Machine Learning Engineer Nanodegree

Capstone Project

Omid Heravi
December 27th, 2017

I. Definition

I.1 Project Overview

Introduction

Inusrence companies can and will use Machine learning more and more frequently in the near future. As a data intensive industry, Insurence companies can all benefit from the use of Machine learning techniques in processing insurence claims. As per Allstate's challenge, we will use several machine learing techniques with the data provided to conduct data analysis and predict the reuqested attribute.

Scope of this project

We will investigate using Machine Learning Tecinques to predict the requested loss column/attribute. Each row in the data corresponds to an insurence claim. Each column/attribute corresponds with a feature that will help us in data analysis process to better predict the 'loss' attribte. Despite the fact that all columns are ambigious, the 'loss' columns does not explictly tell us anything about what we're predicting, however those details are not necessary in the grand scheme of analysis.

Why Insurence is an interesting domain of Machine learning:

1. There are many instances where we or our loved ones have been involved in an accident, tragedy or etc. and which is when we all rely on ur insuence provider for relief and help. The importance of this issue is very personal to me as my father who recently was involved in accident would have benefited from an speedy process. Machine learning can quickly improve the process, effiecenly and productivity of every person involved during an insurence claim process.
2. Machine learning can help us uncover hidden connections and patterns we had never realized before. This can include better indicators for prevention of accidents among individuals, lowering the cost of insurence due to automation and efficiency, better policy desinging and so many more.

3. As a data intensive industry, Machine learning is poised to take over the insurence claim analysis pipeline thus proving to be a formidable business venture for many entrepreneurs and problem solvers.

Aim of this project

The aim of this project is to get a feel for which features are better indicators are of an insurence claim severity, as mentioned in the challenge page. Predicting the severity of an insurence claim is very difficult yet with the data provided, we hope to as accurately predict which features and which model perform the best when it comes to predicting the severity of an insurence claim. More details in Problem Statement.

Data Used in this Project

There are two primary Datasets for this project:

1. Train.csv = train Data provided by Allstate Challenge. The original Shape of the data is in (188318, 131). This dataset also includes the 'loss' column by which we'll be completing the challenge.
2. test.csv = test data provided by the challenge. The original shape of the data is in (125546, 130). This datset however does not include the 'loss' column. also train set does not include as munch insurance isntance claims as much as the train dataset.

I.2 Problem Statement

Problem

Build a Model that can predict the severity of an insurence claim:

Category	Details
Input	The original 129 attributes/features (exclusing the 'ID' and 'Loss' columns). once again, due to the ambiguity of the feature set, we cannot further intrepet what is our input, however we can assume that data such as date, location, estimated cost, type and etc.
Output	Prdicted value/severity of the insurence claim. The output will be labeled 'loss'. Once again due to ambiguity of the overall dataset, we assume 'loss' is basically the integer value of a claim's seveirty.

Interesting characteristics of this problem

There are a few inetresting key points I would like to point out:

1. The data is a mix of both categorical and continious data.
2. The 'loss' column is heavily skewed and will need to be transformed.
3. In fact all the data is already provided, so there is no need for data mining, cleaning and or etc.

Challenges

1. This challenge requires heavy duty data exploartion just so you can wrap your head around the given dataset since most of the columns are ambigious.
2. A simple solution can often be the best solution despite the fact that it's a simple solution even though one might be tempted to try an over-the-top solution/algo/model which is quite unnecesary.

Analysis of Problem

This problem is a simple regression problem since we're asked to predict the 'loss'/severity attribute as a result of the other attributes. It is not initally obvious which kind of model will perform best.

Charahcteristics of the problem:

- continuous data
- categorical data
- Noisy data
- ambigious features
- Regression Problem
- Predicting one single attribute

Strategy

Understanding the Dataset

- Data Shape
- Skew
- Mean, STD deviation, Max, etc.

Data Visualization

- Scatter Plots
- Correlation
- Density Plots

Data Processing

- Transformation
- One-Hot encoding
- Train/Test Data preparation

Model Evalaution

- Algorithm implementations

- Model Evaluations
- Model Predicitons
- Model Analysis

Appendix

- Conclusion
- Personal Thoughts
- Thanks and Appreciations

Predicted Solution

The solution will be in a .csv format where the predicted 'loss'/severty value will be stored. For each insurence claim 'ID', it will follow with a 'loss' value.

