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

CS 229 Final Project
Jincheng Cao, SCPD
Jincheng@Stanford.edu
Stanford University

Introduction

When running a direct mail campaign, lending institutions use predictive models to rank order perspective consumers based on the likelihood to respond. Lending industry has been largely relied on Logistic Regression to build these models. Several other techniques, including Gradient Boosting Trees, Support Vector Machine, and Neural Networks are going to be benchmarked against Logistic Regression, which serves as the baseline.

Data & Features

The dataset that being used to evaluate these modeling techniques is Springleaf Marketing Response dataset available on Kaggle. The dataset contains 145,231 anonymized marketing records with each having 1932 features and one label indicating whether a consumer responded or not to a mail campaign. Feature names are masked. .

Preprocessing

1881 numeric features are standardized after missing values being imputed. 34 categorical features are dropped due to regulation restriction. The rest 17 categorical features are one-hot-encoded. The overall response rate is 23.25%

Models

Logistic Regression – minimize loss function:

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

Gradient Boosting Classifier – Updating procedure is:

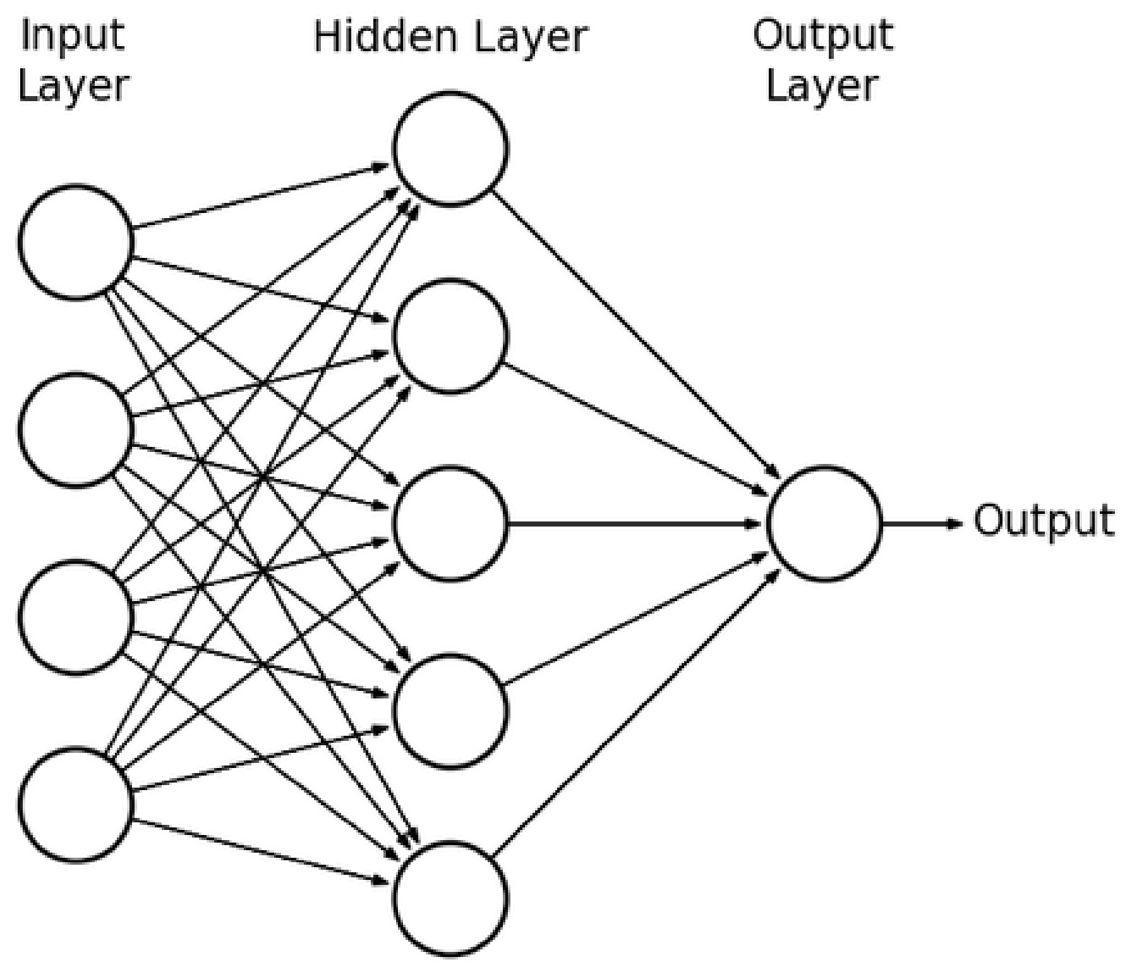
$$F_m(x) = F_{m-1}(x) + \arg\min_{h}\sum_{i=1}^{n} L(y_i, F_{m-1}(x_i) + h(x))$$

where h(x) is individual tree or weak learner.

Support Vector Machine – solve the following optimization problem:

$$\begin{aligned} &\min_{\gamma,w,b} \quad \frac{1}{2}\|w\|^2 \\ &\text{s.t.} \quad y^{(i)}(w^T x^{(i)} + b) \geq 1, \quad i = 1, \ldots, m \end{aligned}$$

Deep Learning/Neural Networks – with one hidden layer consisted of 300 neurons and one output layer.

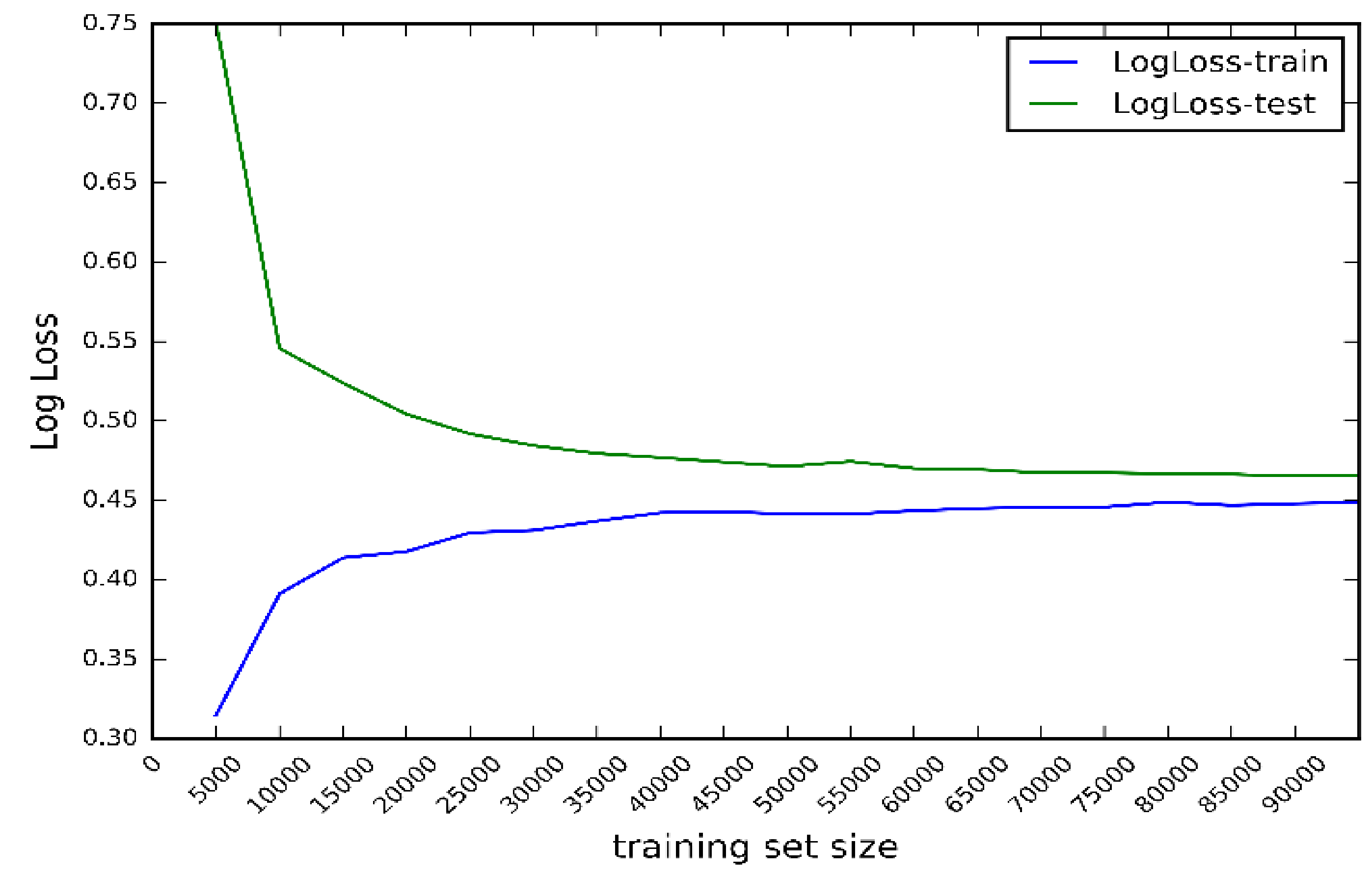


Each neuron is a sigmoid function with following formula:

$$S(x) = \frac{1}{1+e^{-x}}$$

Benchmark Model – Logistic Regression

Logistic Regression needs roughly 70,000 training samples to obtain optimal performance – which generates a Log loss of 0.466, and its corresponding AUC is 0.761.



Results

Logistic Regression needs roughly 70,000 samples to reach optimal performance, while Gradient Boosting Classifier reaches similar performance with only 20,000 samples. Neural Network performs a little worse with one hidden layer and 300 neurons, and it needs more samples to train. SVM doesn't perform well.

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 Classifier	20,000	0.45	0.79	0.47	0.76
Neural Network w/ Reg.	100,000	0.47	0.79	0.50	0.75
SVM w/ RBF Kernel	50,000	6.45	0.61	7.59	0.56

Error Analysis

GBT \ NN	0	1	2	3	4	5	6	7	8	9
0	46%	25%	15%	9%	3%	1%	0%	0%	0%	0%
1	24%	25%	20%	14%	9%	5%	2%	1%	0%	0%
2	15%	19%	20%	17%	13%	9%	4%	2%	0%	0%
3	8%	14%	17%	19%	16%	13%	9%	3%	1%	0%
4	4%	9%	13%	16%	18%	17%	13%	7%	2%	0%
5	1%	5%	8%	13%	18%	18%	18%	12%	5%	1%
6	1%	2%	5%	7%	12%	17%	21%	20%	12%	3%
7	0%	0%	2%	4%	8%	12%	19%	25%	22%	9%
8	0%	0%	1%	1%	3%	5%	10%	22%	33%	25%
9	0%	0%	0%	0%	1%	1%	2%	8%	25%	62%

Test set score decile crosstab between Gradient Boosting Classifier and Neural Network(normalized for each row) reveals certain disagree between the two models. But without feature names, it's difficult to actually investigate further.

Discussion

From above demonstration, clearly Gradient Boosting Classifier is preferred because it generates good performance while requires the least amount of samples. SVM fall short in terms of performance, and also it cannot generate probability-like score. So it might not be a good fit for this application. A very simple Neural Networks already performs almost as well as Gradient Boosting Classifier. A more complicated NNs should outperform Gradient Boosting Classifier by a large margin. However, the implementation of NNs might be a challenge for financial institutions.

Future

1. Train NNs with more complicated architecture (more hidden layers and neurons)
2. Find efficient ways to implement NNs