Deep Learning in Insurance Risk Quantification: A Comparative Study

Eugene Kwak

September 3, 2018

Contents

I. Definition

Project Overview

Balancing losses and premiums is the crux of managing a successful insurance business. This balancing act consists of being able to quantify and understand these financial losses (risk) that will be incurred in order to set premiums at rates that are both profitable and attractive to consumers. Here I focus on methods for quantifying risk. Insurers invest heavily on teams of analysts, domain experts and actuaries to perform this task. Countless studies are also regularly published exploring machine learning approaches to solve this problem and evaluate commercial solutions (see comparative analysis of risk models in healthcare by the Society of Actuaries from 2016)[1]. In short, risk prediction is and always has been one of the biggest challenges in insurance.

With the advent of machine learning and big data in commercial insurance, companies are eagerly seeking new ways to apply novel techniques to improve their business, but are often challenged with the complexities that come with supporting these technologies. Traditional methods used in practice include actuarial sciences and mostly linear models. More robust techniques like ensembling are recently being utilized. However, the industry is still behind in terms of state of the art methods like deep learning and the technologies to support them.

The goal of this project is to [1] determine the feasibility of deep learning in this task and [2] to do a proof of concept to make a case for insurers to invest in deep learning technologies. Allstate ran a Kaggle competition in 2016 and provided an anonymized dataset well-suited for this task[2].

Problem Statement

Can deep neural networks provide a viable solution for risk modeling in insurance claims?

Predictive models for assessing future risk are commercially available across nearly all insurance domains from automobile to healthcare. For example, Equifax offers its Insight Score[3] product while companies like Verscend sells its DxCG[4] product for healthcare insurance. In general, these solutions do not utilize deep learning for predicting future risk. The upfront investments needed simpy do not justify the marginal increase in performance deep learning often provides. Additionally, many domains have a strong preference for model interpretability, which is often why linear models are so well received.

I attempt to make a case for deep learning through this proof of concept. Three models were trained and evaluated.

1. Benchmark Model: A benchmark ensemble model was trained as a proxy for a competitive market solution that a deep learning model needed to beat.
2. Deep Learning Model (CPU): A deep neural network was trained to compare against the benchmark model. This model was also used to benchmark the cost-benefits of technologies optimal for deep learning (namely GPU's).
3. Deep Learning Model (GPU): The same deep neural network architecture was trained on a GPU cluster for comparison against the CPU trained model.

A successful deep learning model had to [1] beat the benchmark model in validation or cross-validation performance, [2] be comparible to the solutions provided via Kaggle submissions for Allstate's competition, and [3] be more computationally efficient on a GPU cluster. I prototyped a functional machine learning platform that can integrate with an organization's database technology stack to further simulate a business setting.

Metrics

The problem statement is broken into three comparisons as outlined above. Each comparison will be based on a different set of metrics specific to the specific goals.

Deep Learning vs Benchmark

One of the most common metrics to compare solutions and models is the humble coefficient of determination (R^2). If a suitable benchmark cannot be beaten, then there is no point in continuing with deep learning. The R^2 was computed on validation partitions.

	R-Squared		MAE		95th Percentile of Error	
	1,000	10,000	1,000	10,000	1,000	10,000
Diagnosis-Only Models						
ACG System	12.1%	16.0%	9.4%	2.9%	22.5%	7.2%
CDPS	7.8%	9.6%	9.2%	3.1%	22.5%	7.3%
DxCG	14.5%	18.8%	9.2%	2.9%	21.9%	7.2%
Impact Pro	13.1%	18.1%	9.2%	2.9%	22.2%	6.9%
MARA	15.2%	19.8%	9.2%	2.9%	21.7%	7.1%
Truven	16.6%	20.7%	9.1%	2.8%	21.3%	7.3%
Wakely	13.8%	17.4%	9.2%	2.9%	21.8%	7.2%
Pharmacy-Only Models						
ACG System	10.6%	14.0%	9.4%	3.0%	22.2%	7.1%
DxCG	13.8%	14.4%	9.2%	3.0%	22.0%	7.2%
Impact Pro	13.8%	13.8%	9.2%	3.0%	21.9%	7.3%
MARA	14.0%	15.3%	9.2%	2.9%	22.1%	7.1%
MedicaidRx	8.8%	8.4%	9.5%	3.1%	22.6%	7.3%
Wakely	10.3%	9.7%	9.4%	3.0%	22.9%	7.5%
Diagnosis-and-Pharmacy Models						
ACG System	13.9%	18.5%	9.4%	2.9%	22.3%	7.1%
CDPS+MRX	9.7%	10.7%	9.5%	3.1%	22.6%	7.2%
CRG	11.8%	12.1%	9.3%	3.0%	22.2%	7.4%
Impact Pro	15.8%	20.6%	9.1%	2.9%	22.0%	6.8%
MARA	18.5%	21.6%	8.9%	2.8%	21.7%	7.0%
Wakely	16.0%	18.5%	9.1%	2.9%	21.4%	7.1%
Prior Cost Models						
ACG System	14.5%	20.0%	9.2%	2.9%	21.9%	6.9%
DxCG	22.1%	24.3%	8.8%	2.8%	23.0%	7.1%
MARA	22.4%	24.9%	8.7%	2.8%	21.9%	6.8%
SCIO	14.3%	15.8%	9.2%	2.9%	22.1%	7.2%

