Predict the Likelihood of Responding to Direct Mail Campaign in Consumer Lending Industry

Jincheng Cao, SCPD
Jincheng@stanford.edu

1. INTRODUCTION

When running a direct mail campaign, it's common practice for lending institutions to use predictive models to rank order perspective consumers based on their likelihood to respond and factor their likelihood to respond into final decision as whom to mail. These predictive models are so-called response model and the lending industry have been largely relied on Logistic Regression to build these models. Several reasons that make Logistic Regression a good modeling choice include: first, easy interpretation as required by financial industry regulations; second, these model does not need to be refreshed frequently; third, size of training data is small. As the changes of dynamics in consumer lending environment, such as loosen regulations, and huge training dataset, it's worthwhile to explore more modeling techniques that could meet those emerging requirements.

In this project, several other modeling techniques, including Gradient Boosting Trees, Support Vector Machine, and Neural Networks are going to be benchmarked against the Logistic Regression, which serves as the baseline. Each input to these models represents a perspective consumer with thousands of features describing almost all aspects of the consumer's credit profile, such as:

- Number of credit cards/auto loans/mortgage loans
- Total balance/total credit line of all accounts
- Utilization of all credit cards
- Number of late payments
- Number of charge-offs
- Credit Score

The output of these models could be either 0/1 with 1 representing the model predicts a consumer will respond to a mail campaign, or probability representing the likelihood of a consumer to respond a direct mail campaign.

2. RALATED WORK

Response model, as an effective marketing tool, is widely used in many industries besides consumer lending, such as online Ads companies and consumer goods companies. Because of its wide usage, many works have been done to establish methodologies to build response models. These methodologies could be divided into three groups. The recency, frequency, monetary (RFM) models[1,2], the machine learning models including logistic regression, tree based methods, SVM and neural networks[3,4], and optimization based methods[5].

The RFM models are essentially a simple descriptive segmentation scheme, which segment consumers based on their past behaviors, such as when an how frequent a consumer last time responded to a marketing offer, and how much value a consumer generated by responding previously. RFM is very easy to interpret, but its accuracy drops when dimension of the dataset becomes large. Optimization based methods divide perspective consumers into small segments in high dimensional space, and then pick segments with the highest response rates. The setup of the method is straightforward. The drawback is that it suffers from high dimensionality as well as prone to overfitting. The machine learning approaches are still the most reliable in terms of prediction accuracy as well as being able to handle large feature space while avoid overfitting.

3. DATASET, PREPROCESSING and EXPORATION

3.1 DATASET

The dataset that being used to evaluate these modeling techniques is Springleaf Marketing response dataset available on Kaggle[6]. The dataset contains 145,231 anonymized marketing records with each having 1932 features and one label indicating whether a consumer responded or not to mail campaigns. All feature names have been masked by the dataset provider on purpose. Among 1932 features, 1881 are numerical and 51 are non-numerical including categorical features and character features.

3.2 PREPROCESSING

1881 numerical features contain small percentage of missing values and they are imputed using feature means. Then, numerical features are standardized with respect to the entire dataset. It means the dataset is standardized before splitting into training and testing datasets later on, rather than splitting first and then standardize. The motivation of standardization first is (a) it's fair to assume that the population on which the model trained is reasonable representative and there is no substantial deviation from future population; (b) when being applied online for real time scoring, standardize new instance using old population mean and standard deviation is the only way; (c) computational faster, for some models are going to be trained iteratively on samples of increasing sizes.

Among those 51 categorical and character features, 34 are dropped because they are inappropriate to be included from business or regulation standpoints; and these 34 features includes date/timestamp, city, state, occupation, social security number and features that only have one unique value other than missing. The rest 17 features are converted into binary attributes using one-hot encoding, and the converted binary features are not standardized.

After preprocessing, the dataset ends up with 1996 numerical features, all of which are either standardized or binaries. The dataset is randomly split into 70% as training set and 30% as testing set.

3.3 EXPORATION

The overall response rate is 23.25%. Considering the ratio between 1 and 0 is roughly 1:4, down-sampling is not necessary. Mutual information between features and label is measured, and there are roughly 1800 features carry non-zero information with respect to the label. But mutual information is not used to reduce number of features, as it only measures unconditional relationship while some of the modeling techniques carry interaction terms explicitly or inexplicitly and unconditional independent variables can become dependent when conditioned on other variables.

