Proof of Concept: can businesses of all sizes find value in deep learning models, even with limited data

Evaluating Output & Performance of a Neural Network for Small-Sized Insurance Company

By: Ali Gowani

Table of Contents

OVERVIEW: PROBLEM STATEMENT

Originally, I was thinking of using a readily available dataset and running a deep learning model. While this would have been a decent start, I did not think it would be a real case as compared to what I would encounter as a consultant. I brainstormed ideas on how I might help a client that is looking to improve its business model. This made me think of my friend, Tawfiq (Tom) Hassan, who owns a small insurance company, Texas Giant Insurance (TGI), that focuses on providing commercial and personal insurance programs to its clients. TGI is an independent insurance company with an in-depth knowledge of multiple insurance products and carriers. They proactively provide service to their policyholders and present them to their clients. After speaking with Tom, about the course and the final project, we decided to look at improving customer experience and create a Neural Network (NN) to see whether it would be a viable model compared to other more traditional algorithms and methods.

The goal of this project is to first, validate that a NN model is more powerful in accuracy than other models and two, how we can leverage this information to mitigate customers from leaving and reclaim customers that have left TGI.

DATA: EVALUATE DATA & CONDUCT EXPLORATORY DATA ANAYLYSIS

The dataset we received was of TGI customers between January 2017 and December 2019. The dataset was not properly formatted to be consumed by our models, but we did not have any missing values. As with insurance companies, their data is usually stored in a system that was not made for analysis but rather for accounting purposes. A significant amount of time was spent to learn the data features and determine any meaningful features that should be extracted. After going back and forward with the client (TGI), we ended up getting access to the data of 794 customers (observations). However, 81 of these observations were of customers who had inquired about products and services from TGI but never ended up becoming a customer. We ignored these observations, and this reduced our dataset to 713 observations. Since the insurance industry is heavily regulated, I was not able to get additional demographic information of the customer and had to do the best I could with the provided dataset.

Provided Features:

Line of Business/Non-Premium	Is Multi-Entity	Policy Status	Active Customer	Customer Number	Type of Business	New Business/ Renewal Group	Total Cost

Policy Effective Date	Policy Expiration Date	Customer Address 1	Customer Address 2	Customer City	Customer State	Customer Zip Code	Cancel Date
Cancel Reason	Writing Company	Policy Number	Invoice Month	Invoice Year	First Written Date	Invoice GL Month	

Selected & Newly Created Features:

Customer ID	Gender	Referrer	State	PaidFull Premium Before	Became Cust	DurationAsCust	Accident/ Health (P) Duration
Accident/ Health (P) Amount	Builders Risk (P) Duration	Builders Risk (P) Amount	Dwelling Fire Duration	Dwelling Fire Amount	Earthquake (P) Duration	Earthquake (P) Amount	Flood Duration
Flood Amount	Homeowners Duration	Homeowners Amount	Life (P) Duration	Life (P) Amount	Motorcycle Duration	Motorcycle Amount	Private Passenger Auto Duration
Private Passenger Auto Amount	Umbrella (P) Duration	Umbrella (P) Amount	Total Duration	Total Amount	Still Customer		

We created new features from the dataset that was provided and formatted the data, so each observation is associated with that customer. One of the features we created was the duration of the customer (DurationAsCust) so that even if the duration of the policy changed or the type of policies changed between the years, we could capture the entire value of the customer. Another feature we created was to capture the significance of the customer so if the customer had multiple policies per year, we wanted to capture the sum of all those policies for the life of the customer (Total Duration).

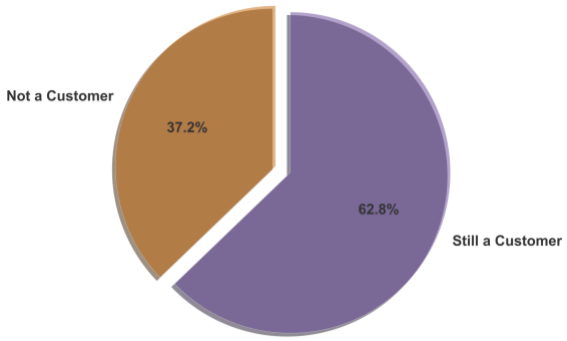
Exploratory Data Analysis (EDA):

After understanding the features and getting the data formatted in a proper manner, we were able to conduct EDA. Most of the EDA figures, as well as, Histograms, Correlation Plot, Mean, Standard Deviations, Minimum, Maximum and other summary statistic as part of EDA are provided in the Appendix.

Figure 1 shows us the split of our response (target) variable: StillCustomer (0: Not a Customer, 1: Still a Customer). Out of 713 observations, 62.7% (448) are still a customer and 37.2% (265) are no longer a customer. While we want a good balance between the classes in our response variable, the 63% to 37% split is not terrible. We did execute class weights function in the sklearn library to balance the model but

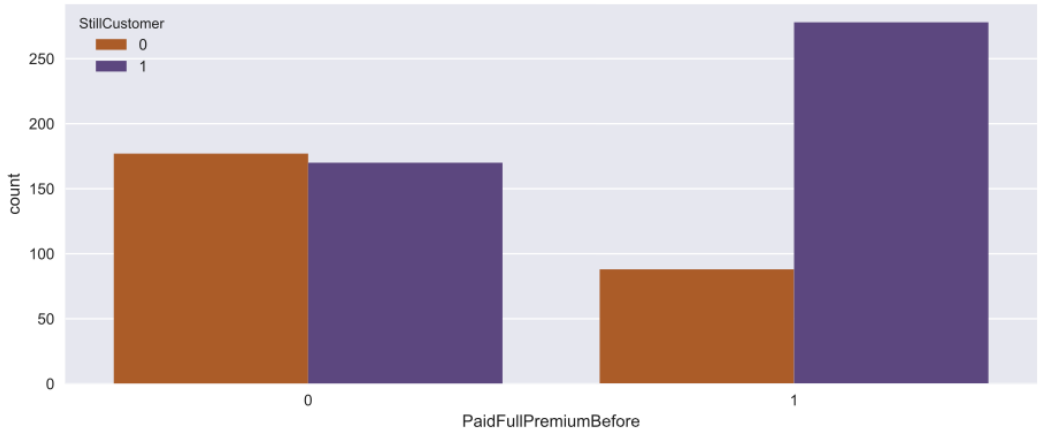
realized that it was not making a significant impact. Therefore, we elected to not balance the data as we did not want to make the model more complicated than it was necessary.

Figure 1: Proportionate of Customers



A feature that we had created was looking at whether during the life of the customer, it ever paid a preimum in full instead of financing it or paying it in installments. Since we did not have any socio-economic information about the customer, we wanted to derive any information that would be indicative of their economic standing. Figure 2 shows that there is a split among customers who are no longer active and whether they have ever paid full their premium. However, if we look at those who are still a customer, we see a large portion of these customers having paid their premiums in full at the least once during their lifetime at TGI.

Figure 2: Comparing Customers with having Paid Full Premium Before



Duration of a customer and total value derived from a customer are quite important when looking at ways to improve customer experience and ultimately increase revenue. Figure 3 shows us a Kernel Density Estimation (KDE) plot to estimate the Probability Density Function (PDF) of durations in months compared to whether the customer is still active. What is interesting is that there is an intersection between the two classes at approximately 40 months. It would require further analysis to gauge

whether that intersection exists because of the type of service that occured with the customer at that time.

Figure 3: KDE between Customers and Duration (months) of Customer

Another interesting KDE plot (Figure 4) was looking at the gender differences between the customers and the duration of their tenure with our client. There was no significant difference that we noticed between the gender as most of these decisions are made as part of the family when it comes to home, health and auto insurance. Regardless of gender, it seems to be a household decision, so we see negligible differences between the gender and the customer's duration.

