure-3 depicts the distribution of the output class variable. There are more examples with value of 'no' than 'yes' and that will also be reflected on our model evaluation.

Our neural network model has four layers two dense hidden layers, one dropout layer and one output layer. The first hidden layer has 20 units and the second hidden layer has 10 units.

Every neural network has activation function along with the layers and the units on the layers. The activation functions decide whether a neuron should be activated or not. There are different activation function such as relu, sigmoid, tanh and so on. Recently, Relu is the popular chosen activation function, because a model that uses relu is easier to train and often achieves better performance. The other activation function suffer the issue of vanishing gradient decent. However, for a classification problem like our scenario it is recommended to use sigmoid activation function at the output layer. The sigmoid activation function has a characteristic of taking any real value and map it to 0 or 1.

The relu activation function is used on the hidden layers and the sigmoid activation function placed on the output layer. Figure-4 depicts the number of layers created on the network and the amount of units on each layers, in addition the number of params is also displayed.

```
Model: "sequential_49"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_146 (Dense)           (None, 20)                1140

 dropout_45 (Dropout)        (None, 20)                0

 dense_147 (Dense)           (None, 10)                210

 dense_148 (Dense)           (None, 1)                 11

=================================================================
Total params: 1,361
Trainable params: 1,361
Non-trainable params: 0
```

Figure-4 Model Layers and params

Our model uses the Adam optimizer as optimization algorithm. Optimization algorithms are used to improve the parameters of neural network model such as weight and learning rate to reduce the loses. Although, gradient descent is the most basic and most used algorithm, recently other form of optimizers are being used. The Adam(Adaptive Moment Estimation) optimizer is extension of stochastic gradient descent that has been used in many different deeplearning models.

Determining a loss function for a given model is another step while building a machine-learning model. Since our model is a binary classification model, the binary crossentropy is being used. It computes the cross-entropy loss between true labels and predicted labels.

Model Performance Evaluation

Evaluating a model is very crucial step in machine learning as building it. The type of evaluation metrics to use for a model directly correlates with the type of task that our model is doing. A classification model and a regression model are evaluated using different evaluation metrics. Accuracy, precision, recall, f-score and others are the metrics that can be used mostly for classification model.

Our model has an accuracy of around 90 percent, which means it can able to predict response of 9 customers out of 10.

As shown in figure-5 class 0 has more score across all the metrics than class 1. This is because of the data we used to train the model, it has more examples with the output value of no than yes.

```
258/258 [==============================] - 0s 1ms/step
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      7294
           1       0.68      0.23      0.35       944

    accuracy                           0.90      8238
   macro avg       0.80      0.61      0.65      8238
weighted avg       0.88      0.90      0.88      8238
```

Figure - 5 classification report

The other metrics used to evaluate the model is a confusion matrix, which shows a summary of prediction results our model generates.

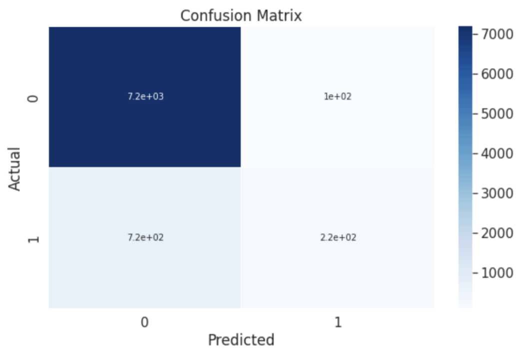

Figure-6 Confusion matrix

In addition to the steps we took in the data preprocessing, different combination of hyperparameters are also be experimented to get good model accuracy. The model can even perform better if we have for example balanced class dataset. The SMOTE(Synthetic Minority Oversampling Technique) was applied to solve the class imbalance issue, however it doesn't show much difference, perhaps instead of randomly generated fake data, having real samples might solve the issue.

Early stopping method is also applied while performing the training to avoid model overfitting problem. Model overfitting is a scenario when a model can't able to generalize well to new data and try to overfit the training data. The early stopping technique allow us to halt the training when the model stops improving on the validation dataset.

The dropout technique is also used along with the early stopping method. It is a way of ignoring randomly selected neurons during training. A dropout layer is added before the second hidden layer. 25 percent dropout rate is given as weight constraint on that layer.

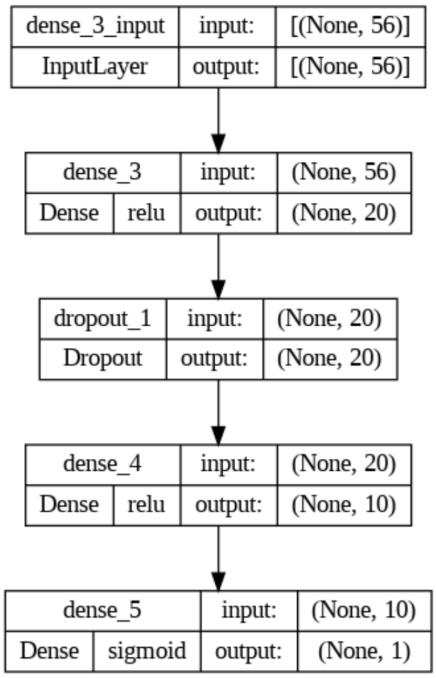

Figure-7 Network architecture

Figure-7 depicts the architecture of our neural network generated by using the built-in keras plot model function.

Conclusion

The objective of this machine-learning project was to build a neural network model that predicts a customer response towards a phone call banking campaign. Data exploration, data preprocessing, model implementation and evaluation have been done and reported on this document.

The scientific bottleneck we faced while doing the project was first the issue of class imbalancing, as mentioned before our dataset mainly contains examples with ‘no’ values and that might affect the performance of our model, different techniques like SMOTE were being tried but they don’t show much result difference. Additionally, deciding the network architecture was difficult, however after experimenting different approaches, the better architecture has been selected.

Finally, this project gives me an opportunity to learn about neural networks implementation and variant workarounds to issues related to deep learning.

References

[1] Activewin, “HOW MUCH DO BANKS SPEND ON MARKETING?”, May. 2020