4. METHODS

4.1 LOGISTIC REGRESSION

Logistic Regression is a supervised learning model that tries to classify a given instance into one of two classes. The training procedure of Logistic Regression is to minimize the following loss function:

$$J(\theta) = \frac{1}{m}\sum_{i=1}^{m} \log(1 + e^{-y^{(i)}\theta^T x^{(i)}})$$

The commonly used optimization algorithm to find the optimal parameters is gradient descent with the following update rule:

$$\theta_j \coloneqq \theta_j + \alpha \cdot (y^{(i)} - h_\theta(x^{(i)})) \cdot x_j^{(i)}$$

The output of Logistic Regression is between 0 and 1 after transformation through Sigmoid function:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-\theta^T x}}$$

In this project, we use *Linear_Model.LogisticRegression* class from Scikit-Learn to train Logistic Regression.

4.2 GRADIENT BOOSTING TREES[7]

Gradient Boosting Trees (GBT) is a binary classifier that ensembles many shallow trees of the same depth. It's an additive model that has the following form:

$$F(x) = \sum_{i=1}^{n} \gamma_n h_n(x)$$

where $h_m(x)$ represents individual shallow tree, and γ_n is the weight parameter that controls the contribution of each individual tree to the overall model.

The training procedure of GBT involves a series of optimization which tries to find the optimal $h_m(x)$ at each iteration. The optimization problem could be formularized by the following form:

$$F_n(x) = F_{n-1}(x) + arg\min_h \sum_{i=1}^{m} L(y_i, F_{n-1}(x_i) + h(x))$$

where L is the chosen loss function. Another way to interpret how GBT works is for each iteration, the optimization algorithm tries to find the optimal tree that best fits the residual of current GBT and then update the GBT by incorporating the new optimal tree. We use *ensemble.GradientBoostingClassifier* class from Scikit-Learn to train GBT.

4.3 SUPPORT VECTOR MACHINE

Different from Logistic Regression and Gradient Boosting Trees which output probability, support vector machine (SVM) is a classification model that outputs 0/1 labels. Intuitively, SVM tries to find a hyperplane that separates the positive and negative instances. The training procedure is to solve the following optimization problem:

$$\min_{w,b} \frac{1}{2}\|w\|^2$$

$$\text{s.t. } y^{(i)}\left(w^T x^{(i)} + b\right) \geq 1, i = 1, \dots, m$$

This optimization problem can be solved using sequential minimal optimization (SMO) algorithm.

Because the optimization problem listed above could be expressed with inner products of input vectors, kernel tricks could be applied to SVM. Kernels tricks allow SVM to construct the separating hyperplane in higher dimensional space without explicitly performing computation in higher dimensional space. Kernel tricks are powerful because instances not separateable in lower dimensional space might become separateable in higher dimensional space. We use *SVM.SVC* class from Scikit-Learn to train SVM.

4.4 NEURAL NETWORK

Neural network is a supervised learning algorithm that processes input information through layers of neurons. Each neuron absorbs all output information from previous layer and makes its prediction which feeds into the next layer. The architecture shown in Figure. 1 represents a two-layer neural

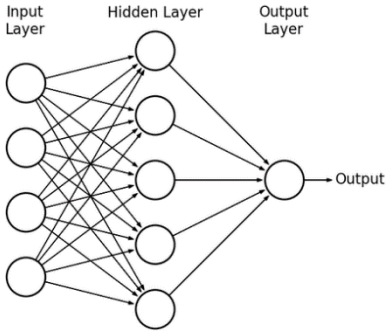

Figure. 1. Two layer neural network

network with one hidden layer and one output layer. Each neuron in hidden layer and output layer is a Sigmoid function:

$$h_\theta(x) = g(\theta^T x) = \frac{1}{1 + e^{-W^T x + b}}$$

The neural network is trained using batch gradient descent algorithm through forward- and backward-propagation. The pseudo-code is:

0. Initialize $W^{(l)}$, $b^{(l)}$

While not converged {

For each batch {

1. *Use backpropagation to compute* $\nabla_{W^{(l)}} J(W, b; x, y)$ *and* $\nabla_{b^{(l)}} J(W, b; x, y)$
2. *Update* $W^{(l)} := W^{(l)} - \alpha \cdot \nabla_{W^{(l)}} J(W, b; x, y)$
3. *Update* $b^{(l)} := b^{(l)} - \alpha \cdot \nabla_{b^{(l)}} J(W, b; x, y)$

}

}

We use *Numpy* to implement a two-layer neural network. We also add L-2 penalty term to the gradient descent to prevent from overfitting.