Figure 4: KDE between Gender and Duration (months) of Customer

BASELINE: SUPERVISED LEARNING

Unlike other deep learning models that involve computer vision, time series analysis and such, our focus is a classification model with structured data. We wanted to run a baseline of various supervised learning models and determine whether our NN model performs better. We created dummy variables for various categorical features, such as, State, and transformed data to normality using Yeo-Johnson transformation.

Figure 5 shows the results from executing 15 different supervised learning models and their respective accuracy, AUC, Recall, Precision, F1 and Kappa metrics. In addition, each model used a stratified cross validation function with a K-fold of 10. The highest Accuracy (73.6%) and F1 (78.7%) score was associated with the Logistics Regression model.

Figure 5: Comparing 15 different supervised learning models using our dataset

Index	Model	Accuracy	AUC	Recall	Prec.	F1	Kappa
0	Logistic Regression	0.736	0.803	0.831	0.768	0.797	0.418
1	Extreme Gradient Boosting	0.728	0.803	0.816	0.769	0.790	0.403
2	Ridge Classifier	0.727	0.000	0.842	0.753	0.794	0.390
3	CatBoost Classifier	0.722	0.794	0.809	0.764	0.784	0.394
4	Linear Discriminant Analysis	0.722	0.798	0.816	0.760	0.785	0.391
5	Random Forest Classifier	0.721	0.773	0.767	0.784	0.774	0.408
6	Gradient Boosting Classifier	0.718	0.797	0.812	0.758	0.782	0.381
7	Ada Boost Classifier	0.712	0.776	0.802	0.757	0.777	0.368
8	K Neighbors Classifier	0.709	0.747	0.819	0.747	0.779	0.352
9	Light Gradient Boosting Machine	0.708	0.773	0.791	0.756	0.771	0.364
10	SVM - Linear Kernel	0.699	0.000	0.833	0.731	0.774	0.320
11	Extra Trees Classifier	0.693	0.736	0.772	0.749	0.759	0.335
12	Decision Tree Classifier	0.677	0.645	0.769	0.731	0.748	0.295
13	Naive Bayes	0.616	0.747	0.649	0.758	0.646	0.201
14	Quadratic Discriminant Analysis	0.598	0.661	0.689	0.733	0.634	0.132

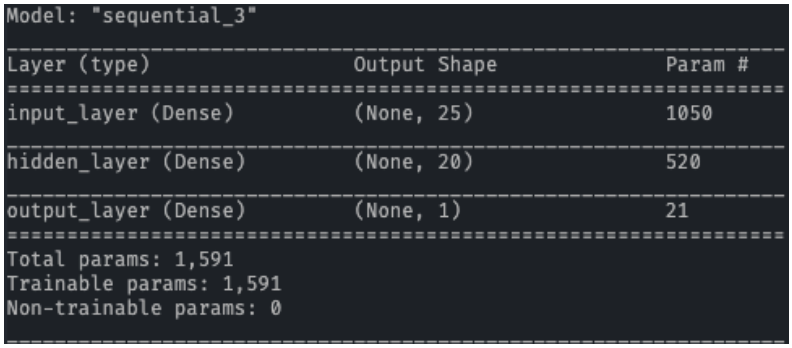
By executing and comparing these models, it provides a baseline for us to evaluate the accuracy of the single-layered NN model.

BASELINE: SINGLE-LAYER NUERAL NETWORK MODEL

We created a single-layer artificial neural network model. This model has an input layer with 41 inputs and 1,050 weights. It then outputs to 25 with 520 weights. The output layer is our response feature that has 21 weight parameters. The activation function for the hidden layer is Rectified Linear Unit (ReLU) and the optimizer we used is Adam. The ReLU activation function will output the input directly if is positive, otherwise, it will output zero. It has become a popular and often a default activation function for many types of neural networks. It is easier to train and often achieves better performance than other activation functions. In addition, ReLU activation function overcomes the vanishing gradient problem, allowing models to learn faster and perform better. The vanishing gradient problem refers to the gradient that will be vanishingly small and essentially prevent the weight from changing its value.

The Adaptive Moment Estimation (Adam) optimizer nearly always works faster and usually more reliable reaching a global minimum when minimizing the cost function in training neural nets. We do not change any default hyper parameters for the Adam optimizer (e.g.: learning rate, decay rate, etc.) in this single-layer model.

Figure 6: Single-layer Artificial Neural Network (ANN)



The accuracy of this single-layer ANN is 75%. This is a slight improvement from our previous models, even though from a deep learning perspective, this is a very simple model.

OPTIMIZE THE NUERAL NETWORK MODEL

Now that we have established our baseline for a NN model, we can tune our hyper-parameters to see whether we can achieve better results. Creating a model is always challenging as people are not generally good at optimizing many variables with so many calculations. This often becomes a trial and error approach. We can look at conducting a grid or random search across the variables (i.e.: number of hidden layers, dropout rate, learning rate, input layer nodes, batch size, activation functions, learning rate decay) but it may not always yield the best results and may take too long for the model to generalize well. In addition to the complexity of the model, we should consider how long it takes to train the model, though, given our dataset it will not be a deciding factor.

Brief Description of Terms:

1. Hidden layer(s) are between the input and output layers that perform the function of applying weights to the inputs and directing them through an activation function.
2. Learning rate controls the rate at which our model responds to each error when the weights are updated. A small learning rate may take too long for the model and a large learning rate value may provide suboptimal weights.
3. Dropout rate prevents overfitting and improves the overall generalization for the model. The common suggestion for a dropout rate is between 0 and 0.5.
4. A batch size controls the number of training samples the model goes through before the internal parameters of the model are updated.
5. Epoch often gets confused with batch size; however, epoch refers to the number of passes through the entire training dataset. We also employ a call back function to stop the training early. Training too many epochs can lead to overfitting of the training dataset and therefore, early stopping allows us to specify an arbitrary large number of training epochs and stop training once the model performance stops improving on the validation dataset.

6. Back-propagation is the essence of training a neural network model. The idea is to fine-tune the weights of the neural network based on the Loss (error rate) that is provided from the previous epoch. By tuning the weights after each epoch, the model tries to increase its generalization.
7. Activation functions determine the output of a neural network. The function is attached to each neuron of the network and determines whether it should be activated ("fired") based on each neuron's input that is relevant for the model's prediction. The activation functions also normalize the output of each neuron between 1 and 0 or between -1 and 1. There are several activation functions that we will evaluate:
    a. Exponential Linear Unit (ELU) tend to converge cost to zero faster and produce more accurate results. Different to other activation functions, ELU has an extra alpha constant that is a positive number.
    b. Sigmoid takes the real value as input and outputs a value between 0 and 1. It is non-linear, continuously differentiable, monotonic, and has a fixed output range.
    c. Rectified Linear Units (ReLU) is not linear and provides the same benefits as Sigmoid but with better performance.
    d. Tanh flattens a real-valued number to the range between -1 and 1. It is non-linear but unlike Sigmoid, its output is zero-centered.
    e. Scaled Exponential Linear Unit (SELU) has an internal normalization component and is faster than external normalization, which means the network converges faster. In addition, it does not have vanishing or exploding gradient problems.