I.3 Metrics

As requested by the kaggle challenge its self, we will be using **MEAN ABSOLUTE ERROR** or also known as **MAE**.

$$\mathrm{MAE} = \frac{1}{n}\sum_{j=1}^{n} |y_j - \hat{y}_j|$$

The Mean Absolute Error as a metric is ideal for most of simple regression problems, hence why it has been requested of us to use it. MAE can covey very usefu informaiton. The formula takes the avergae of the absolute difference between the predicted label value and the actual label value. This makes it much easier to interpret incomparison to other alternative regression error metrics such as MSE, RMSE, and etc. JJ (https://medium.com/human-in-a-machine-world/mae-and-rmse-which-metric-is-better-e60ac3bde13d) explains in detail the beautiful simplicity and ease of interpreation of MAE.

II. Analysis

II.1 Data Exploration

Description of Primary Dataset

The provided dataset is in two sets. Train and Test datasets. Both vary. The train set has a lot more insurence claim instances (rows) than the test set. It also includes the 'loss' attribute hence why we'll be predicting the 'loss' value. The datasets both contain categorical and continious data. With 129

actual attributes/features, and ~188000 rows, the training dataset is vast.

Some key facts about the dataset:

- Categirical data is in 116 columns
- The categorical data is quite large, and will require one-hot encoding
- Continious data set is about 15 columns
- This is fairly easy to process and will need to further transformation
- 'Loss' or the label to be predicted is heavily skewed and will require log-transformation
- No data is missing.

Dataset Statistics

Continuous Descriptive satistics:

The descriptive statistics does not prove to be very useful, however it is a great method to understand the variance in our data.

Skewness:

Column	Skewness
cont1	0.516424
cont2	-0.31094
cont3	-0.01000
cont4	0.416096
cont5	0.681622
cont6	0.461214
cont7	0.826053
cont8	0.676634
cont9	1.072429
cont10	0.355001
cont11	0.280821
cont12	0.291992
cont13	0.380742
cont14	0.248674
loss	3.794958

Descriptive Stats:

	cont1	cont2	cont3	cont4	cont5	cont6	1
count	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000	188318.000000	
mean	0.493861	0.507188	0.498918	0.491812	0.487428	0.207202	
std	0.187640	0.207202	0.202105	0.211292	0.209027	0.205273	

min	0.000016	0.001149	0.002634	0.176921	0.281143	0.012683
25%	0.346090	0.358319	0.336963	0.327354	0.281143	0.336105
50%	0.475784	0.555782	0.527991	0.452887	0.422268	0.440945
75%	0.623912	0.681761	0.634224	0.652072	0.643315	0.655021
max	0.984975	0.862654	0.944251	0.954297	0.983674	0.997162

Categorical Descriptive satistics:

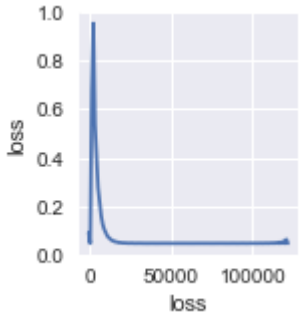
The same analysis could be applied to Categorical dataset as well. Allbeit, since it is categorical data, Descrptive statistics wont be the same. Niether will there be any inherent 'skewness'. Thus the following data is appropriate:

	cat1	cat2	cat3	cat4	cat5	cat6	cat7	cat8	cat9	cat10	cat11
count	188318	188318	188318	188318	188318	188318	188318	188318	188318	188318	188318
unique	2	2	2	2	2	2	2	2	2	2	2
top	A	A	A	A	A	A	A	A	A	A	A
freq	141550	106721	177993	128395	123737	131693	183744	177274	113122	160213	168186

'Loss':

The 'Loss' attribute/feature or also the label which we'll try to predict, is considered its own separate data. The descriptive and skewness of the 'Loss' attribute/feature is shows below:

	Loss
count	188318.000000
mean	3037.337686
std	2904.086186
min	0.670000
25%	1204.460000
50%	2115.570000
75%	3864.045000
max	121012.250000