5. RESULTS

5.1 EVALUATIN METRICS

We use AUC of ROC and Log-Loss to measure model performance. AUC is the area under the ROC curve, which plots the true positive rate against false positive rate for varying thresholds. The interpretation of AUC is likelihood of the classifier scoring a positive instance higher than a negative instance. Log-Loss is a measure of distance between predicted class versus actual class of an instance. Its formula is:

$$Log\ loss = -(y \cdot \log(p) + (1 - y) \cdot \log(1 - p))$$

where p is the predicted probability of the instance being positive.

5.2 LOGISTIC REGRESSION

The training and testing sets have 101,661 and 43,570 samples, respectively. The Logistic Regression is trained over a series of smaller training sets which are randomly drawn from those 101,661 samples, and sizes of the series of smaller training sets range from 5,000 to 100,000 with 5,000 increment. Each

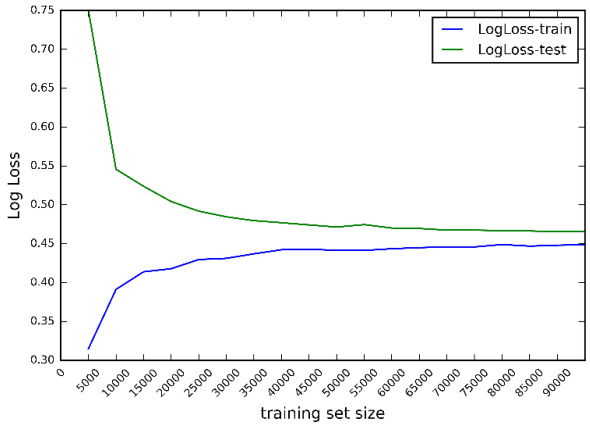

Figure. 2. Logistic Regression Learning Curve

model's performance is evaluated on both of the dataset it's trained and the testing set of size 43,570. The goal is not only to establish a benchmark as how well Logistic Regression model performs, but also to estimate how large sample Logistic Regression needs to reach the optimal performance.

As shown in Fig.2 and Table. 1, Logistic Regression model overfits the training dataset and also suffers from high bias when trained on datasets of size below 40,000. It needs roughly 70,000 training samples to obtain optimal performance – which generates an AUC of 0.761 and a Log-Loss of 0.466.

Training Set Size	AUC-Train	AUC-Test	LogLoss-Train	LogLoss-Test
5,000	0.904	0.654	0.315	0.750
10,000	0.841	0.707	0.391	0.546
15,000	0.818	0.714	0.414	0.524
20,000	0.811	0.728	0.418	0.504
25,000	0.802	0.736	0.430	0.492
30,000	0.799	0.744	0.431	0.485
35,000	0.792	0.747	0.437	0.480
40,000	0.789	0.750	0.442	0.477
45,000	0.787	0.751	0.443	0.474
50,000	0.786	0.755	0.442	0.471
55,000	0.784	0.753	0.441	0.475
60,000	0.784	0.756	0.444	0.470
65,000	0.782	0.757	0.445	0.470
70,000	0.780	0.759	0.446	0.468
75,000	0.782	0.760	0.446	0.467
80,000	0.779	0.759	0.449	0.467
85,000	0.780	0.760	0.447	0.466
90,000	0.778	0.761	0.448	0.465
95,000	0.777	0.761	0.449	0.466
100,000	0.778	0.762	0.449	0.465

Table. 1. AUC and Log loss on training and testing datasets, for Logistic Regression models trained on vary size of samples

Figure. 3 shows the test set ROC plot of the Logistic Regression trained using 70,000 samples and has a AUC of 0.76

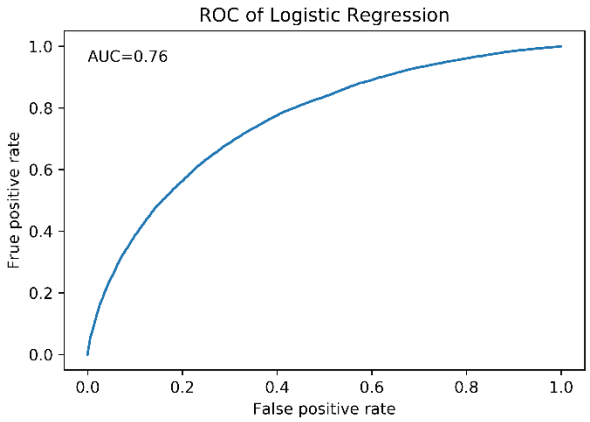

Figure. 3. Logistic Regression ROC plot on testing set

5.3 GRADIENT BOOSTING TREES

A randomized grid-search narrowed down the parameter space, and followed by several iterations of model fitting to find optimal parameters. Table. 2 shows the evaluation metrics of Gradient Boosting Trees trained with different parameters and sample sizes (shrinkage is set to 0.05 for all models)