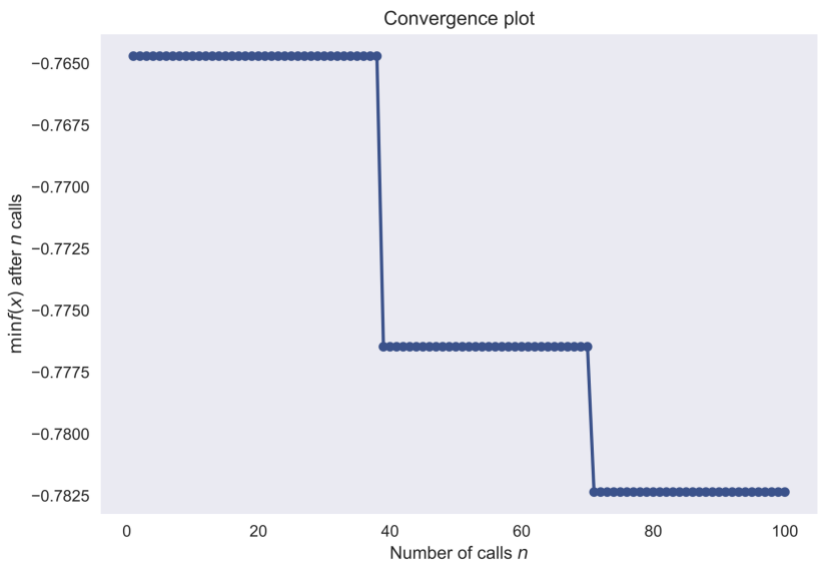
After conducting some research, I came across a more efficient way of hyper parameter tuning. Instead of using a random search or a grid search, we can leverage Bayesian optimization method which allows us to approximate the function using a Gaussian process. The notion is that the function values are assumed to follow a multivariate gaussian and the covariance of the function values are given by the Gaussian Process between the parameters. The next set of parameters to evaluate are selected by the acquisition function over the Gaussian prior. This is a much faster way to evaluate the parameters. We executed 100 iterations (n_calls) using the range of the following parameters

Parameters and Ranges with Bayesian Optimization using Gaussian Processes:

- Learning rate between 0.01 and 0.0001
- Number of Dense Layers between 1 and 5
- Number of Input Nodes between 1 and 512
- Number of Dense Nodes between 1 and 28
- Activation Functions of ReLU, Sigmoid, ELU, SELU and Tanh
- Batch size between 1 and 256
- Adam Decay rate between 0.1 and 0.000001
- Dropout rate between 0 and 0.5

Once the search was complete, we see that the convergence occurred around the 70th. In addition, the accuracy score does not improve after that. The convergence plot for the iterations is shown below and a more detailed plot for each hyper parameter is shown in the Appendix.

Figure 7: Convergence Plot



The top 5 models based on accuracy are listed below and a complete listing of the parameters and results are in the Appendix. What is quite interesting is that in my previous attempts to execute the code for the optimization parameters, I had perhaps 1 or 2 models that had the same accuracy score. However, in this case, our first 5 models all have the same accuracy score. It would require further analysis to determine how each observation is being classified and whether the observations are being classified differently between the models but still end up with the same accuracy. Aside from the accuracy score, the remaining parameters seem to be different for each of the models. The top 5 models had 2 different activation functions and for the most part, the remaining parameters were all different.

I selected the second model (index: 60), as the first model had an execution time that was much greater than the second model and it was not as complicated with one fewer layer.

Figure 8: Top 5 Optimized Models

index	learning rate	hidden layers	input layer nodes	hidden layer nodes	activation function	batch size	adam learning rate decay	dropout rate	accuracy	time (seconds)
99	0.010000	5	512	28	elu	1	0.007296	0.000000	78.235292	43.58
60	0.010000	4	481	13	elu	150	0.006926	0.042091	78.235292	3.37
29	0.003240	5	143	15	relu	19	0.001240	0.068106	78.235292	4.59
75	0.010000	1	445	13	relu	208	0.003714	0.000000	78.235292	1.65
10	0.007348	3	277	15	elu	76	0.003821	0.029607	78.235292	2.9

Our selected model (index: 60) is compiled with 4 hidden layers, a learning rate of 0.01, input layer nodes of 481, hidden layer nodes of 13, batch size of 150, learning rate decay of 0.0069, dropout rate of 0.04209 and an activation function of elu and is summarized in Figure 7.

