Name: Satishkumar Moparthi

Email address: satishkumarmoparthi@gmail.com

Contact number: 9884227666

Anydesk address: Peridepi (post and vill), Kondapi (MD) , Prakasam (Dt), Andhra Pradesh. PIN - 523273.

Years of Work Experience: 4.5 years of experience on production support.

Date: **17th Aug 2020**

Self Case Study -1: Allstate Claims Severity

"After you have completed the document, please submit it in the classroom in the pdf format."

Please check this video before you get started:
https://www.youtube.com/watch?time_continue=1&v=LBGU1_JO3kg

Overview

*** Write an overview of the case study that you are working on. ***(MINIMUM 200 words)*** ***

The Allstate Corporation is an American-based insurance company ,and has for the past couple of years been developing automated methods of predicting the cost, and hence severity, of claims.The goal of this challenge is to build a model that can help Allstate to predict the server of the claims accurately. At the same time, to provide the important factors that have a strong impact on the severity. With this information, Allstate can propose or adjust more suitable insurance packages for their customers.

As we are measuring the accuracy of a continuous variable, below are some metrics which can be used:

- Mean squared error (considering magnitude , + /-)
- Mean squared logarithmic error
- Mean absolute error (without considering magnitude , L1 norm loss)
- Variance score { The best possible score is 1.0, lower values are worse }
- R-Squared

I chose mean absolute error (MAE) because kaggle competition evaluated on MAE and MAE measures the average magnitude of the errors in a set of predictions, without considering their direction. It's the average over the test sample of the absolute differences between prediction and actual observation where all individual differences have **equal weight.**

MAE formula:

$$MAE = \frac{1}{n}\sum_{j=1}^{n}|y_j - y_j^*| = \frac{1}{n}\sum_{j=1}^{n}|e_j|$$

1. The training set consists of 130 features, 116 are categorical, and 14 are real-valued features. and the loss value for each observation (the output variable to predict) with no missing value and total 188,318 observations.
2. The continuous variables have already been scaled into the interval [0,1] and all have a mean of 0.5 (nearly)
3. From the Pearson correlation matrix, large correlations were spotted between continuous variables 11 and 12 (0.99).
4. we can keep a threshold level and remove all features below that threshold level.
5. From Pairplots , I notice the same as the Pearson correlation, data spread across the planes and not so easily separable one from another.
6. We can make use of a new library called XGboost feature importance (xgbfir) to extract important features from the dataset.

   https://github.com/limexp/xgbfir/

7. Notice that the loss (the one which we need to predict is Right skewed (3.794) , we can apply log on loss to make.It removes large outliers and makes the model perform better.

8. Categorical1 to categorical72 , we have only 2 classes A and B , which are highly imbalanced (In most of the cases, B has very few entities). From Categorical73 to Categorical76 and from Categorical77 to Categorical88, we have 3 (A,B,C) and 4 (A,B,C,D) classes respectively. For other classes we have more than 10 classes.

   binary encoding and multi-level encoding can be used and the other few are below.

9. We can make use of one hot encoding, when the value of the encoded category is produced from its name (A becomes 0, B — 1, Z — 26, AA — 27, and so on).

get_dummies() to directly convert series or data frame into dummy variables (equivalent to one-hot encoder)

Label encoding method converts categorical data into [0,number of classes-1].

tf-idf encoding, TfidfVectorizer. And many more can be used for categorical encoding.

10. As we have to predict continuous variables , We chose to explore four supervised learning methods: Linear Regression, Random Forests, Gradient Boosted Trees, and Support Vector Regression.

Research-Papers/Solutions/Architectures/Kernels

*** Mention the urls of existing research-papers/solutions/kernels on your problem statement and in your own words write a detailed summary for each one of them. If needed you can include images or explain with your own diagrams. it is mandatory to write a brief description about that paper. Without understanding of the resource please don't mention it***

https://medium.com/kaggle-blog/allstate-claims-severity-competition-2nd-place-winners-interview-alexey-noskov-f4e4ce18fcfc

Solution by Alexey Noskov , Kaggle 2nd place winner.

Explanation :

As part of feature engineering Noskov applied log transform on loss as it is right skewed (3.794) with a shift of 200 , which worked well.

Used basic one-hot encoding for some models,when the value of the encoded category is produced from its name (A becomes 0, B — 1, Z — 26, AA — 27, and so on).

First approach was applying SVD on numerical variables and one-hot encoded categorical features. It helped to improve some high-variance models.

Second approach was applying a well known radial basis function (rbf) kernel and finding better performance than the classical K-means algorithm.

Third, was formed by using categorical interaction features, applying lexical encoding to them. These combinations may be easily extracted from XGBoost models by just trying the most important

categorical features, or better, analysing the model dump with the excellent Xgbfi tool.

http://cs229.stanford.edu/proj2017/final-reports/5242012.pdf

Allstate Insurance Claims Severity: A Machine Learning Approach paper published by SUNet, Swedish University Computer Network team.

Explanation :

SUNet team used One Hot Encoding on categorical data.

Team used Root Mean Squared Error (RMSE) to minimise the large errors (optimize to minimize large errors)

Team chosen to explore four supervised learning methods: Linear Regression, Random Forests, Gradient Boosted Trees, and Support Vector Regression

Partitioned the training examples into three sets: training set (80%), cross validation set (10%), and test set (10%).

Used linear regression with no regularization, with L1 regularization, and with L2 regularization.

TABLE I

LINEAR REGRESSION ERROR

Regularization	λ_{opt}	RMSE (CV)	RMSE (Train)
(none)	(N/A)	3470.8	2143.0
L_2	3100	2242.4	2210.4
L_1	0.018	2235.3	2198.5
$0.9*L_1 + 0.1*L_2$	0.002	2235.6	2200.0
$0.8*L_1 + 0.2*L_2$	0.022	2235.8	2200.4

Regression with no regularization seems to be overfitting. With L1 and L2 regularization , the model is slightly overfitting.

L1 regularization produces the best result. So, team added second degree (x + x 2) and third degree (x + x 2 + x 3) terms with L1 regularization, where x is the value of the continuous features.

TABLE II

L_1 REGULARIZATION WITH HIGHER DEGREE TERMS

L_1 Regularization	λ_{opt}	RMSE (CV)	RMSE (Train)
L_1	0.018	2235.3	2198.5
2nd degree terms	0.002	2231.7	2192.8
3rd degree terms	0.002	2230.4	2190.5

Team concluded that the model with the third degree terms gives the best result.

In RF , Using 50 trees (Hyper parameters tuning), the team achieved a final RMSE of 2198.25.