Training Set Size	# of Trees	Tree Depth	Min Leaf Size	AUC-Train	AUC-Test	LogLoss-Train	LogLoss-Test
20,000	1000	4	1%	0.940	0.767	0.300	0.460
30,000	1000	3	1%	0.865	0.774	0.377	0.454
30,000	1500	3	1%	0.890	0.770	0.350	0.457
30,000	800	3	0.50%	0.857	0.773	0.388	0.456
30,000	300	3	0.50%	0.814	0.773	0.425	0.456
30,000	200	3	0.50%	0.803	0.770	0.430	0.459
30,000	100	3	0.50%	0.783	0.763	0.453	0.465
20,000	100	3	0.50%	0.790	0.763	0.447	0.467
100,000	100	3	0.50%	0.770	0.765	0.459	0.465

Table. 2. AUC and Log-Loss on training and testing datasets, for Gradient Boosting Models trained with different parameters and sample sizes

Figure. 4 shows the testing set ROC plot of the Gradient Boosting Trees trained using parameters {# of trees = 100, tree depth = 2, min leaf size = 0.5%, shrinkage = 0.05} and 20,000 samples. It has an AUC of 0.76.

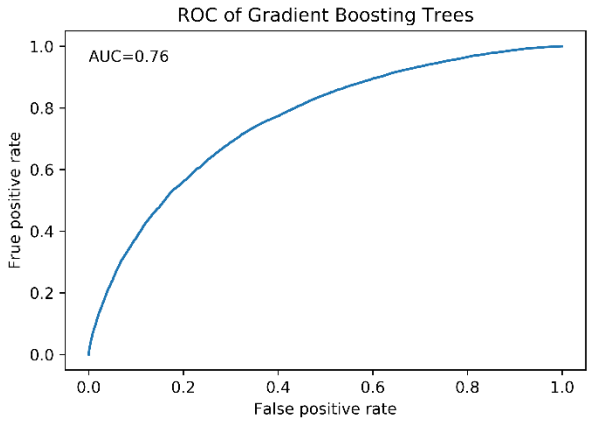

Figure. 4. Logistic Regression ROC plot on testing set

5.4 SUPPORT VECTOR MACHINE

To make training faster, top 500 features were retained based on mutual information value, while the rest 1400 features were dropped. Support vector machine with RBF kernel and C =1 was trained on 50,000 samples.

Confusion Matrix		Predicted	
		0	**1**
Actual	**0**	32,422	940
	1	8,713	1,495

Table. 3. Confusion matrix on test set

Since SVM only output 0/1 labels, confusion matrix is used to evaluate its performance. Table. 3 is the confusion matrix based on testing set. It has a precision score of 0.61 and a recall score of 0.15. The low recall score indicates the SVM is unable to correctly identify consumers who are likely to respond. Indeed, SVM classifies most instances as being 0, representing non-responders. Therefore, SVM performs poor in this use case.

5.5 NEURAL NETWORK

On entire 100,000 samples, several neural network were trained using different batch sizes, numbers of hidden neurons and scaling parameters of L-2 penalty term. With single hidden layer, the best neural network turns out to be using the parameter combination {batch size = 1000, number of hidden nodes = 300, scaling parameter of L-2 penalty term = 0.0003}. On training set, it generates a Log-Loss of 0.47 and an AUC of 0.79, while on testing set, the Log-Loss is 0.50 and the AUC is 0.75.