Figure 9: Optimized Neural Network

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 481)               20202
_________________________________________________________________
dropout_1 (Dropout)          (None, 481)               0
_________________________________________________________________
layer_dense_1 (Dense)        (None, 13)                6266
_________________________________________________________________
dropout_2 (Dropout)          (None, 13)                0
_________________________________________________________________
layer_dense_2 (Dense)        (None, 13)                182
_________________________________________________________________
dropout_3 (Dropout)          (None, 13)                0
_________________________________________________________________
layer_dense_3 (Dense)        (None, 13)                182
_________________________________________________________________
dropout_4 (Dropout)          (None, 13)                0
_________________________________________________________________
layer_dense_4 (Dense)        (None, 13)                182
_________________________________________________________________
dropout_5 (Dropout)          (None, 13)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 14
=================================================================
Total params: 27,028
Trainable params: 27,028
Non-trainable params: 0
_________________________________________________________________
```

EXECUTE AND INTERPERET THE OPTIMIZED MODEL

After we had identified our optimized model, we then compiled it so we can execute it on the unseen holdout dataset. Our unseen holdout dataset consists of 36 (5%) of the original 713 observations. We executed the optimized model and employed an early stop mechanism to ensure the model did not over

train. If the model over trains, then it is likely to overfit. Therefore, we want to stop the model from over training when we see that it is not learning anything new (i.e.: loss rate is not improving).

Even though we set the epoch to 50, it did not take long for the model to stop training.

Figure 10: Loss Results by Epoch for Unseen Holdout Dataset

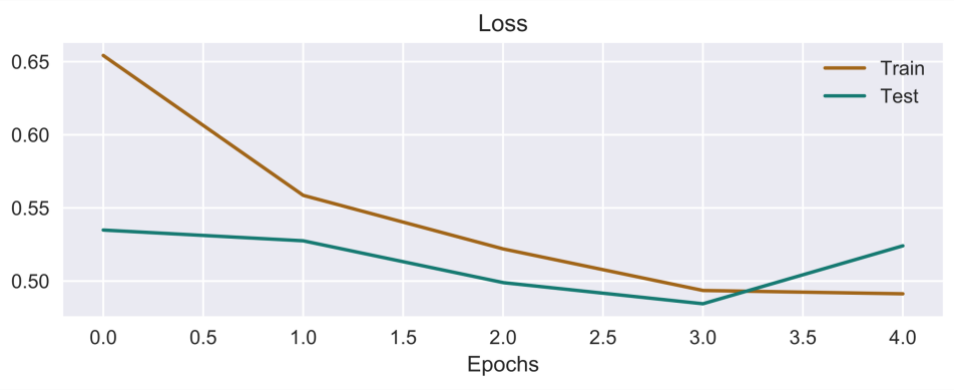
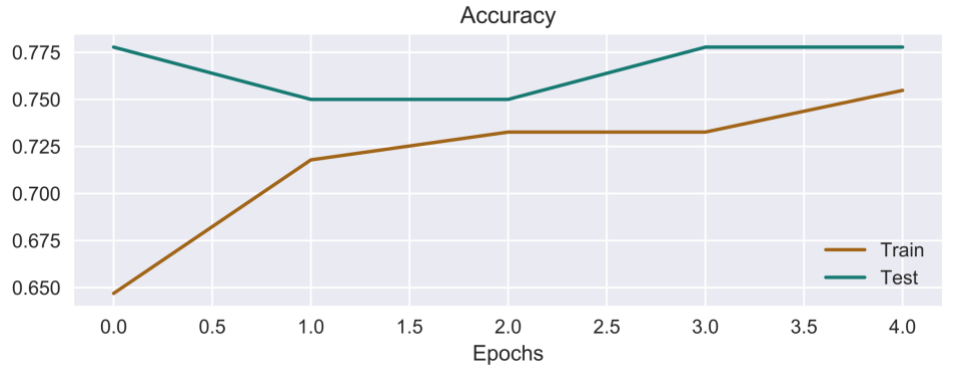


Figure 11: Accuracy Results by Epoch for Unseen Holdout Dataset



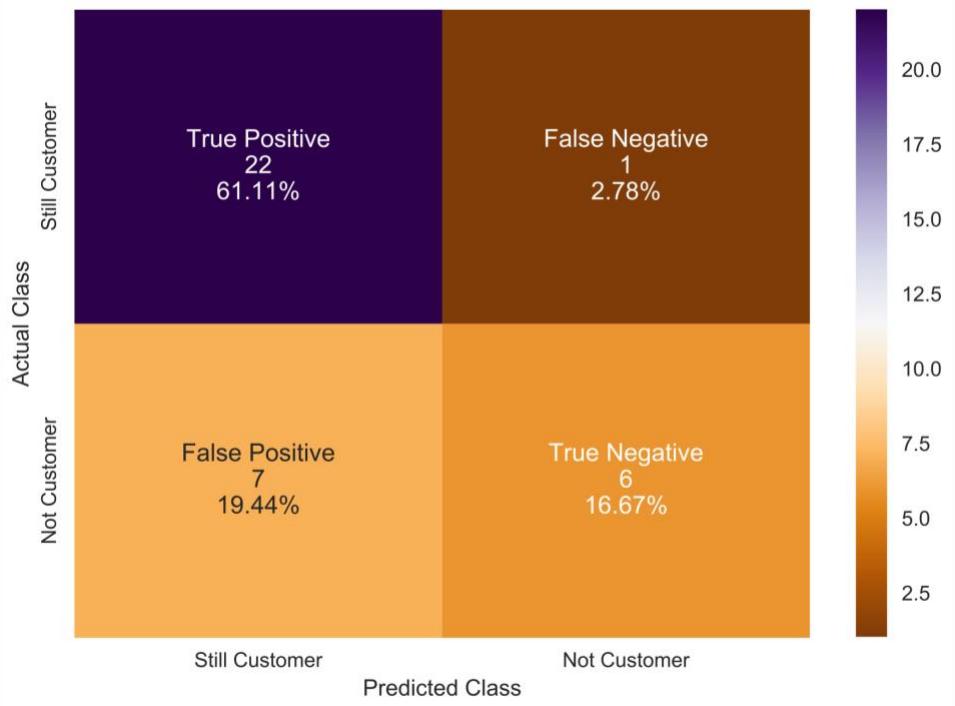
The accuracy results from this optimized model using the unseen holdout dataset is: 77.8%. Below is the summary of our accuracy results from the 3 models:

- Supervised Learning (Logistic Regression): 73.6%
- Baseline Single-Layered NN: 75%
- Bayesian Optimized Deep Learning NN: 77.8%

It is quite clear and substantial that our Bayesian optimized deep learning model is much better than the previous models and should be used on the client's dataset. It is interesting to note that for every observation we said that the customer is still at TGI (StillCustomer: 1) then we would be correct 67% of the time. However, since our model performed much better than 67%, it should be utilized to improve customer experience and retain customers for TGI.

Another metric that we should look at is the F1 score for our selected model. F1 score is the harmonic mean of Precision and Recall. It provides a better measure of the incorrectly classified cases than the accuracy metric. The F1 score for the optimized model 84.6% (Precision: 75.9% and Recall: 95.7%). The F1 score is much higher than our Logistic Regression model (79.7%). The figure below shows the confusion matrix of our results.

Figure 12: Confusion Matrix of Bayesian Optimized Deep Learning Model