In Gradient Boosted Trees, With optimal parameter values, using max tree depth = 7 and min child weight = 4, we are able to achieve an RMSE of 1959.3.

SUPPORT VECTOR REGRESSION

Another non-linear model we explored was Support Vector Regression. It is similar to the SVM algorithm but for a regression task. It allows us to explore higher dimensional feature space using different kernels. Unsurprisingly, SVR performed better than linear regression. Below are the results.

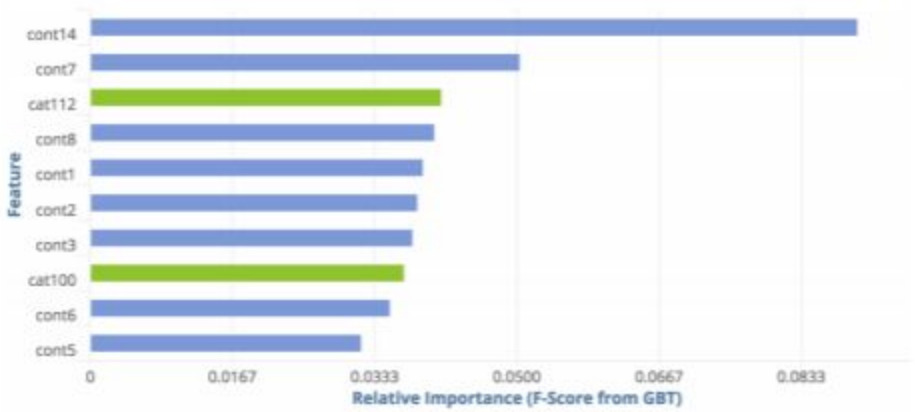
SVR RESULTS

Kernel	RMSE (CV)	RMSE (Training)
Linear	2248	2279
Polynomial	2407	2398
RBF	2162	2129
Sigmoid	2203	2184

One challenge the team faced is of training an SVR model is the long training time, due to the complexity of the model. This makes it harder to experiment, and inhibits the ability to rapidly iterate.

Model Comparison:

GBT gave the minimum error compared to other models.



https://nycdatascience.com/blog/student-works/machine-learning/kaggle-competition-allstate-claims-severity/

Student blog on stacking solution approach from NYC data science academy , Educational institution in New York City, New York.

Explanation :

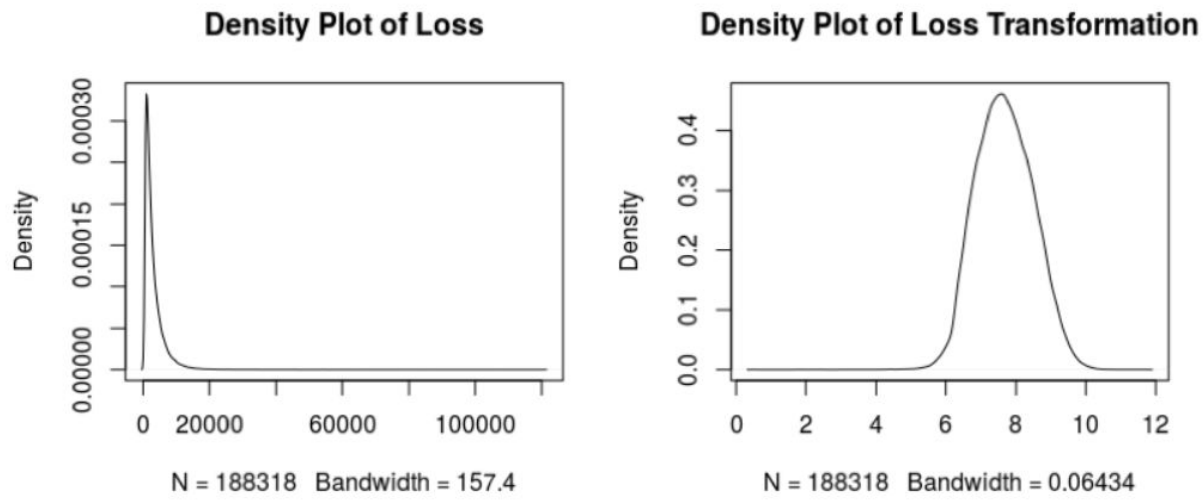
All the features are anonymous, EDA became a very important stage for understanding more insights of the data. This stage is also crucial for the next stage - Feature Engineering

As part of Feature Engineering , used unsupervised learning methods, PCA and K-means to see if there is any possibility to reduce dimensions and cluster the features. After the first attempt, also used three different encoding methods:

- Dummify the categorical variables and keep all of them
- Dummify the categorical variables and drop the variables that have near zero variance
- Dummify the categorical variables and relevel the variables that have high numbers of levels.
- Model implementing: the third stage is implementing single models and fine-turning them. From there, we have selected the best performing models to do model stacking.

EDA Analysis :

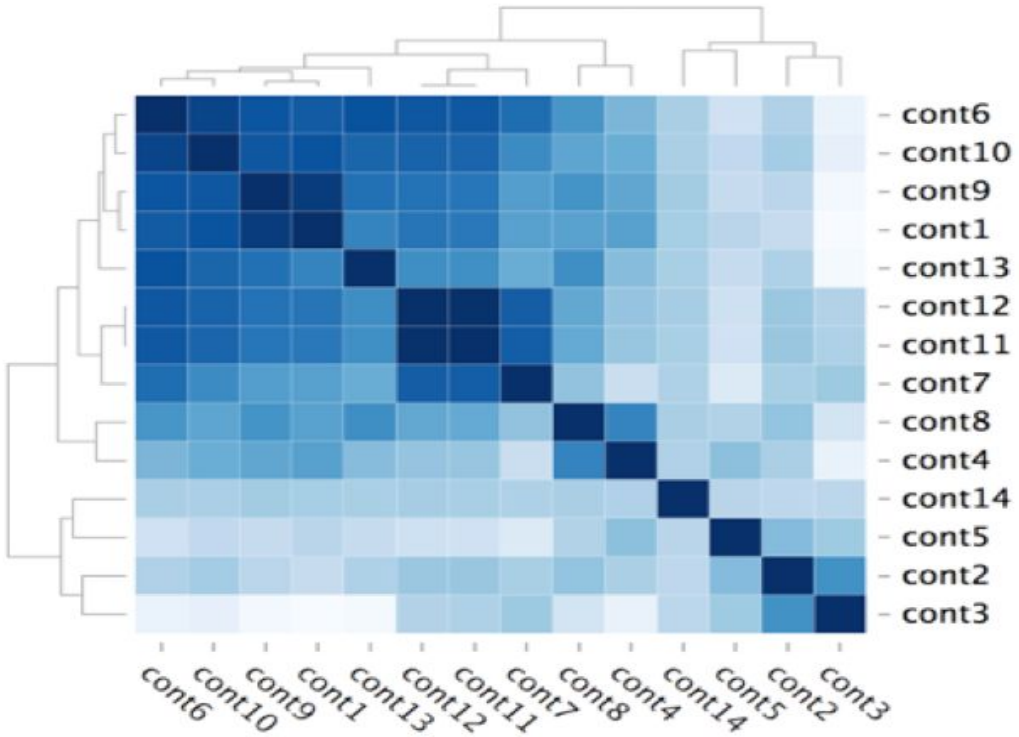
As observed the targeted variable is right skewed , applied log transformation with shift on 'loss', so it can be more normally distributed without changing the order. Second density plot is the result after transformation of the output variable.