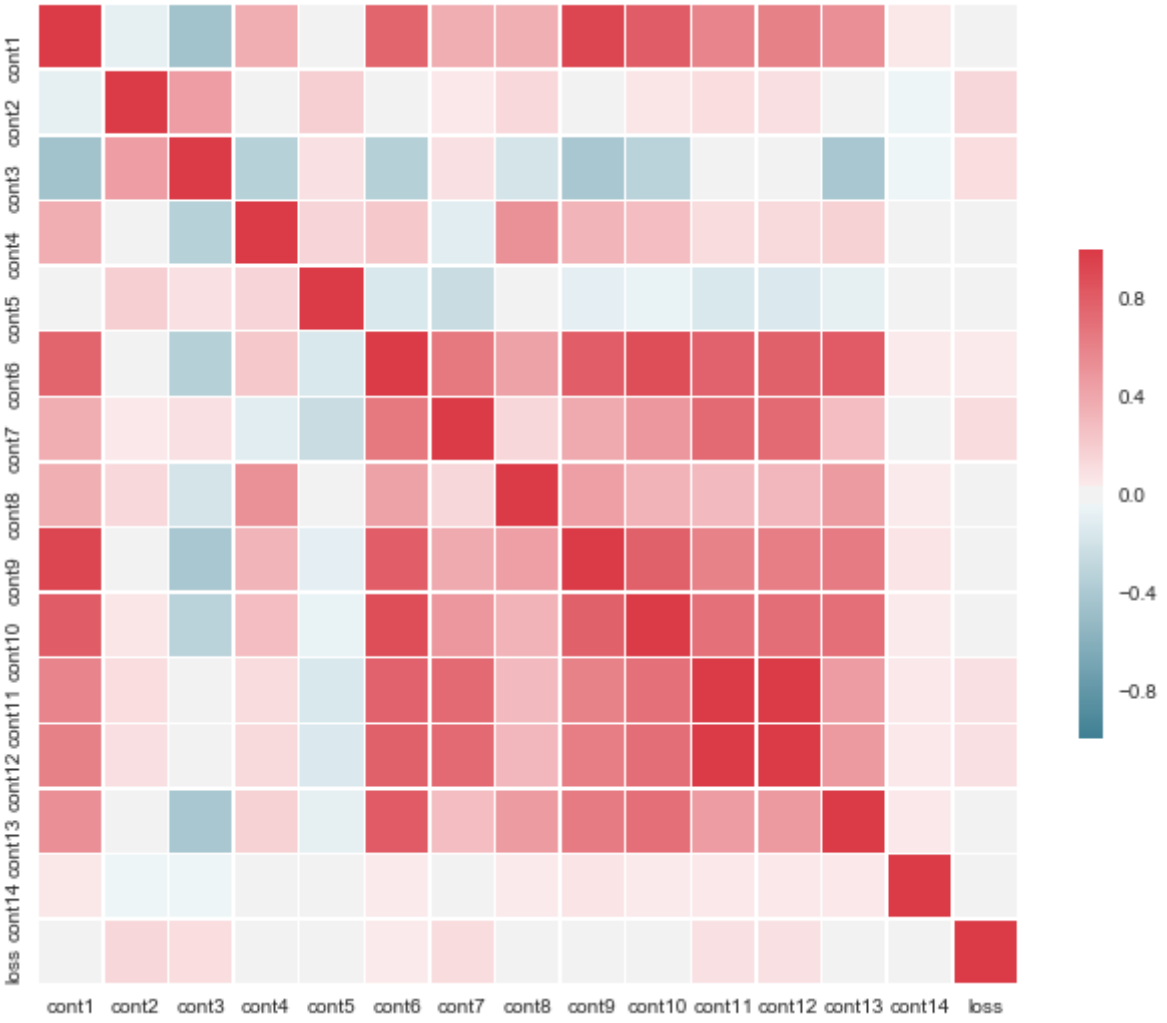


The values for the 'loss' feature are abnormally large and if we ever plan to feed em into a model, we must take into consideration that the data from 'loss' will have to be normalized. Now we'll go into further detail in future in regards to the type of normalizaiton we use.

II.2 Exploratory Visualisations

Continious Data

One amazing way we can convey and understand the relationship among some continious varibale is using a heatmap/pairplot sort of plotting which can clearly explain the commonality among all features in a matrix.



Observations:

- There are quite a lot of features that strongly correlate with one another. For example: 'Cont11' & 'Cont12' are highly correlational. Same goes for 'Cont1' & 'Cont9'. and so on untill we have crossed the margin of 0.0. In which case those columns are not correlated to one another.
- Dimension reduction would be an ideal strategy if the data one had was quite extensive/large, however since ours is not, we'll leave it as is for now.