What is quite apparent in the confusion matrix is the low precision (high number of False Positives). This means that our model predicted the customer is still with the company but that is actually not the case. This could mean that the customer left the company but could have stayed. Perhaps something during their tenure or a minor incident caused the customer to leave but it may not take much effort to bring the customer back. I wanted to pursue this further so I can Tom at TGI, the list of the clients from our unseen holdout dataset and the classification that our model predicted. I asked him to call them to get feedback, especially, those customers whose probability between either class (Still a Customer versus Not a Customer) was borderline.

POSITIVE IMPACT TO COMPANY REVENUE

After speaking with Tom, he agreed to call the 36 customers (see Appendix) to determine whether he can either ensure their commitment to stay by renewing their premiums at maturity or win them back as TGI's customers. Retaining existing customers is big but bringing the customer back to the company is even bigger! Optimizing and executing models are fun but unless it helps the company then playing with models is just fun. I wanted to get Tom's feedback as to whether the model was beneficial in identifying

which of the 36 clients he should focus on and provide a much more meaningful approach in keeping or bringing back his customers than calling which every customer screamed the loudest. He had agreed to call all 36 customers, but I asked him to focus on those customers that were on the border between either one of the classes and those which our model misclassified.

Below is a highlight of the customers he called and their reaction.

SUMMARY AND CONCLUSION

From the beginning of this project, we decided to take a novel approach for this class. There would have been easier approaches than to identify a client that would be willing to provide confidential data and then making sense of the data and executing various models to identify actionable tasks for our client. On top of all this, being able to share the results back with the client and receive feedback on how the model performed in the real-world and not just on my computer screen.

In my last call with Tom, we discussed how CEOs of companies will randomly call their customers to get feedback on how the company is doing and what can be done more effectively. After having the experience to call a small subset of the customer population, Tom tell me that "this model allows me to prioritize which customers I should call."

Tom provided the following feedback after calling 14 customers:

- Two customers who are no longer his customers said they would be interested in coming back but are price sensitive.
- Two customers who are no longer his customers are interested in being TGI's customers, but Tom would prefer they do not return because of high losses.
- One customer who is still a customer asked for a quote to another line of insurance products (automobile). This was a pleasant surprise and may provide future up-selling opportunities.

Overall, Tom's feedback was quite positive and he has asked whether I can focus on the commercial side of his insurance business as it generates over 80% of revenue for TGI. He wants to leverage this model to make targeted attempts at retaining and upselling to his existing customers and regaining those customers who are no longer his customers.

SOURCES

Learn: Learn: Machine learning in Python - scikit-learn 0.16.1 documentation. (n.d.). Retrieved June 11, 2020, from https://scikit-learn.org/

Home. (2020, May 11). Retrieved June 11, 2020, from https://pycaret.org/

TensorFlow. (n.d.). Retrieved June 12, 2020, from https://www.tensorflow.org/

Team, K. (n.d.). Simple. Flexible. Powerful. Retrieved June 12, 2020, from https://keras.io/

Optimize. (n.d.). Retrieved June 12, 2020, from https://scikit-optimize.github.io/stable/index.html

APPENDIX

A: Exploratory Data Analysis: Data

Dataset info

Number of variables	30
Number of observations	713
Missing cells	0 (0.0%)
Duplicate rows	0 (0.0%)
Total size in memory	167.2 KiB
Average record size in memory	240.2 B

Variables types

Numeric	15
Categorical	7
Boolean	2
Date	0
URL	0
Text (Unique)	0
Rejected	6
Unsupported	0

Accident/Health_(P)...
Categorical

Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0