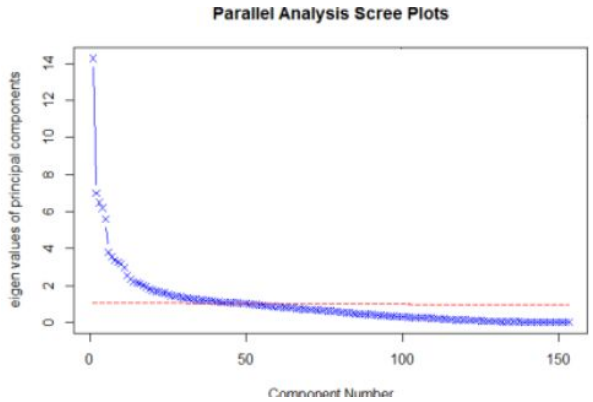
For 116 categorical, the lowest number of levels of variables is 2 levels, and the highest number of levels that appears in the variables is 326 levels. After further investigation, another important information about the categorical variables is that some variables contain levels only in the test set, but not in the training set. The chart below lists out the unique levels for the categorical variables that have this behavior.

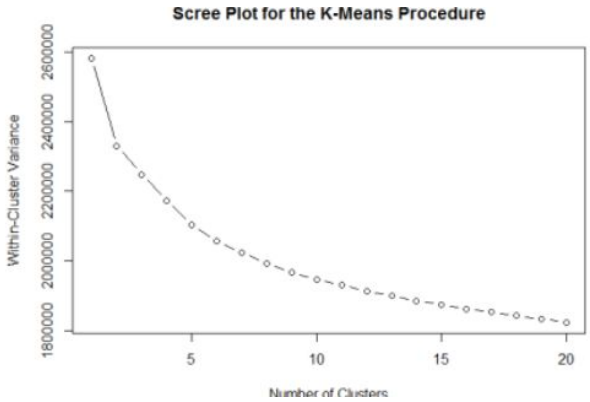
Variable	Train	Test	Variable	Train	Test
cat89	I	F	cat105	R S	
cat90	G		cat106		Q
cat92	F	G E	cat109	BM CJ BV BY BT B BF BP J AG AK	AD
cat96		H	cat110	BK H BN DV EI BD BI AN AF CB EH	BH CA EN
cat99		U	cat111	D	L
cat101	N U		cat113	BE T AC	AA R
cat102	H J		cat114	X	
cat103		M	cat116	BI V BL X FS P GQ AY MF JD AH EV CC AB W AM IK AT JO AS JN BF DY IB EQ JT AP MB C IO DQ HO MT FO JI FN HU IX	AQ EM FY AI N ET KO BJ IW DB LP MX BR BH JS ER A BN BE IS LS HS EX

The correlation plot provides the insight among the variables. The plot indicates that there are some strong relationship between some variables. This is important information for building model later on.

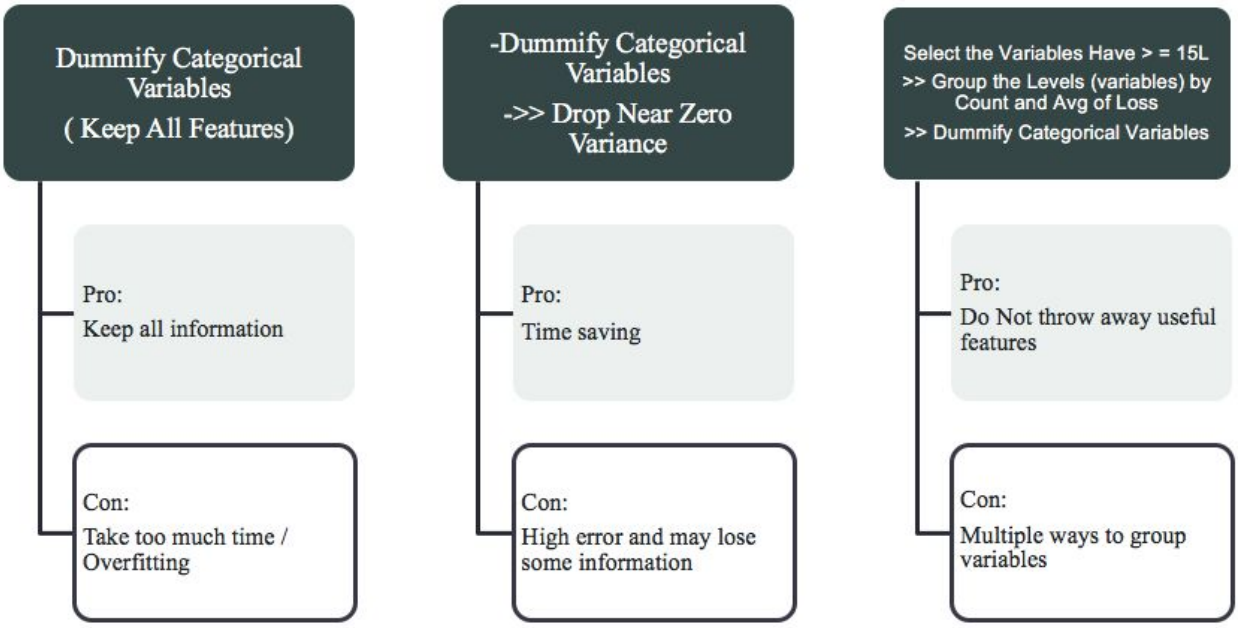


PCA and K-means didn't work well on dimensional reduction. For PCA, with about 48 principal components, around 88% variance is explained, under 95%. For K-Means, even with 20 clusters, the within-cluster variance is still very large.





Feature Engineering :



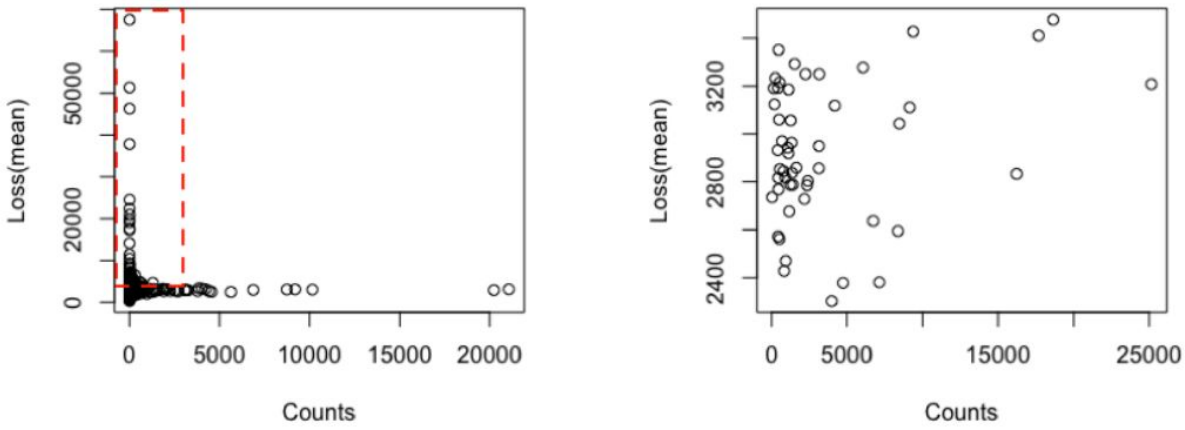
As the first attempt, Team broke categorical variables with many levels into several dummy variables. It means that for a categorical variable, when one of its levels has one observation, it becomes a dummy variable having (n-1) zeroes. Hence, after the first step, there were many dummy variables which were near zero variance (NZV) predictors. Keep all of them or drop the features which had NZV. They used both of them to build models, however, the results were not good.

On the other hand, the team used the function nearZeroVar from the Caret package to remove the NZV predictor. By default, a predictor is classified as NZV if the percentage of unique values in the samples is less than 5%. The advantage was that this method saved us a lot of time to build models, however, the drawback was that it increased the error and resulted in losing some potentially important information.

In order to balance between time consumption and error reducing, this project regroups those categorical variables over 15 levels, to keep as much information as possible.

Why are 15 levels here a good cutting point ?

When using near zero variance to drop features, for categorical variables, it removes those levels with less than 5 percent of observations. For such a large dataset, 5% means around 9500 observations. Even if a level having 9000 observations, which must have some useful information, will be removed from the dataset. For variables like cat116, which has 326 levels, only 3 levels are kept after near zero variance feature engineering. Even in an ideal case, only 19 out of 326 levels can be used for further machine learning, which is certainly not desirable. Meanwhile, this 'ideal' situation also tells the fact that 20 is not a good cutting point. To be more conservative, this project determines to regroup categorical variables over 15 levels. In other words, keep over 5 percent of observations in each level.



Here are two graphs. Each graph represents one categorical variable. Each point here is one level. The x-axis is counted. That is how many observations in each level. The y-axis is the mean of the loss for each level. An interesting finding is that, in most cases, the scatterplot is like two straight lines, one vertical, one horizontal, like in the left graph. The right graph, Cat112, is an exception.

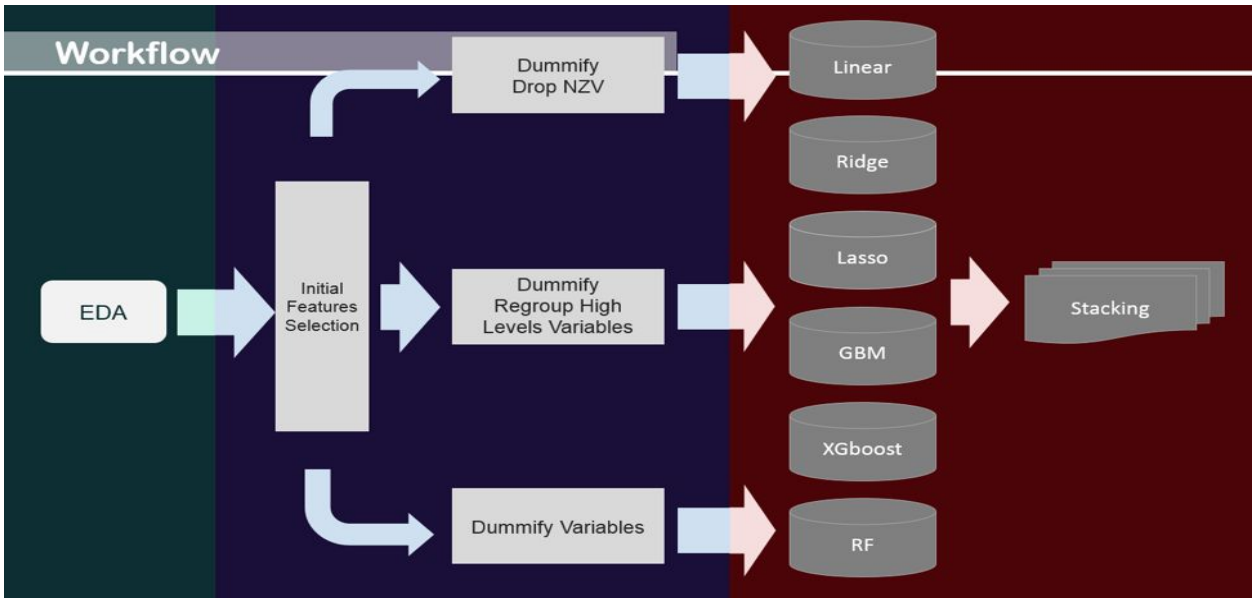
However, the ways to regroup are using the same idea: first, keep the original levels having over 5% of observations, and then split them based on the average loss and counts, regroup them into high loss, low count group, low loss low count group, low loss low count group, and so on and so forth.

By doing feature engineering in this way, it also manually builds a connection between loss and those categorical variables. The next step is the third round of machine learning.

Models :

Model	Features Engineering	Parameters	RMSE
MLR	Drop NZV		0.5765
MLR	Drop High Cor + New Group		0.5655
Ridge	New Group	Lambda: 1e-05	0.5641
Lasso	New Group	Lambda: 1.592283e-05	0.5641
RandomForest	Drop NZV	Ntree: 500 mtry = 51	0.5779
GBM	Drop NZV	Ntree: 2640 n.Minobsev: 20	0.5116
XGB (xgbTree)	New Group	nrounds = 300 max_depth = 4 eta = 0.3 gamma = 0.2 colsample_bytree =0.6 min_child_weight = 1 subsample = 0.85	0.5436

According to the RMSEs of each model, the best individual model is the gradient boosting machine learning. The rationale behind this best performance is multiple attempts of cross-validation and the third method of feature engineering - new group. However, model stacking gives the best predictive accuracy. The greater the differences of the learning algorithms, the better performance the stacking model provides.

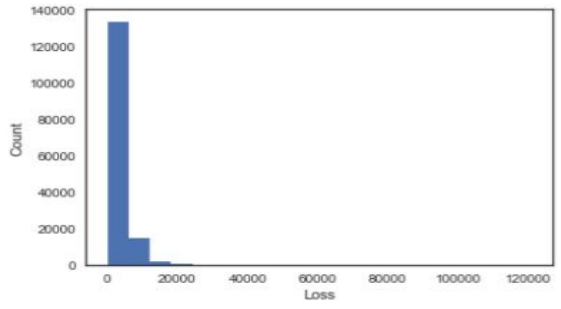


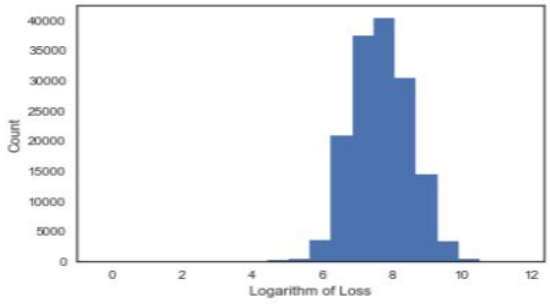
https://notendur.hi.is/heh89/Allstate_Claims_Severity.pdf

Solution paper by students, Bergþór Traustason and Helgi Hilmarsson , at University of Iceland under the guidance of professor Stein Guðmundsson.

Explanation :

Histograms were made to analyse distributions of the variables to get a better understanding of the data visualization. From the histogram it is quite obvious that the dataset is very asymmetrical which can lead to various problems. To fix this, logarithm transformation is applied to the loss. The result is a much more symmetrical distribution.





Plotted correlation matrix and decided to remove cont12 from the dataset along with the id column as high correlation observed between cont11 and cont12.

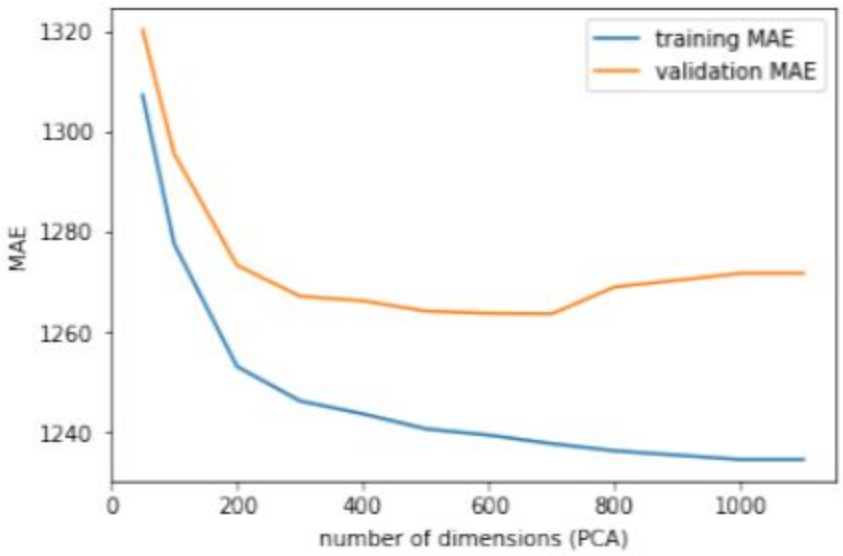
Used OHE to convert the categorical data to numerical data. {Letters: [A,B,B,D]} -> {Letters_A: [1,0,0,0], Letters_B: [0,1,1,0], Letters_C: [0,0,0,1]}. For those specific prediction models, a "multi-label" encoding method was used: {Letters: [A,B,B,D]} -> {Letters: [1,2,2,3]}. For neural networks however, OHE had no issues and was in fact the superior encoding method.

After using OHE on Cat dimensions increased to 1190. At first, PCA was used to reduce the dimensions to 2 components and to see if PCA was generally relevant. An effort was made to make a T-SNE dimensionality reduction without success. The Kernel would always shut down even though the dimensions had already been reduced to a small number.

Models :

Linear Regression (LR). The most simple regression model there is. This model had trouble with the OHE-data. The two ways around this were using multi-labeled data and OHE-data with PCA component reduction. Below, training and validation errors of the regression are plotted against the number of components in order to find an optimal number of components in the PCA method.

Overfitting model.



Ridge and Lasso Regression. Small improvements of the LR, adding penalizations to the model

to avoid overfitting. But L2 regularization seems to be overfitting and L1 works fine here.

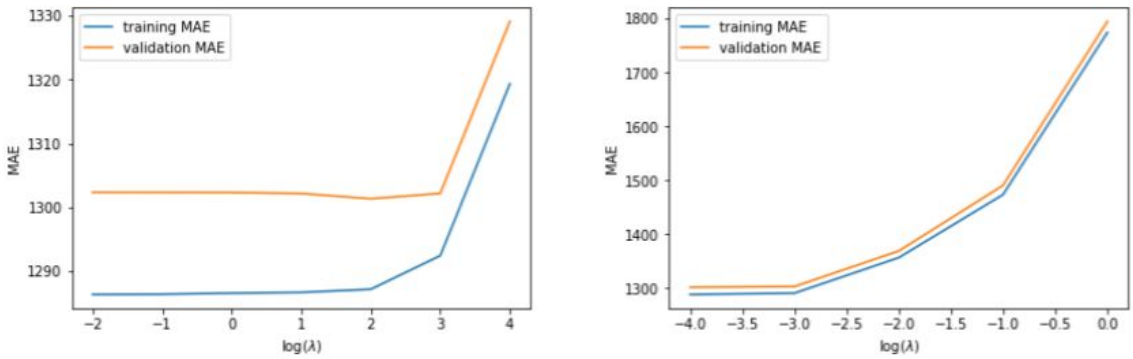


Figure 4: Optimization of the hyperparameter λ in for the Ridge regression (left) and the Lasso regression (right)

Support Vector Machines (SVM). Resemble some kind of a linear regression, but maximizes the gap between two classifiers on each side of the regression line while holding the classification error down. This can be extended to a regression model (SVR) which was used in this project. The optimization of the hyperparameter C for the OHE data is shown below. Training a SVR model is a very time consuming process and in early stages. That is why the same method was used for validating the optimization of the hyperparameters as was done for the previous models (instead of time consuming things like cross validation). The dataset is very big and the OHE data took about 100x more time than the multi-labeled one. Model seems to be overfitting.