Figure 1: 2016 Risk model comparisons by Society of Actuaries

Figure 1 shows a great example of how different models from a Society of Actuaries report[1]. I compared a deep learning model to the ensemble benchmark using the r2_score implementation from scikit-learn.

Eq 1. R^2

$$R^2(y, \hat{y}) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - \hat{y}_i)^2}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y_i})^2}$$

Deep Learning vs Kaggle

If a simple deep learning model beats the ensemble, it will be compared to other solutions provided through the 2016 Kaggle competition. The competition uses mean absolute error (MAE) on the provided test data to rate submissions. Again, I used scikit-learn's implementation. This metric answers the question of viability. During training, the MAE was evaluated on validation partitions. The final MAE was computed via Kaggle submission. I tried to beat the median submission MAE of 1,125, but fell short during the time I had for this project. I plan to continue modifying my network to beat this score.

Eq 2. Mean Absolute Error

$$MAE(y, \hat{y}) = \frac{\sum_{i=0}^{n_{samples}-1} |y_i - \hat{y}_i|}{n_{samples}}$$

Return on Investment

Lastly, I gathered metrics for computation performances, financial costs and opportunity costs to make a case for the return on investment on a full deep learning solution (including the technology stack). More specifically:

- Cost per R^2 gain
- Training and scoring times
- Storage and computational overhead

II. Analysis

Data Exploration and Visualizations

Data Description

The data was provided in a fully anonymized state from Allstate via Kaggle. A training and test dataset were given.

Table 1: Counts from data provided by Allstate

partition	rows	columns	count_rows_na	count_rows_dupe_id
train	188318	133	0	0
test	125546	131	0	0

The training data had about 188k rows. There are 130 continuous and categorical features and 1 label "loss". I created an additional column called "log_loss" which is just the natural log of the label column provided. Each row represents a single claim. The data were scrubbed and cleaned so minimal data processing was needed for this project. There were no duplications or nulls. This is actually how a machine learning application may work in practice. The application will only see data that is in some canonical format provided by the organization's data engineering processes.

There were 14 continuous features prefixed with "cont", 116 categorical features prefixed with "cat", and a claim identifier.