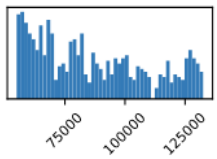
Builders_Risk_(P)_A...
Categorical

Distinct count	3
Unique (%)	0.4%
Missing (%)	0.0%
Missing (n)	0

Customer_ID
Numeric

Distinct count	713
Unique (%)	100.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

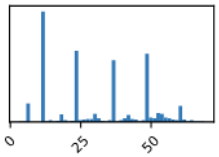
Mean	86390.00421
Minimum	54928
Maximum	132548
Zeros (%)	0.0%



DurationAsCust
Numeric

Distinct count	60
Unique (%)	8.4%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

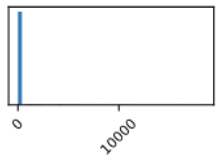
Mean	30.89481066
Minimum	3
Maximum	69
Zeros (%)	0.0%



Dwelling_Fire_Amo...
Numeric

Distinct count	47
Unique (%)	6.6%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

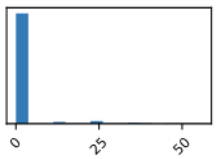
Mean	281.9679804
Minimum	0
Maximum	18478.92
Zeros (%)	93.4%



Dwelling_Fire_Durat...
Numeric

Distinct count	15
Unique (%)	2.1%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	1.786816269
Minimum	0
Maximum	56
Zeros (%)	93.4%

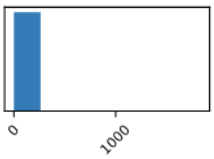


Earthquake_(P)_Am...
Categorical

Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0

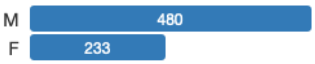
Flood_Amount
Numeric

Distinct count	7
Unique (%)	1.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	10.52920056
Minimum	0
Maximum	1832
Zeros (%)	99.2%



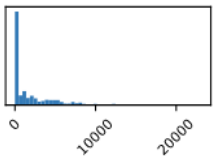
Gender
Categorical

Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0



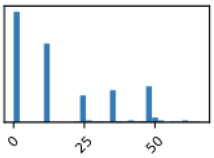
Homeowners_Amount
Numeric

Distinct count	431
Unique (%)	60.4%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	2025.001515
Minimum	0
Maximum	23141.95
Zeros (%)	36.2%



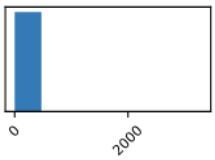
Homeowners_Durat...
Numeric

Distinct count	31
Unique (%)	4.3%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	17.71248247
Minimum	0
Maximum	66
Zeros (%)	36.3%



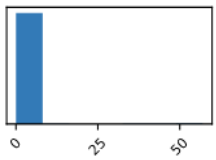
Life_(P)_Amount
Numeric

Distinct count	7
Unique (%)	1.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	7.332847125
Minimum	0
Maximum	3295
Zeros (%)	99.2%



Life_(P)_Duration
Numeric

Distinct count	7
Unique (%)	1.0%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	0.3646563815
Minimum	0
Maximum	57
Zeros (%)	99.2%

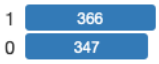


Motorcycle_Amount
Categorical

Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0

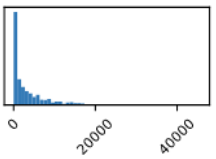
PaidFullPremiumBe...
Boolean

Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0



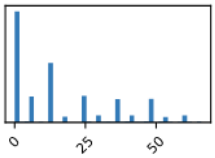
Private_Passenger_...
Numeric

Distinct count	460
Unique (%)	64.5%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	3829.242342
Minimum	0
Maximum	45661.82
Zeros (%)	35.3%



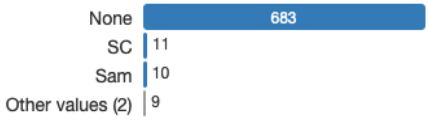
Private_Passenger_...
Numeric

Distinct count	39
Unique (%)	5.5%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0
Mean	16.70967742
Minimum	0
Maximum	66
Zeros (%)	35.3%



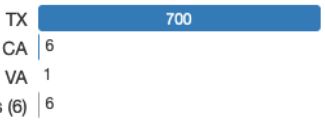
Referrer
Categorical

Distinct count	5
Unique (%)	0.7%
Missing (%)	0.0%
Missing (n)	0



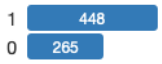
State
Categorical

Distinct count	9
Unique (%)	1.3%
Missing (%)	0.0%
Missing (n)	0



StillCustomer
Boolean

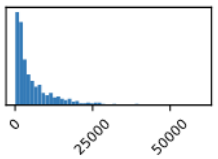
Distinct count	2
Unique (%)	0.3%
Missing (%)	0.0%
Missing (n)	0



Total_Amount
Numeric

Distinct count	695
Unique (%)	97.5%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

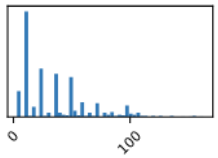
Mean	6171.380659
Minimum	81
Maximum	60330.92
Zeros (%)	0.0%



Total_Duration
Numeric

Distinct count	78
Unique (%)	10.9%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

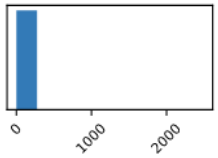
Mean	37.20757363
Minimum	3
Maximum	163
Zeros (%)	0.0%



Umbrella_(P)_Amount
Numeric

Distinct count	9
Unique (%)	1.3%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

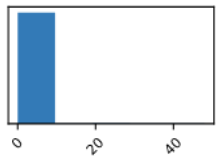
Mean	12.085554
Minimum	0
Maximum	2490
Zeros (%)	98.9%



Umbrella_(P)_Durati...
Numeric

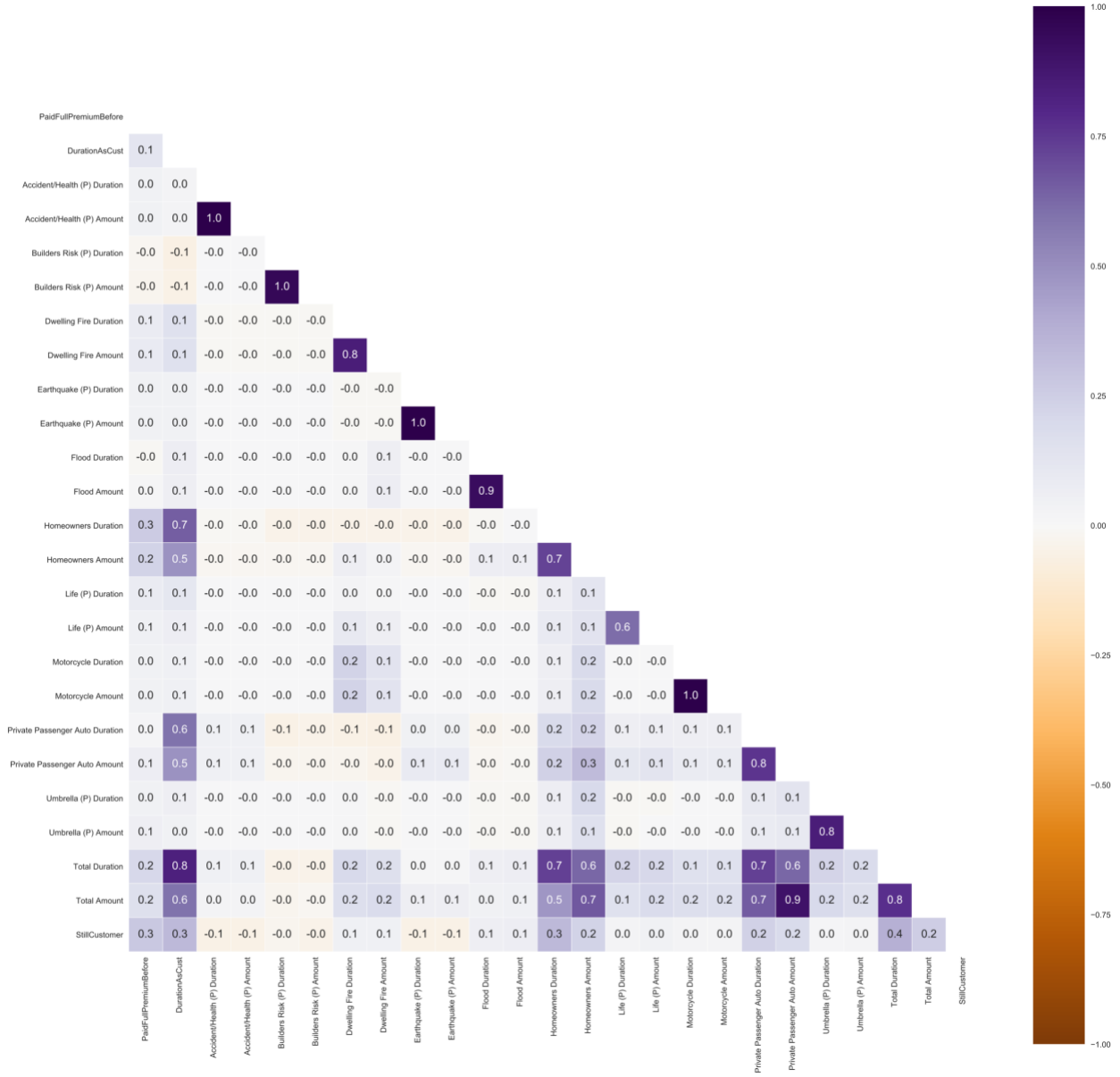