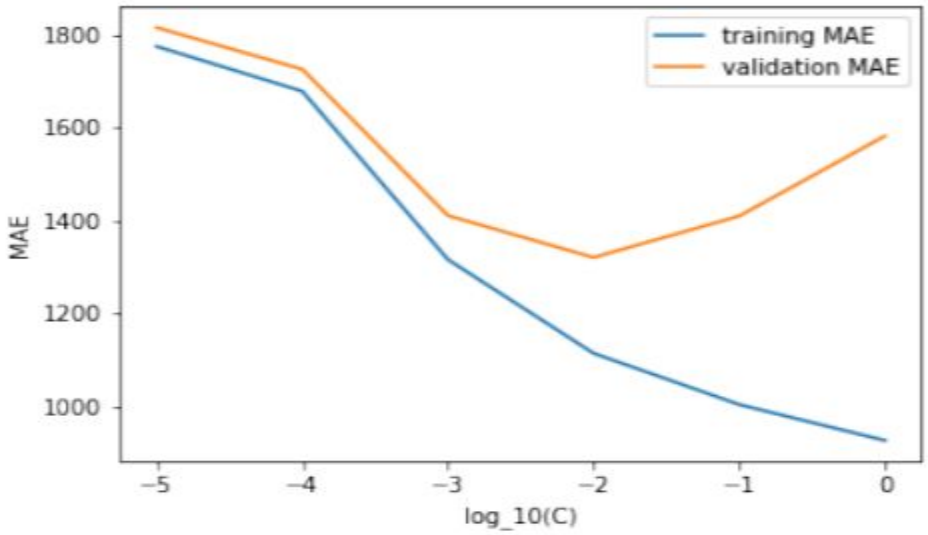
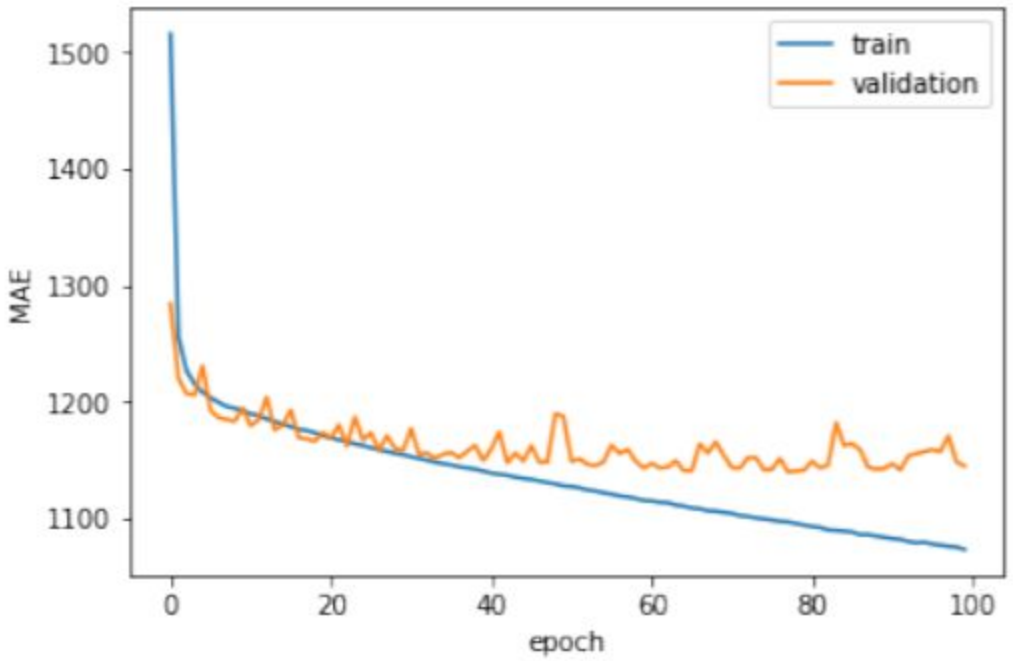


Figure 5: Optimization of the hyperparameter C

.**Random Forest (RF)**. Tree method that uses random selection of features to avoid overfitting. The best results were achieved with the values max_depth = 14 and max_features=73. Using 50 trees and these values for the parameters in the Random Forest method an MAE score of 1222.878 was achieved.

Neural Networks (NN). The most popular model these days. Mimics the brain in using neurons to process information and learn from data.Training and Validation MAE plotted against the number of epochs. An obvious overfitting starts to happen at around epoch 30.



Some Feature Engineering Technique:

Almost all solutions are similar. Now we can focus on some feature engineering techniques. From the above solutions, we saw some feature engineering techniques like

- One Hot Encoding
- lexical encoding
- Dumify the Categorical variable
- Dumify the Categorical variable and dropping near zero variance
- Dumify the Categorical variable and Grouping levels/ class by count and average by loss.
- Extracting most important categorical features, or better by using Xgbfi tool
- Used log transformation on loss (target value) to convert into normal distribution form.
- Applied Correlation on Continuous variables.

Here are few more feature engineering techniques to apply in our case study,

https://www.kaggle.com/tilii7/bias-correction-xgboost

In this solution, box-cox transformation on skewed data which are having skewness `> 0.25.` Transformation technique is useful to stabilize variance, make the data more normal distribution-like, improve the validity of measures of association. The problem with the Box-Cox Transformation is estimating lambda. This value will depend on the existing data, and should be considered when performing cross validation on out of sample datasets.

Used pandas factorize instead of OHE , lexical for categorical.

Used StandardScaler function , Standardize features by removing the mean and scaling to unit variance.

With above feature engineering and hyper parameter tuning using XGBoost , good MAE score.

Average eval-MAE: 1146.108520 (best score :: 1113.12)

https://www.kaggle.com/sharmasanthosh/exploratory-study-on-ml-algorithms

In this solution, applied log1p function which applies log(1+x) on loss { target variable }

Removed few continuous variables which are having correlation less than 0.5

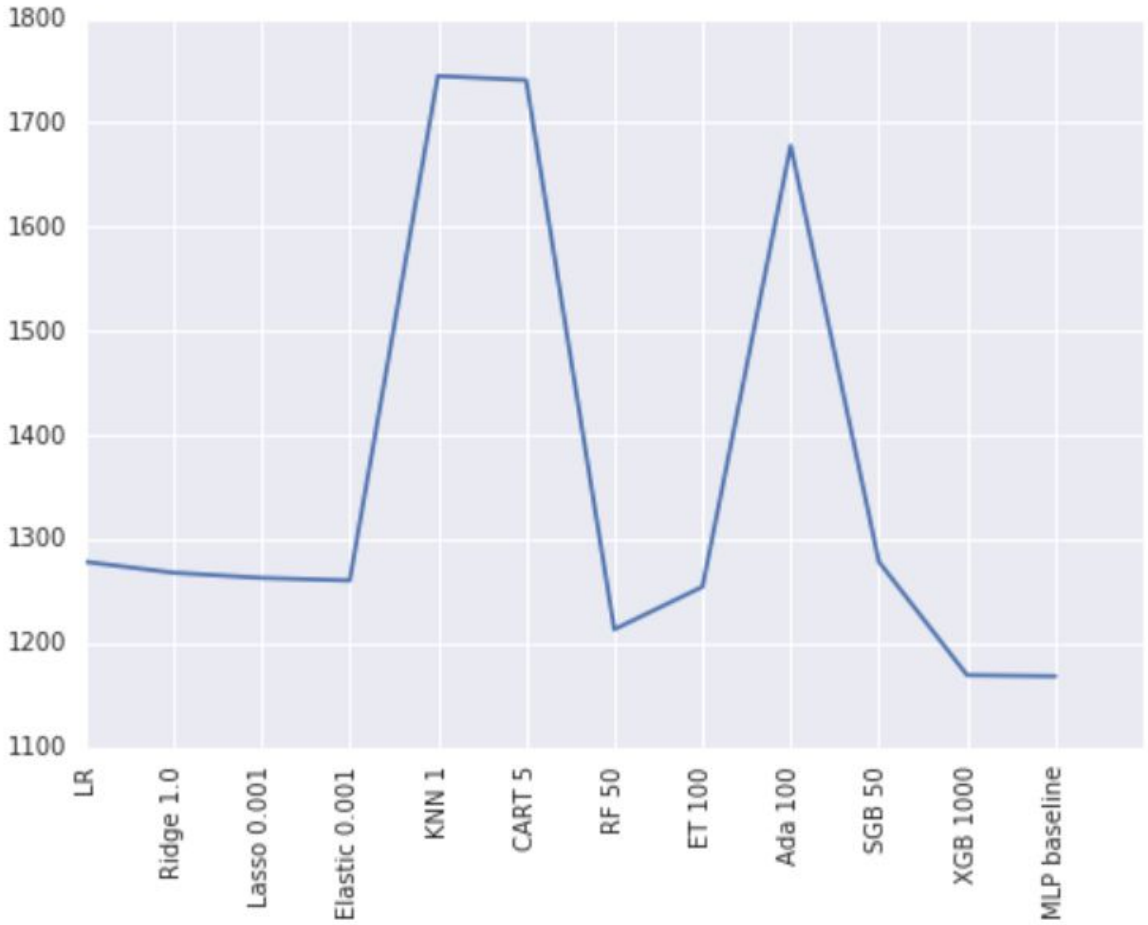
Removed high correlated variables. General practice is to exclude highly correlated features while running linear regression.

```
#cont11 and cont12 give an almost linear pattern...one must be removed

#cont1 and cont9 are highly correlated ...either of them could be
safely removed
#cont6 and cont10 show very good correlation too
```

Applied One Hot Encoding of categorical data

With above feature engineering and hyper parameter tuning using almost all regressions , MLP baseline and XG boost worked well. Please find the below screenshot.



https://www.kaggle.com/cuijamm/allstate-claims-severity-score-1113-12994 (Kaggle Competition best score 1113.12)

Used log transformation on target { loss } variable as it is right skewed to transform into normal distribution.

90+% of loss falls under 40000 , so removed all rows which are greater loss value (more than 40K)

Applied label encoding in categorical features.

Here data trained with three algorithms CatBoostRegressor , XGBoost and LightGBM.

CatBoost is a recently open-source machine learning algorithm from Yandex. CatBoostRegressor is a machine learning library to handle categorical (CAT) data automatically. "CatBoost" name comes from two words "Category" and "Boosting". It is especially powerful in two ways:

- It yields state-of-the-art results without extensive data training typically required by other machine learning methods, and
- Provides powerful out-of-the-box support for the more descriptive data formats that accompany many business problems.

https://catboost.ai/docs/concepts/algorithm-main-stages_cat-to-numberic.html

We can use CatBoost without any explicit pre-processing to convert categories into numbers. CatBoost converts categorical values into numbers using various statistics on combinations of categorical features and combinations of categorical and numerical features.

He took the average of the predicted value from three models.

A summary of solutions

https://zhouyu-sadahanu.com/2017/04/24/a-summary-of-solutions-for-allstate-claims-severity-kaggle-challenge-1-feature-engineering/

This is a feature Engineering blog written by Zhou Yu taking 1st , 2nd and 3rd place kaggle winer solutions.

For the raw data, among all columns, 116 out of 130 are categorical data. So, simply converted each category into an integer. The downside of this method is that the integer value is sometimes meaningless.

train_test[c].astype('category').cat.codes

1st Place Solution

Category Embedding with NN ,TF-IDF on categorical features, Interaction between categorical features

2nd Place Solution (Already explained above)

Different categorical encodings in different models – lexical, dummy, Bayes, applying SVD to numeric + dummy categorical (for example, CV of libfm model with SVD improved from 1196 to 1177)", what helped for non-tree models is clustering data and then transforming it to cluster-distance space and applying RBF function. It was especially useful for linear regression (improved from 1237 to 1202), but also good for NN models.In my late XGB models I also used categorical combinations, the best way to select combinations was using excellent Xgbfi tool –

just take dump of good XGB model without combinations, run Xgbfi and take some number of 2-way interactions.

Faron's 3rd Place Solution

One hot encoding, label encoding, creation of Interactions, target loss encoding, tf-idf encoding, XGB-embedding, Keras-embedding, removal of rare values. So many technical terms! To focus our efforts, here I will just focus on these three terms: encoding, Interaction and embedding.

First, encoding. Usually encoding refers to transforming categorical data into meaningful numerical values. The methods mentioned by those top Kagglers include: One-Hot(dummy), label encoding (lexical) and target loss encoding.

One-hot encoding method simply converts categorical data into multiple columns each is either 0 or 1. The latest version of pandas has a function get_dummies() to directly convert series or data frames into dummy variables (equivalent to one-hot encoder).

Second, Interaction：It seems that sometimes adding interactive terms of important features may improve the model performance. For example, if feature b1 and b2 are both important, then adding b1*b2 may make sense too. Xgbfir library is used for interactive features.

Third, Embedding: Generally, feature embedding means using models to convert features to some sort of intermediate results. There is no standard way to embed features. In general, from reading through those posts, it seems to be a good practice to convert dummy variables into more continuous features through embedding.

Those feature engineering methods all make sense in some way to improve prediction results, but using them is probably the best way to learn.

First Cut Approach

*** Explain in steps about how you want to approach this problem and the initial experiments that you want to do. ***(MINIMUM 200 words)*** ***

*** When you are doing the basic EDA and building the First Cut Approach you should not refer any blogs or papers ***

Due to all the features being anonymous, EDA became a very important stage for us to understand more insights of the data.

From research and kaggle solutions I have learned few new techniques,

1. Using Box-Cox transformation on skewed data works better.

https://www.kaggle.com/misfyre/encoding-feature-comb-modkzs-1108-72665/code (1108 MAE)

2. We can use CatBoostRegressor which handles categorical (CAT) data automatically.
3. By using lexical encoding and label encoding is better compared to one hot encoding
4. Boosting algorithms worked fine compared Linear, Lasso and Ridge algorithms.
5. As the continuous variables have already been scaled into the interval [0,1] and all have a mean of 0.5 (nearly). But applying Standard Scalar/ Minmax scalar gives good results.

Approach 1:

From the research, it is clear that lexical encoding and removing low and high correlated features gave best results with applying box-cox on skewness with log transform on target variable.

Applying lexical encoding on categorical features and by removing the low and high correlated continuous features.

Applying log transformation on targeted variable and box-cox transform on features having skewness greater than threshold level (like > 0.25)

Applying XGBoos algorithm with hyper parameters on final train data.

Approach 2:

Extracting categorical interaction features, applying lexical encoding to them. These combinations may be easily extracted from XGBoost models by just trying the most important categorical features, or better, analysing the model dump with the excellent Xgbfi tool.

Training XGBoost model with interaction features and all continuous features after fixing skewness issues. If MAE is not upto mark , we can train the model after removing low and high correlated features.

Approach 3:

Instead of using the OHE, label encoding and lexical encoding , will try with TF IDF values for categorical features. Interaction between categorical features.

Training with different models and stacking all of them finally.

Approach 4:

Get the best feature by applying Random Forest after feature engineering and train model with best features instead of all.

Below is the blueprint of training a model.

Step -1 : Data Exploration

The dataset contains 2 .csv files with information needed to make a forecast. They are::

Variables in train.csv and test.csv:

1. **id:** the id of a couple of questions from the training set, int64(1)
2. **cat1 to cat116:** category variables , object(116)
3. **cont1 to cont14:** continuous variables, float64(15) { Cont1 to cont14 + loss }
4. **loss:** the amount that the company has to pay for a particular claim. This is the target variable. In test.csv, the loss is not present, since we will predict this.

Step -2 : Data Cleaning

1. Check missing values
2. Check duplicate values
3. Check outliers { skew() , Box-plots , violin plots }
4. Convert Types , categorical string values to numeric values

There are 116 categories with non-alphanumeric values, most of the machine learning algorithms don't work with alpha numeric values. So, lets convert it into numeric values

Step - 3 : Data Visualization

1. Continuous feature
    - Visualize data continuous
    - Correlation data continuous
    - Matrix correlation: continuous features
2. Categorical feature
    - Transform categorical feature
    - Visualize data categorical
    - Matrix correlation: categorical features

Step - 4 : Training the model

1. Split train and test
2. The variable (Shift)is applied to the log transformation to the target variable as it right skewed.
3. The models in this project use the mean absolute error (MAE) between the predicted loss and the actual loss for each claim in the test set. The goal was to minimize the MAE in our model's predictions.
4. Split training data into validation data (K-Folds Cross Validation).
5. The training and test is in function because guaranteed reuse.
6. The predictions run in validation set in each fold.
7. Guaranteed with array is the inverse of the log transformation about column loss.
8. Calculate MAE

Step - 5 : Algorithms with Hyperparameters.

1. Linear Regression (Linear algo)
2. Ridge Regression (Linear algo)
3. Lasso Linear Regression (Linear algo)
4. Elastic Net Regression (Linear algo)
5. SVM (Non-linear algo)
6. Bagged Decision Trees (Bagging)
7. Random Forest (Bagging)
8. Extra Trees (Bagging)
9. AdaBoost (Boosting)
10. Stochastic Gradient Boosting (Boosting)
11. MLP (Deep Learning)
12. XGBoost

We can make use of Model Stacking as well and with best algorithms , which give low MAE.

Notes when you build your final notebook:

1. You should not train any model either it can be a ML model or DL model or Countvectorizer or even simple StandardScalar

2. You should not read train data files
3. The function1 takes only one argument “X” (a single data points i.e 1*d feature) and the inside the function you will preprocess data point similar to the process you did while you featurize your train data
    a. Ex: consider you are doing taxi demand prediction case study (problem definition: given a time and location predict the number of pickups that can happen)
    b. so in your final notebook, you need to pass only those two values
    c. def final(X):
        preprocess data i.e data cleaning, filling missing values etc
        compute features based on this X
        use pre trained model
        return predicted outputs
    final([time, location])
    d. in the instructions, we have mentioned two functions one with original values and one without it
    e. final([time, location]) # in this function you need to return the predictions, no need to compute the metric
    f. final(set of [time, location] values, corresponding Y values) # when you pass the Y values, we can compute the error metric(Y, y_predict)
4. After you have preprocessed the data point you will featurize it, with the help of trained vectorizers or methods you have followed for your train data
5. Assume this function is like you are productionizing the best model you have built, you need to measure the time for predicting and report the time. Make sure you keep the time as low as possible
6. Check this live session: https://www.appliedaicourse.com/lecture/11/applied-machine-learning-online-course/4148/hands-on-live-session-deploy-an-ml-model-using-apis-on-aws/5/module-5-feature-engineering-productionization-and-deployment-of-ml-models