Categorical Data

Categorical data by nature is data in terms of strings and alphabet. There are quite a lot of ways analysize categorical data. As is done by a fellow kaggler, Allstate Feature Analysis by Achal (https://www.kaggle.com/achalshah/allstate-feature-analysis-python), you could also extract correlation data among categorical features. However, for my own analysis I only relied on a simple countplot which would display the unique occurence of each string and its count.

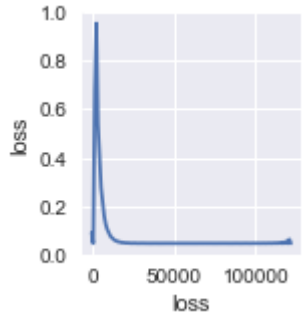
For visual images please refer to the main notebook

'Loss'

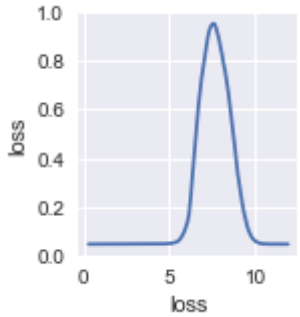
The dependant varibale, also known as the label thats going to be predicted, can be visualzied in multiole ways. Other fellow kagglers have amazing descriptive visual analysis of this feature, however I will be explainiing my own approach.

Since not a lot of visualization is necessary to get an idea of how 'loss' is represented, a simple visualization of its skew would be enough.

The original Skew before transformation:



After log-transofrmation:



II.3 Algorithms and technique

Types:

For this Challenge, as mentioned earlier the problem is a regression problem hence regression algorithms are my choice of algoithms/models.

Algorithm Description:

- LinearRegression: This is a simple Linear regression. It takes in a train sample and attempts to minimize the error by trying to fit the model to the train set.
- SGDRegressor: Unlike Linear Regression, SGD takes the gradient of it's error is computed, applied at each time step with a decreasing multiplicator/learing rate taken into consideration.
- Decision Tree Regressor: A tree like graph in which a statistical probabiity is assigned to each node and thus each data that is fed in, outputs a prediction with each node/attribute in mind.
- Random Forest Regressor: Can be considered a meta-DecisionTree which encompasses multiple classifying decison trees at each node, which inevitably improves the predictive accuracy of the model over a single Decision Tree.
- GradientBoostingRegressor: An ensemble of decison trees that build on top of each other's previous error, summed and itertively minimzies the error untill a certain tolerance is reached.
- MLPregressor = A mulitlayer perceptron which can be a mix of various inout_layers, hidden_layers, and output_layers. Input layers usually correlate with the amount of features an input training data set has, hidden_layers correlate with multple activation functions that compute a gradient and and output that is fed out to output_layers.

Algorithm Parameters:

Not a lot of parameter tunning was done to avoid overfitting since the data is not as noisy.

- LinearRegression: Eevrything was kept as default, but n_jobs was changed to '-1' so that it can provide a speedup if the problem is large and you ask the algorithm to use more CPUs, but it will not change error measures.
- SGDRegressor: Everything else was kept as default, except max_iter, tol, epsilon, and alpha just for to avoid the default settings and to introduce a bit randomnesss. The manual paramters did not change the error as much.
- Decision Tree Regressor: Everything else was kept as default, however only the random_state parameter was set to 0, to improve consistency.
- Random Forest Regressor: Everything was kept as default, however n_jobs was set to '-1' for reasons mentioned above, and most importantly the criterion had to manually be set to 'mae' to avoid a huge bug as mentioned in this link. https://mail.python.org/pipermail/scikit-learn/2016-October/000757.html (https://mail.python.org/pipermail/scikit-learn/2016-October/000757.html) which caused a huge time consumption for some reason. Hence why i chnaged to 'mse' despite the fact that the challenge did not permit it.
- GradientBoostingRegressor: Everything was kept as default, except Random_state, n_estimators. n_estimators was manually set to 25 b/c the default value, 100, took an insame amount of time on a macbook and did not yield any significant error loss.
- MLPregressor: Everything was kept as default, except Random_state to improve consistency.

Techniques

1. **train_test_split**
    - We will train our model on what we'll call the training set, a subset of the data that we have.