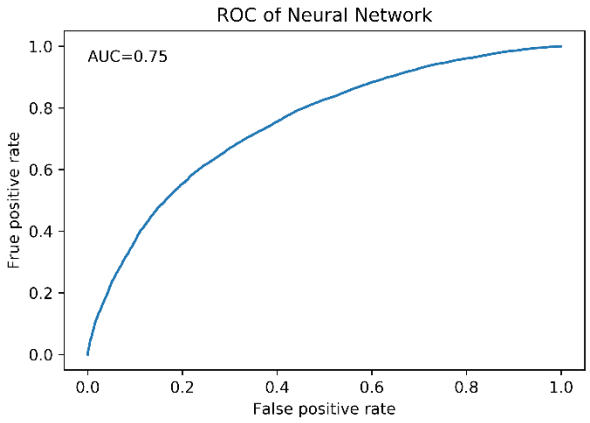

Figure. 5. Neural Network ROC plot on testing set

Figure. 5 shows the testing set ROC plot of the Neural Network with {gradient descent batch size = 1000, hidden layer = 1, hidden neurons = 300, scaling parameter of L-2 penalty term = 0.0003}. The AUC is 0.75.

5.6 COMPARISON AND DISCUSSION

Table. 4 summarizes the comparison of performance among these four models.

Model	Sample Size	Training Set		Test Set	
		Log-Loss	AUC	Log-Loss	AUC
Logistic Regression	70,000	0.45	0.78	0.47	0.76
Gradient Boosting Trees	20,000	0.45	0.79	0.47	0.76
Neural Network w/ Reg.	100,000	0.50	0.76	0.52	0.74
SVM w/ RBF Kernel	50,000	6.45	0.61	7.59	0.56

Table. 4. Comparison of model performance

Compared with Logistic Regression model, Gradient Boosting Trees reaches similar performance in terms of AUC and Log-Loss, but requires much smaller training dataset. Gradient Boosting Trees only requires 1/3 of samples that Logistic Regression model would require to reach optimal model performance. The training time of Gradient Boosting Trees is significantly shorter than that of Logistic Regression.

The drawback of SVM is three-fold. First, it doesn't perform so well as Logistic Regression or Gradient Boosting Trees does, for it has very low recall score and classifies most instances as 0. Second, SVM doesn't output probability, while financial institutions require to have probability because perspective consumers have to be ranked based on the probability. Third, the training time of SVM is much longer even just with a subset of features.

Model performance of Neural Network with one hidden layer is slightly worse than Logistic Regression or Gradient Boosting

Trees. With batch size being 1000, the training speed is very fast on 100,000 samples. When adding more layers, Neural Network should be able to achieve better performance than Logistic Regression and Gradient Boosting Trees.

From model implementation perspective, when implemented on large scale dataset, Logistic Regression might be the easiest one to implement, followed by Gradient Boosting Trees given the number of trees is small and tree depth is shallow. There might be some challenge to implement Neural Network to score consumers in batch, because Neural Network could easily have hundreds of thousands parameters and effective scoring needs vectorization.

6. Conclusion and Future Work

When there are sufficient training samples, Logistic Regression provides a good modeling choice as it is not prone to overfitting and doesn't have parameter to tune. Its training procedure is very straightforward. When the amount of training samples is limited, Gradient Boosting Trees performs better as it requires relatively small sample size to reach optimal performance. The training procedure of Gradient Boosting Trees involves careful parameter tuning, otherwise it could easily suffer from overfitting. Neural Network is fast and could potentially perform even better when training set is large. The only concerns are implementation hurdle and less interpretable. In consumer lending industry, these three models all provide probability as output, so they all fit in this industry. SVM may not be a good fit as it doesn't output probability.

Future work primarily focus on the deployment and implement of multi-layer Neural Network using various deep learning software, including Caffe, Torch, Theano, TensorFlow, Keras, PyTorch. Also, for this specific use case, it would be ideal to perform error analysis when feature names are known. As error analysis will allow further probe of the pros and cons of different learning algorithms from what types of errors they tend to make, as some types errors might be more cost then the other type of errors.

7. Reference

[1] David L. Olson, Bongsug(Kevin) Chae. *Direct marketing decision support through predictive customer response modeling.*

[2] Traci Lee Chu. *How to Drive Revenue With Customer Response Modeling.*

[3] Arun K Mandapaka, Amit Singh Kushwah, Dr. Goutam Chakraborty. *Role of Customer Response Models in Customer Solicitation Center's Direct Marketing Campaign*.

[4] Decision Sciences Institute, Merrill Warkentin. *Predictive Modeling of Customer Response Behavior in Direct Marketing*.

[5] Experian white paper of Marketing Optimization. http://www.experian.ie/assets/decision-analytics/white-papers/experian_marketing_opt.pdf

[6] Springleaf Marketing Response dataset on Kaggle: https://www.kaggle.com/c/springleaf-marketing-response/data

[7] http://scikit-learn.org/stable/modules/ensemble.html#gradient-boosting