Distinct count	5
Unique (%)	0.7%
Missing (%)	0.0%
Missing (n)	0
Infinite (%)	0.0%
Infinite (n)	0

Mean	0.3197755961
Minimum	0
Maximum	48
Zeros (%)	98.9%

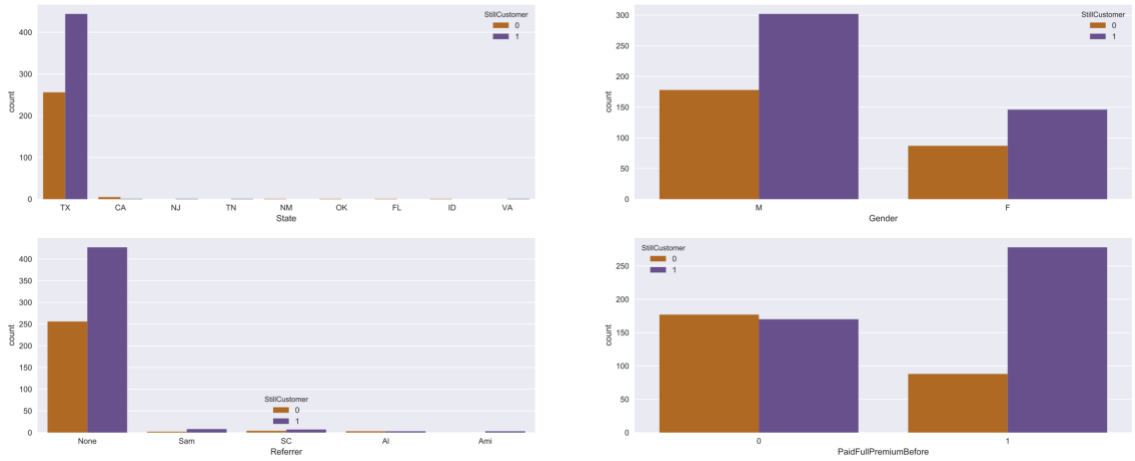


B: Exploratory Data Analysis: Visuals

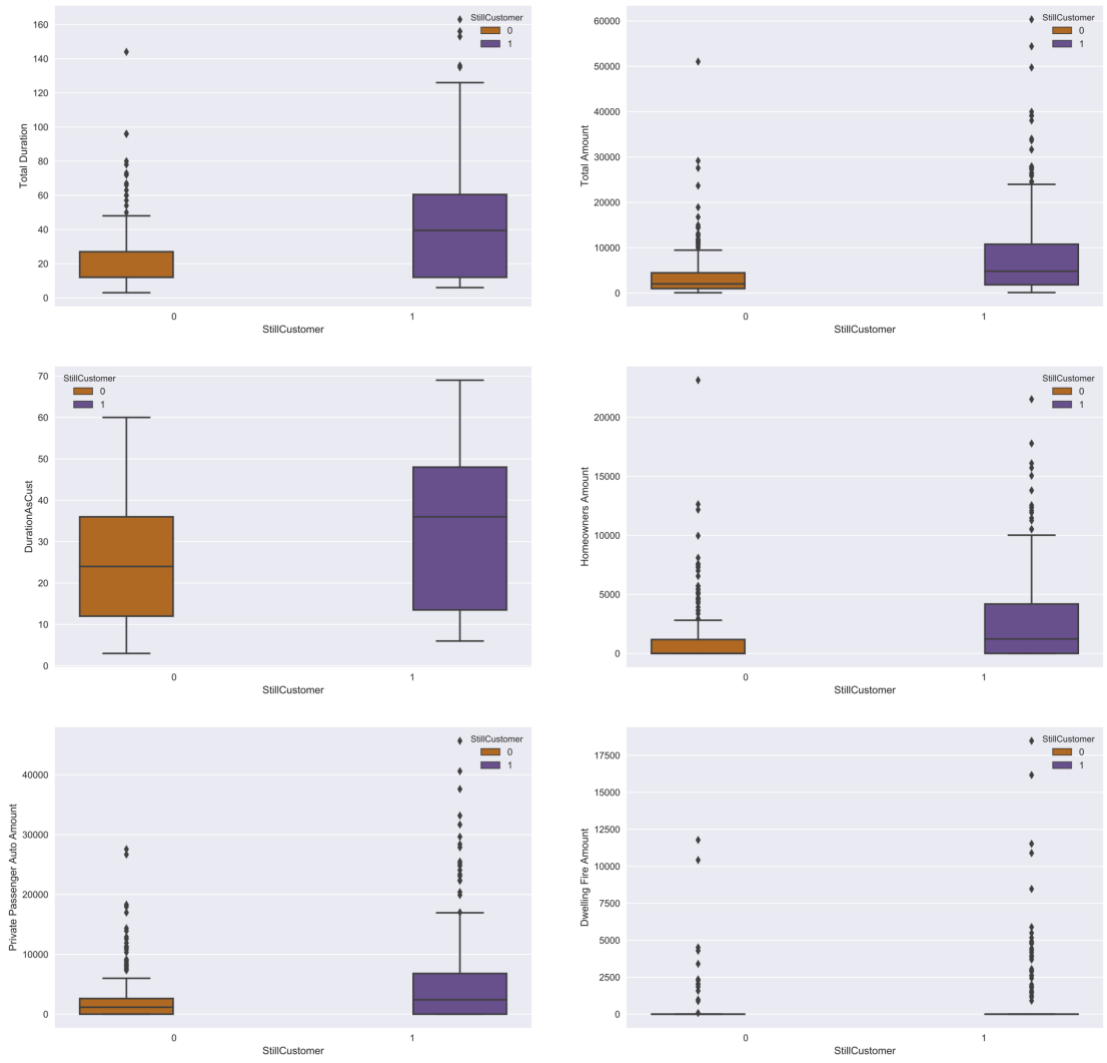
Correlation Matrix of the Dataset



Plots of Categorical Features (State, Gender, Referrer and PaidFullPremiumBefore)



Box Plots of Numeric Features (Total Duration, Total Amount, DurationAsCust and Homeowner Amount)



KDE Plot of StillCustomer and Amount

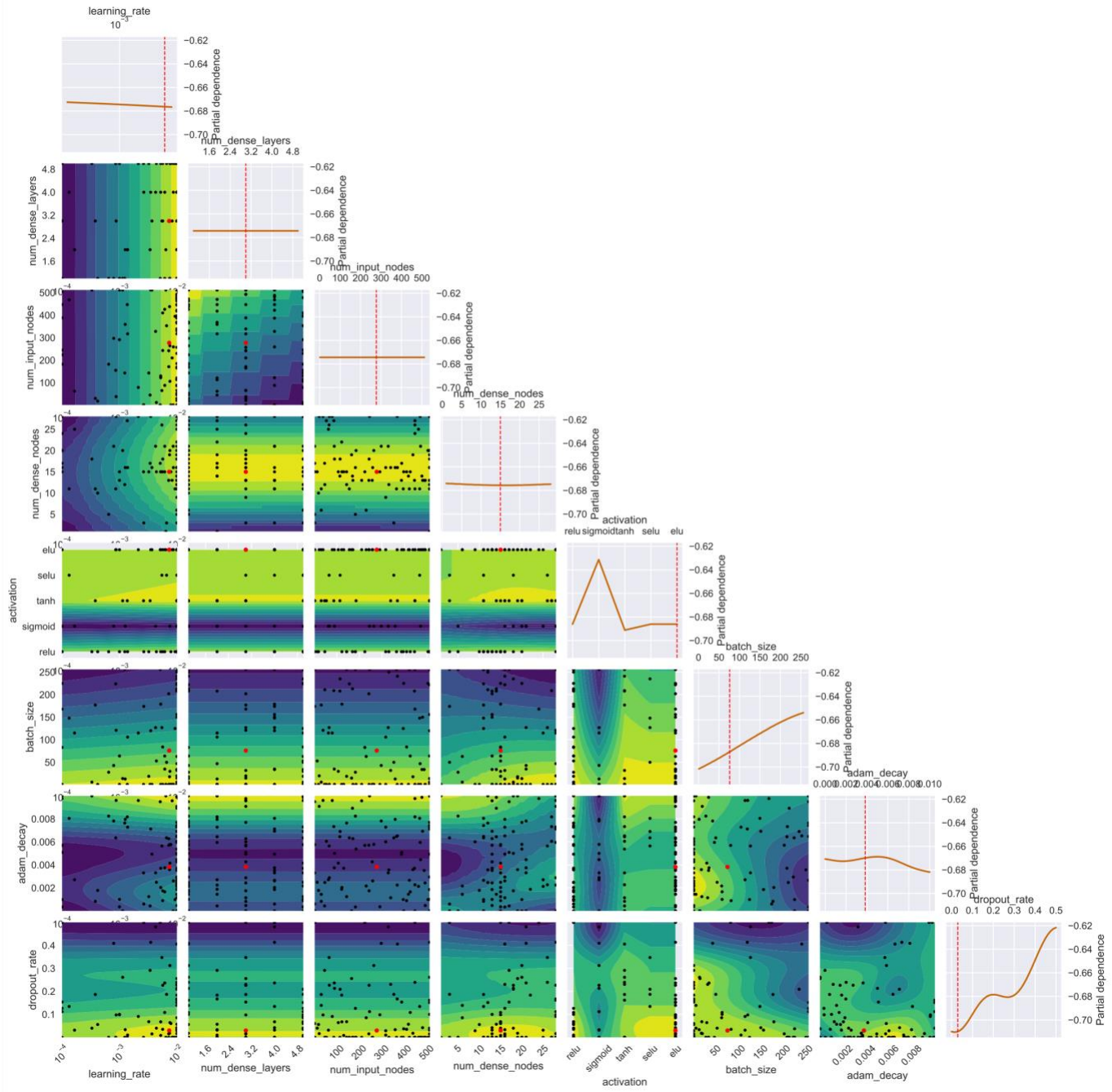
C: Bayesian Optimization Search Results (n_call = 100)

index	learning rate	hidden layers	input layer nodes	hidden layer nodes	activation function	batch size	adam learning rate decay	dropout rate	accuracy	time (seconds)
99	0.010000	5	512	28	elu	1	0.007296	0.000000	78.235292	43.58
60	0.010000	4	481	13	elu	150	0.006926	0.042091	78.235292	3.37
29	0.003240	5	143	15	relu	19	0.001240	0.068106	78.235292	4.59
75	0.010000	1	445	13	relu	208	0.003714	0.000000	78.235292	1.65
10	0.007348	3	277	15	elu	76	0.003821	0.029607	78.235292	2.9
81	0.007222	3	171	13	elu	1	0.006281	0.000000	78.235292	87.81
24	0.010000	5	397	15	elu	225	0.000001	0.037377	78.235292	3.98
92	0.002129	5	79	28	elu	115	0.009600	0.090900	78.235292	4.03
35	0.004392	5	12	15	relu	53	0.002489	0.040072	78.235292	4.07
80	0.010000	4	447	11	elu	256	0.000001	0.000000	77.647060	2.1
47	0.001236	1	143	12	relu	256	0.007640	0.000000	77.647060	2.81
72	0.010000	1	53	27	tanh	1	0.006051	0.000000	77.647060	33.28
16	0.010000	3	512	20	elu	1	0.007697	0.000000	77.647060	36.2
39	0.003678	3	512	16	elu	1	0.004938	0.015060	77.058822	29.02
43	0.000860	5	155	11	elu	224	0.006355	0.000000	77.058822	5.46
25	0.008683	4	259	16	elu	1	0.002608	0.000000	77.058822	26.72
20	0.010000	3	1	3	tanh	256	0.000001	0.000000	77.058822	4.15
32	0.010000	3	339	16	tanh	32	0.001143	0.230385	77.058822	2.87
88	0.010000	3	21	28	elu	20	0.009087	0.016378	77.058822	2.91
89	0.010000	2	121	22	elu	1	0.001881	0.183693	77.058822	9.09
97	0.010000	1	447	26	tanh	1	0.002272	0.239294	77.058822	9.17
57	0.010000	1	512	11	relu	256	0.007423	0.000000	76.470590	1.45
54	0.007209	3	242	13	elu	202	0.005712	0.015449	76.470590	3.3
67	0.010000	1	365	11	relu	256	0.006472	0.000000	76.470590	1.34
74	0.010000	1	512	28	tanh	1	0.010000	0.161682	76.470590	9.92
84	0.008268	3	210	13	elu	1	0.009574	0.000000	76.470590	26.64
91	0.010000	1	30	6	elu	6	0.001440	0.138962	76.470590	2.45
70	0.010000	4	387	13	relu	1	0.004932	0.000000	76.470590	25.43
0	0.001000	1	512	13	relu	64	0.001000	0.100000	76.470590	2.77
53	0.003415	2	45	17	tanh	1	0.000433	0.291390	75.882351	8.19
52	0.007554	1	264	20	sigmoid	1	0.005786	0.312361	75.882351	20.05
98	0.005665	3	243	18	elu	1	0.001748	0.226518	75.882351	14.75
62	0.001367	2	447	27	tanh	1	0.008179	0.000000	75.882351	97.42
94	0.010000	2	368	28	elu	1	0.002247	0.166889	75.882351	16.14
4	0.006148	4	88	21	elu	35	0.003747	0.236619	75.882351	3.87
36	0.006134	1	426	15	relu	64	0.001663	0.108286	75.882351	2.23
21	0.010000	3	317	9	elu	1	0.006005	0.028686	75.882351	19.6
38	0.001136	1	512	20	tanh	14	0.000001	0.261287	75.882351	3.59
86	0.010000	2	470	28	elu	47	0.005954	0.000000	75.882351	3.31
77	0.010000	3	257	13	elu	1	0.004268	0.000000	75.882351	25.93

73	0.010000	1	86	27	tanh	209	0.007330	0.000000	75.882351	2.84
69	0.010000	5	512	12	elu	256	0.010000	0.151501	75.294119	4.03
65	0.010000	4	488	13	elu	34	0.003518	0.000000	75.294119	2.37
68	0.010000	3	492	13	elu	1	0.002433	0.000000	75.294119	10.28
78	0.010000	1	46	12	relu	211	0.002471	0.000000	75.294119	1.45
55	0.010000	1	185	20	tanh	1	0.005378	0.000000	75.294119	14.02
87	0.010000	3	66	28	sigmoid	1	0.002881	0.226700	75.294119	11.29
31	0.010000	2	113	14	tanh	208	0.006449	0.000000	75.294119	1.52
82	0.010000	4	422	11	elu	256	0.002659	0.000000	74.705881	2.32
63	0.010000	1	463	25	elu	1	0.002131	0.000000	74.705881	16.32
95	0.010000	1	89	18	sigmoid	1	0.005002	0.213116	74.705881	20.81
17	0.010000	1	512	16	elu	222	0.002948	0.000000	74.705881	1.71