- To make sure our model generalises, we need to test it on some data it has not seen before and evaluate how well it does predicting on that data.
- To do this, we need to set aside data for testing our model - the test set.
- The 'Loss' label column of our will be split into two sets of data, X_test, y_test.
- Every other feature will be split into two training sets, X_train, y_train.

III. Methodology

III.1 Data Preprocessing

Data Tranformation

- Since our Loss feature value as mentioned above needs to be log-normalized, we should do that before we feed our data into the models.
- This can prevent any abnormal behavior and ourright normalize the data that will be outputted.
- The log transform formula is:

$$Log(1+x)$$

- Further detail on this transformation can be found here; https://scicomp.stackexchange.com/questions/20629/when-should-log1p-and-expm1-be-used (https://scicomp.stackexchange.com/questions/20629/when-should-log1p-and-expm1-be-used)
- keep in mind, to return the data back to it's original form you have to multiply the test set, aka the split 'loss' labels, by expm1, aka its inverse function to get the original value back.

$$exp(x)-1$$

One-Hot encoding

- In order for us to feed in the categorical data, we have to also transform the categorical data into a numerical data.
- The most common method for such transformation is One-Hot Encoding. It is widely used and proven by many reaserchers to work very well and often improve machine learning as explained by another fellow data scientists at this link; https://stackoverflow.com/questions/17469835/why-does-one-hot-encoding-improve-machine-learning-performance (https://stackoverflow.com/questions/17469835/why-does-one-hot-encoding-improve-machine-learning-performance)
- Anyways, I had to encode all the categorical data by first using the labelencoder, which in turn will turn our string data into numerical data.
- Right after, the data must be reshaped from a column array into vector array so it can be fed into One_Hot encoder next.
- next, we'll use the One-Hot encoding algorithm to output a sparse matrix in which where each column corresponds to one possible value of one feature.
- This process will output a matrix that is suitable for most if not all scikit estimators, which is why we must always encode the charcther data ino numerical data.

III.2 Initial implementation

Process

1. Imported algorithm
2. Initialized the algorithm
3. Fit and trained the model on the split training data; X_train, y_train.
4. I used the predict method available to the algorithm and used it on the X_test data.
5. I obtained the result of Mean Absolute Error algorithm by using the y_test as ground truth, and the prediction as my estimated lable.
6. I appended the score achived from this algorithm, along with the name of the alogrithm to a separate list for later visualizing and analyzing each algorithm.

This process was applied to each and every algorthm since they share a common library.

III.4 Refinement

1. Remembering to transform back my previously log normalized 'loss' column as mentioned before, I had to log normalize my labels and I initally forgot to transform them back upon feeding them into the Mean_absolute_error function hence Why abnormal results were produced.
2. I decided to re-edit the linear regression model, in which it iteratively goes through each column successively, training, predicting, and summing the residual error. This prooved to be quite amazing in fixing the atrocious result I kept getting before in which the error was in 6-7 figures. The new error seems much more reasonable and less like an outlier. This is the only process by which the error remains reasonable. For the reviewer feel free to play around with the model and let me know if there are any better improvements.

IV. Results

IV.1 Model Evaluation and Validation

Model Choice

The final Model is:

- Features: The test.csv was used in the final prediction process. With all features included, Categorical data encoded and continius data remained the same.
- Classifier: The final choice of classifier was SGD. This was the shocking result.
- Target: Predict Insurance claim severity, aka the value of the 'loss' column/feature.

The model had the lowest mean absolute error across a 10 trails. Insight: Most of the improvements and complexer models I tried to make only made the model worse. This goes to show that simple soluions/models are just enough to solve a regression probelm and over-the-top complex models are unnecessary.

IV.2 Justification (Comparison with expectations)

Overall, this model aligns with solution expectations and on average performed close to what other kaggle competitors have achieved. Shocking result was the fact that SGD came on top even if that margin was in the decimal places, however since there is no benchmark to compare our model to, the true ground truth labels, the results we have obtained are as is. That being said, the significant speed up of using SGD over other time extensive algo's such as XGBoost, random forest, and even MLP, make SGD a great contender.