Target Variable

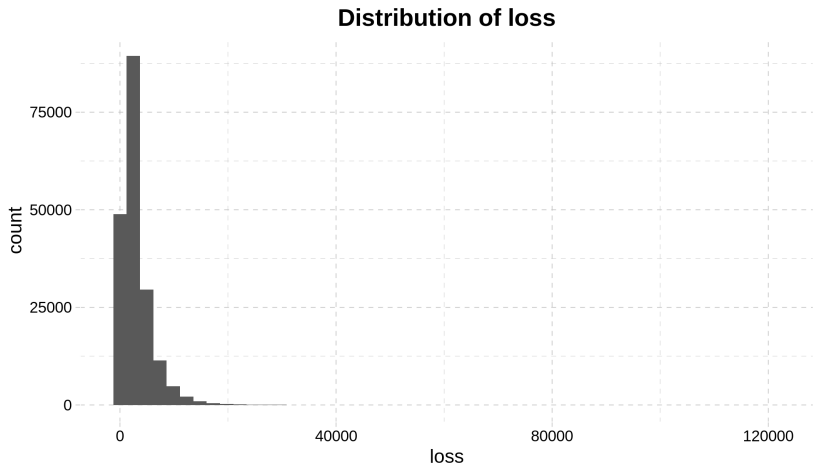

Figure 2: Histogram of the loss variable

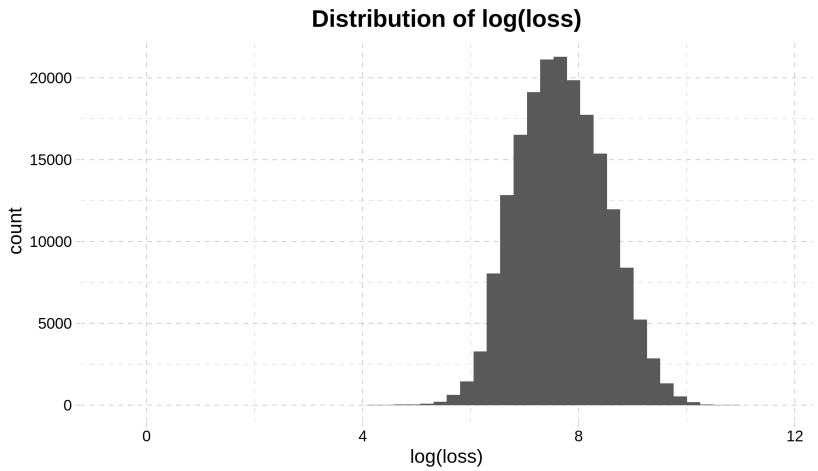

Figure 3: Histogram of the loss variable log transformed

Figures 2 and 3 show the distribution of the provided target variable before and after log transformation to improve the distribution for machine learning.

Continuous Features

The 14 continuous variables provided appear to have been normalized to a mean of 0.5 and standard devation of 0.2 by Allstate.

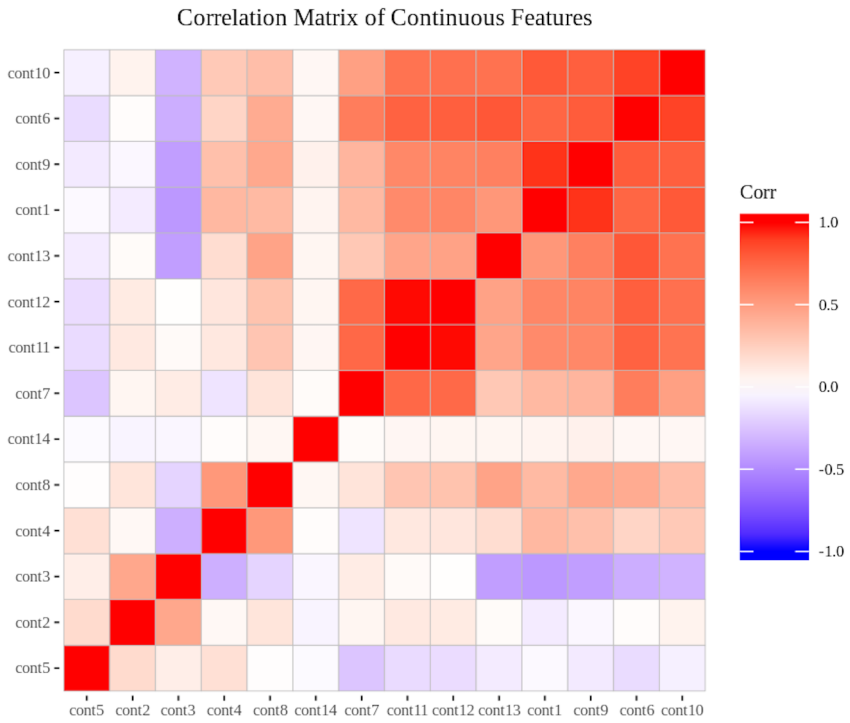

Figure 4: Correlation heatmap of the continuous features

Several features are highly correlated with each other such as cont6 and cont10. Those at the bottom left of the correlation heatmap in Figure 4 have much less collinearity and ended up being good predictors.

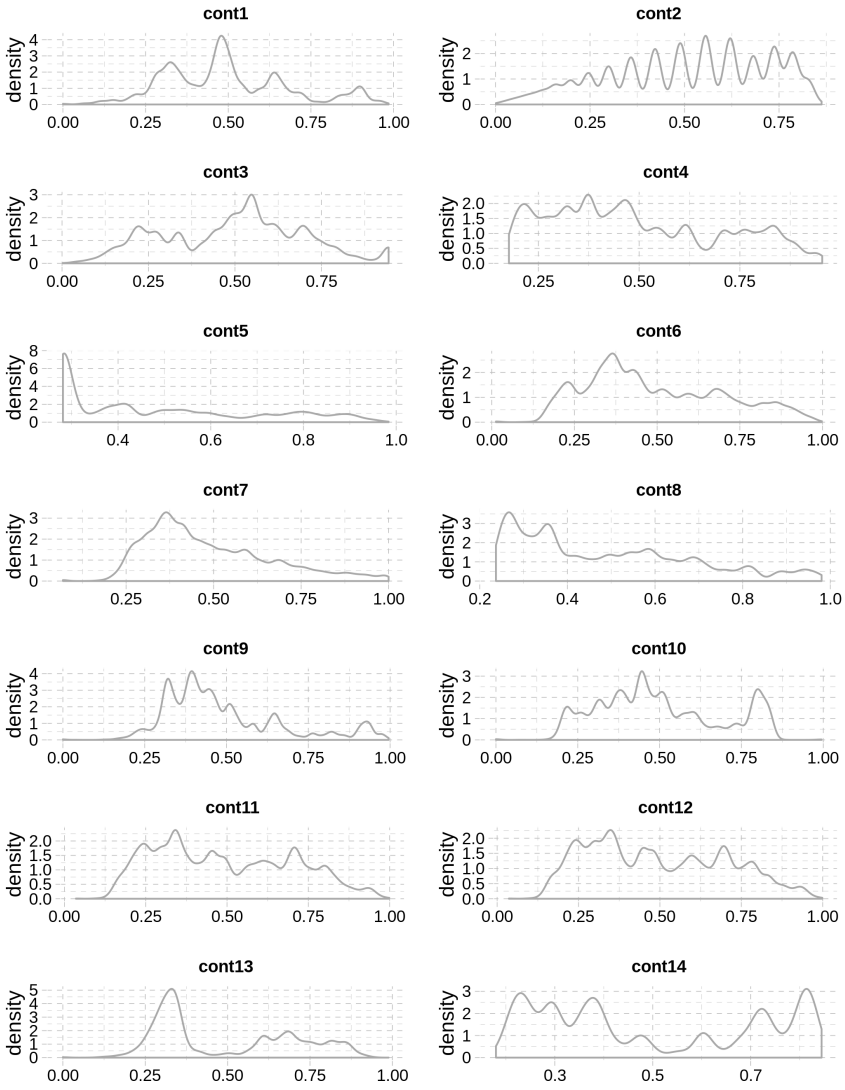

Figure 5: Density plots of the continuous features

The density plots show that not all continuous variables appear to be truly continuous. In particular cont2, looks like it could have been a discretized numeric variable given its density patterns.

Categorical Features

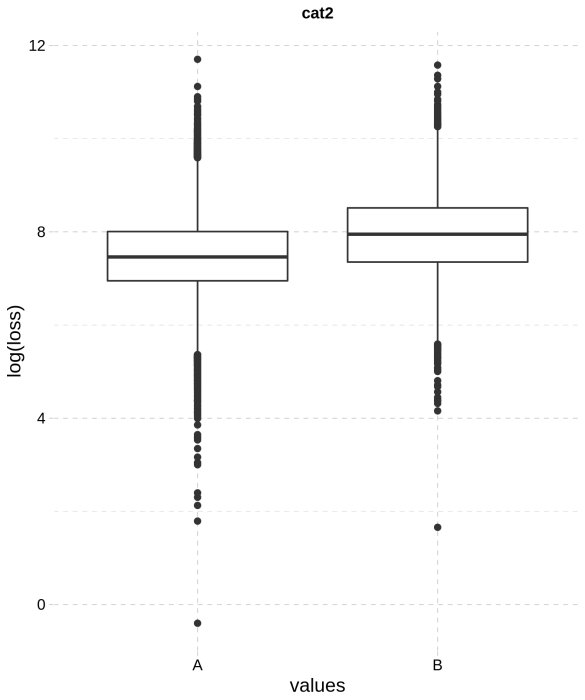

Figure 6: Box plot of cat2

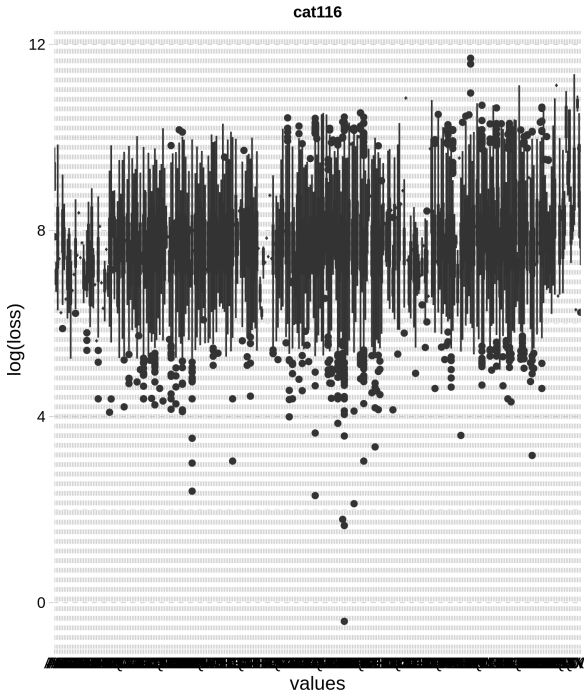

Figure 7: Box plot of cat116

Figure 6 shows an example of how most categorical variables were distributed. They had a cardinality of two with some outliers. There were 17 categorical variables with a cardinality greater than ten with cat116 (Figure 7) having the highest cardinality of 326.

Note: I did not apply any logic for dealing with high cardinality as my goal was not winning Kaggle, but rather to prove a case for deep learning. Also, this kept things simple enough for this project to remain tractable given the limited time provided.

Algorithms and Techniques

Feature Selection

The only pre-processing step on the feature space was to one-hot encode all of the categorical variables. This gave me 1,153 features in total. A decision tree regressor was fit to this processed dataset and feature importances extracted. Over many iterations, I found that a threshold of $1.05 \times mean(allfeatureimportances)$ gave me reasonably performant models. In other words, I selected variables with a feature importance score greater than this threshold value, netting me 152 input features.

Benchmark Ensemble Model

I tried multiple regressor classes available in scikit-learn's library and settled on simpler algorithms for the three sub-models that I ensembled. Two were simple linear models and one was a decision tree. I started with more complex algorithms, but found them to be too slow to iterate on and decided to use very simple models that are easy to tune and scale well.

1. Decision tree
2. Ordinary least squares
3. Stochastic Gradient Descent regressor

The final ensembler was an ordinary least square linear regression model to stack the results of the three sub-models. Grid search and 10-fold cross-validation were used to explore a wide hyperparameter space.

Deep Neural Network Model

For the purposes of this exercise, I designed a simple Multilayer Perceptron that can train relatively quickly, allowing me to iterate and test my development in the shortest amount of time possible. Again, the goal was not to build the most accurate model. The details of the model is discussed in the Methodology section of this document.

Network architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 1024)              156672
_________________________________________________________________
dense_2 (Dense)              (None, 512)               524800
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 256)               131328
_________________________________________________________________
dropout_2 (Dropout)          (None, 256)               0
_________________________________________________________________
dense_4 (Dense)              (None, 64)                16448
_________________________________________________________________
dropout_3 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_5 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_6 (Dense)              (None, 16)                528
_________________________________________________________________
dense_7 (Dense)              (None, 1)                 17
=================================================================
Total params: 831,873
Trainable params: 831,873
Non-trainable params: 0
_________________________________________________________________
```

Benchmark

In this section, you will need to provide a clearly defined benchmark result or threshold for comparing across performances obtained by your solution. The reasoning behind the benchmark (in the case where it is not an established result) should be discussed. Questions to ask yourself when writing this section:

Has some result or value been provided that acts as a benchmark for measuring performance? Is it clear how this result or value was obtained (whether by data or by hypothesis)?

III. Methodology

Data Preprocessing

As mentioned earlier, categorical variables were one-hot encoded, the target variable was log-transformed, and a decision tree was used to select relevant predictors. To allow more time on the deep learning portion and prototyping a machine learning platform, I decided to forego any further improvements to the feature space as I found reasonable comparisons can be made with just these few. Also, the data were anonymized so my ability to engineer better features was greatly limited.

Implementation

In this section, the process for which metrics, algorithms, and techniques that you implemented for the given data will need to be clearly documented. It should be abundantly clear how the implementation was carried out, and discussion should be made regarding any complications that occurred during this process. Questions to ask yourself when writing this section:

Is it made clear how the algorithms and techniques were implemented with the given datasets or input data? Were there any complications with the original metrics or techniques that required changing prior to acquiring a solution? Was there any part of the coding process (e.g., writing complicated functions) that should be documented?

Refinement

In this section, you will need to discuss the process of improvement you made upon the algorithms and techniques you used in your implementation. For example, adjusting parameters for certain models to acquire improved solutions would fall under the refinement category. Your initial and final solutions should be reported, as well as any significant intermediate results as necessary. Questions to ask yourself when writing this section:

Has an initial solution been found and clearly reported? Is the process of improvement clearly documented, such as what techniques were used? Are intermediate and final solutions clearly reported as the process is improved?

IV. Results

(approx. 2-3 pages)

Model Evaluation and Validation

In this section, the final model and any supporting qualities should be evaluated in detail. It should be clear how the final model was derived and why this model was chosen. In addition, some type of analysis should be used to validate the robustness of this model and its solution, such as manipulating the input data or environment to see how the model's solution is affected (this is called sensitivity analysis). Questions to ask yourself when writing this section:

Is the final model reasonable and aligning with solution expectations? Are the final parameters of the model appropriate? Has the final model been tested with various inputs to evaluate whether

the model generalizes well to unseen data? Is the model robust enough for the problem? Do small perturbations (changes) in training data or the input space greatly affect the results? Can results found from the model be trusted?

Justification

In this section, your model's final solution and its results should be compared to the benchmark you established earlier in the project using some type of statistical analysis. You should also justify whether these results and the solution are significant enough to have solved the problem posed in the project. Questions to ask yourself when writing this section:

Are the final results found stronger than the benchmark result reported earlier? Have you thoroughly analyzed and discussed the final solution? Is the final solution significant enough to have solved the problem?

V. Conclusion

(approx. 1-2 pages)

Free-Form Visualization

In this section, you will need to provide some form of visualization that emphasizes an important quality about the project. It is much more free-form, but should reasonably support a significant result or characteristic about the problem that you want to discuss. Questions to ask yourself when writing this section:

Have you visualized a relevant or important quality about the problem, dataset, input data, or results? Is the visualization thoroughly analyzed and discussed? If a plot is provided, are the axes, title, and datum clearly defined?

Reflection

In this section, you will summarize the entire end-to-end problem solution and discuss one or two particular aspects of the project you found interesting or difficult. You are expected to reflect on the project as a whole to show that you have a firm understanding of the entire process employed in your work. Questions to ask yourself when writing this section:

Have you thoroughly summarized the entire process you used for this project? Were there any interesting aspects of the project? Were there any difficult aspects of the project? Does the final model and solution fit your expectations for the problem, and should it be used in a general setting to solve these types of problems?

Improvement

In this section, you will need to provide discussion as to how one aspect of the implementation you designed could be improved. As an example, consider ways your implementation can be made more general, and what would need to be modified. You do not need to make this improvement, but the

potential solutions resulting from these changes are considered and compared/contrasted to your current solution. Questions to ask yourself when writing this section:

Are there further improvements that could be made on the algorithms or techniques you used in this project? Were there algorithms or techniques you researched that you did not know how to implement, but would consider using if you knew how? If you used your final solution as the new benchmark, do you think an even better solution exists? Before submitting, ask yourself. . .

Does the project report you've written follow a well-organized structure similar to that of the project template? Is each section (particularly Analysis and Methodology) written in a clear, concise and specific fashion? Are there any ambiguous terms or phrases that need clarification? Would the intended audience of your project be able to understand your analysis, methods, and results? Have you properly proof-read your project report to assure there are minimal grammatical and spelling mistakes? Are all the resources used for this project correctly cited and referenced? Is the code that implements your solution easily readable and properly commented? Does the code execute without error and produce results similar to those reported?

Sources

[1] https://www.soa.org/research-reports/2016/2016-accuracy-claims-based-risk-scoring-models/

[2] https://www.kaggle.com/c/allstate-claims-severity

[3] https://www.equifax.com/business/insight-score-insurance/

[4] https://www.verscend.com/solutions/performance-analytics/dxcg-intelligence

[5] Exploratory Data Analysis guided by Kaggle Kernel submissions (https://www.kaggle.com/c/allstate-claims-severity/kernels)

Supplemental Information

All code and additional documentation and visualizations can be retrieved from my github repository.

```
git clone https://github.com/eugenekwak/Allstate_DL.git
```