58	0.010000	3	165	19	elu	15	0.000001	0.255806	74.705881	3.54
64	0.010000	1	472	21	sigmoid	1	0.004455	0.000000	74.705881	64.27
50	0.001385	2	317	16	relu	106	0.007282	0.010000	74.705881	2.62
42	0.007567	5	106	15	elu	30	0.004155	0.064097	74.705881	4.75
37	0.000331	5	512	18	relu	6	0.010000	0.039929	74.705881	42.4
46	0.005223	5	130	15	elu	1	0.003708	0.008415	74.705881	7.79
33	0.000999	5	291	15	elu	1	0.000357	0.098302	74.117649	12.12
40	0.010000	2	418	14	elu	1	0.001444	0.057080	74.117649	6.28
5	0.000133	4	468	11	selu	148	0.008048	0.134381	74.117649	6.46
61	0.010000	4	512	13	elu	156	0.001860	0.031572	74.117649	3.11
76	0.010000	1	169	14	relu	213	0.008927	0.000000	74.117649	1.31
30	0.010000	5	478	14	elu	34	0.009874	0.000000	74.117649	2.61
56	0.010000	1	512	11	tanh	256	0.000001	0.000000	73.529410	1.55
66	0.007451	1	438	19	elu	18	0.003591	0.018033	73.529410	2.13
23	0.000648	1	182	5	relu	28	0.008244	0.000000	72.941178	5.6
48	0.005229	4	122	15	relu	179	0.002036	0.067612	72.941178	4.08
71	0.010000	2	512	19	tanh	1	0.010000	0.000000	72.941178	8.88
59	0.007869	5	1	15	relu	1	0.005768	0.060109	71.764708	10.55
51	0.000417	1	18	10	relu	1	0.001784	0.055935	71.764708	66.65
49	0.005651	5	410	15	relu	84	0.000891	0.034974	71.176469	3.39
19	0.000100	3	388	17	elu	1	0.005089	0.153600	70.588237	135.79
45	0.010000	1	369	16	tanh	241	0.006528	0.405735	69.411767	2.07
9	0.001259	2	359	16	sigmoid	13	0.007962	0.033381	69.411767	14.31
3	0.000379	3	30	11	tanh	177	0.000833	0.186489	69.411767	5.62
96	0.010000	5	230	6	elu	63	0.002167	0.085303	68.823528	3.75
1	0.000165	2	62	25	sigmoid	115	0.002711	0.031304	68.823528	5.34
18	0.000100	5	505	20	relu	225	0.000984	0.000000	68.235296	6.29

28	0.009277	5	395	11	relu	125	0.000984	0.000000	68.235296	2.67
6	0.000793	1	297	21	sigmoid	43	0.007821	0.408936	67.647058	5.82
13	0.005126	1	1	17	elu	256	0.009495	0.500000	67.058825	3.36
12	0.000100	3	447	24	elu	168	0.009769	0.500000	67.058825	5.95
8	0.005477	3	65	26	selu	127	0.005644	0.347275	65.882355	4.01
93	0.000875	3	38	28	elu	121	0.009610	0.097560	65.294117	2.72
15	0.000100	1	224	15	relu	83	0.005696	0.500000	63.529414	4.83
27	0.003499	4	512	28	elu	120	0.007124	0.413416	63.529414	2.81
34	0.010000	5	114	18	selu	235	0.006906	0.209167	62.941176	4.93
7	0.006611	5	353	3	selu	50	0.006352	0.187104	62.941176	5.72
79	0.010000	1	331	12	relu	250	0.003573	0.000000	62.352943	1.16
14	0.002651	4	1	3	relu	174	0.000599	0.500000	61.176473	5.93
83	0.010000	1	1	5	sigmoid	256	0.000001	0.000000	61.176473	1.33
11	0.006419	2	512	1	elu	1	0.010000	0.000000	61.176473	32.71
44	0.002623	5	1	15	tanh	1	0.003303	0.000000	60.000002	77.24
41	0.000100	5	182	16	relu	256	0.000340	0.126159	58.823532	7.46
85	0.010000	3	493	13	elu	1	0.000469	0.000000	54.705882	18.75
26	0.008612	5	1	21	elu	1	0.000001	0.000000	53.529412	8.44
22	0.010000	3	512	28	relu	1	0.000001	0.000000	45.294118	7.42
2	0.004257	4	328	21	sigmoid	127	0.003310	0.480414	38.823530	3.5
90	0.000100	5	245	28	sigmoid	256	0.010000	0.111936	38.823530	8.56

D: Bayesian Optimization Search (pairwise dependence plot of the objective function)



E: Probability and Class for Customers in Unseen Holdout Dataset

Note: the numbers in Customer ID have been redacted for privacy purposes.

StillCustomer_x	predicted_prob	Customer ID	predicted_class	cf_matrix
0	9%	[redacted]	0	TN
	22%	[redacted]	0	TN
	29%	[redacted]	0	TN
	29%	[redacted]	0	TN
	39%	[redacted]	0	TN
	43%	[redacted]	0	TN
	53%	[redacted]	1	FP
	63%	[redacted]	1	FP
	76%	[redacted]	1	FP
	76%	[redacted]	1	FP
	81%	[redacted]	1	FP
	90%	[redacted]	1	FP
	95%	[redacted]	1	FP
1	48%	[redacted]	0	FN
	53%	[redacted]	1	TP
	61%	[redacted]	1	TP
	74%	[redacted]	1	TP
	74%	[redacted]	1	TP
	76%	[redacted]	1	TP
	76%	[redacted]	1	TP
	86%	[redacted]	1	TP
	87%	[redacted]	1	TP
	89%	[redacted]	1	TP
	89%	[redacted]	1	TP
	90%	[redacted]	1	TP
	91%	[redacted]	1	TP
	91%	[redacted]	1	TP
	91%	[redacted]	1	TP
	92%	[redacted]	1	TP
	93%	[redacted]	1	TP
	95%	[redacted]	1	TP
	95%	[redacted]	1	TP
	95%	[redacted]	1	TP
	95%	[redacted]	1	TP
	95%	[redacted]	1	TP
	96%	[redacted]	1	TP