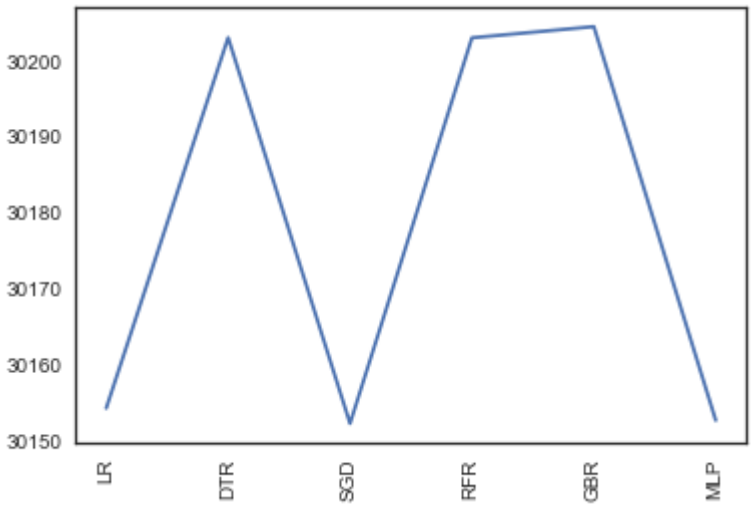
Despite the ambiguity in dataset and lack of benchmakr, I belive the model given the constraints performed very well, with +/- 5% error margin on each trial proving to be quite consistent.

V. Conclusion

V.1 Free-Form Visualisation

This graph visualized the final model perfromance stack up when compared withone another:

Inline-style:



As you can see by the slightest margin, SGD is the model with the lowest MAE error, hence the best model. And keep in mind the final scores had to be multiplied by scalar 10 to reduce ambiguity of which is lower from decimal place to an integer.

Original scores in order of Linear Regression, DTR, SGD, RFR, GBR, MLP:

V.2 Reflection

Summary

In this project, we predicted the severity of an insurence claim.

We started off by using a simple linear regression, based on the given train data split into four datasets. This dataste did not miss any data, was a mix of both continious and categorical data.

In this process we conducted the follwing steps:

```
1. Imported .csv Data by using pd.read_csv() into a pd.Dataframe
2. Selected the dataframe and separated the continious and categoric
al data
3. Separated the target variable, dependant variable, label, aka 'lo
ss'
4. Split the data into four separate sets using sklearn.train_test_s
plit function
5. Trained all the Models
6. Predicted the lable with each model
7. Evalaute the accuracy with the Mean Absolute Error as our metric
```

After so, this process was repeated multiple times with different algorithms ranging from Lasso Linear regression to several others, manually parameter tuned, denugged and finally settled with this result.

Interesting Aspects of the Project

1. Ambiguuty of the data features: Even though this data was provided as part of the challenge, the data was not equipped with meaningful explicit feature names. This made the data analysis part somewhat very challenging to interperet at times.
2. Data was already provided: This is such an understatment, for most Data challenges getting the data is often the hardest part, and thankfully this was already provided.
3. Abundant refrences and resources: thankfully everytime I was confused or just clueless, the kaggle community has provided amazing public kernels for beginers and data scientists such as I to help understand the challenge and often the nitty gritty bugs that one should expect to encounter.

Difficult Aspects of the Project

1. The difficult part for me was understadning the data and as someone who is so used to data with explicit menaingful features, I could not get over the fact that this dataset was void of any meaningful feature/column names which often resulted in-productivity and stressed fits over trying to understand why and how each feature would really be useful besides the statistaical correlation among them. If provided, I belive I could have been much more able to draw conclusions and explanations.
2. Some sklearn algorithms have many little bugs, often one little parameter could ruin the whole training process and exponentially increase the time needed. This as quite frequently happened caused me to often just waste tremendous time looking for very small bugs such as the 'mse' vs. 'mae' bug refrenced in RandomForest.

V.3 Improvements

Things to Explore

```
1. try many different algorithms
   * Different types of regressions
   * Use keras
   * Shuffle, mix the dataset
   * more ensemble algorithms

2. Trying this problem as multi class, classificaiton problem in whi
ch we rank the severness of an insurence claim from low prority, med
ium priority, high priority.

3. Use of grid search maybe
```

A Better Solution?

Given all the data points, hidden features, and open endedness of this problem many other betetr solutions must exist. Hence why health records, insurence claimer wealtha and so many more could be better indicatiors of an insurence severity. Thus we can always find better more important features that are highly correlated to insurence claim severity.

In [ ]: