Allstate Claim Severity

## Domain Description:

The aim of the project is to predict the loss that the insurance company incurs based on the severity of the claims by the customers. More accurate the prediction is, lesser the chances of loss for the company.

## Description of the algorithm:

During the implementation, I tried the following algorithms: linear regression model, gradient boosting model, decision tree with gradient boosting and XGBoost model. Out of the four XGBoost with estimators greater than 500 proved to be efficient since it gave the minimum MAE (Mean Absolute Error) on training data. Decision tree with gradient boosting model was very slow with higher number of estimators and predicted values with high MAE for lesser number of estimators.

Python’s sklearn library has the implementation of all the algorithms and it also includes the implementation of XGBoost. XGBoost stands for eXtreme Gradient Boosting, a variant of Gradient Boosting algorithm often known as *blackbox*, since everyone knows that its accurate and fast but how it works is not so trivial. You can find more about the algorithm here: <http://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/> .

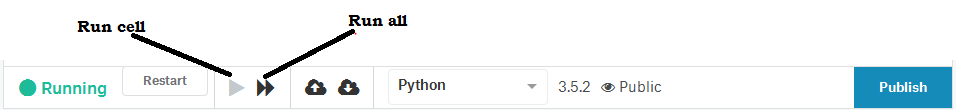
## Code flow:

The data consists of categorical and continuous values. It’s easy to predict values for continuous values, but for categorical values, they need to be encoded into numerical values. The encoding method what I have followed is a simple labeling technique, where in the distinct categorical values are collected in a dictionary and for each category a number starting from 1 is assigned. So there will be ‘n’ unique labels in total and will be replaced in the actual data read from the csv file. The encoding occurs for both training and testing data. Encoding is done is modules named **categoryEncoder** and **testCatEncoder** for train and test data respectively. I should agree that I haven’t used one hot encoder based on the comments in the link: <https://www.kaggle.com/c/allstate-claims-severity/forums/t/24480/one-hot-coding-for-categorical-variables> .

## MAE calculation:

The training data is divided into a ratio of 80:20. That is, 80% of data will be used for training and 20% of data will be used for testing. The target column, that is **loss,** is also divided into same parts. Train the model using the 80% data and try predicting the 20% target. Using the available **mean\_absolute\_error** package from sklearn, the MAE calculated (which varies between 1150 and 1199).

## How to run the ipynb file:

1. Go to the <https://www.kaggle.com/c/allstate-claims-severity> and in the side panel named **Dashboard,** under kernels click on New Notebook option.
2. Click upload button. It looks something like in the screenshot shown below: 
3. Locate the .ipynb file in the submission and upload it to the newly created file on cloud.
4. There are two options to run the file **Run cell** and **Run all** which is shown as below: ****
5. Click on **Run all** option and wait for like 10-15 mins.
6. After completion of each module defined in the file, there will be an output **“<modulename> done”,** just to indicate which cell is recently completed.
7. After the execution of the last cell, you’ll find MAE (Mean Absolute Error) of the training data.
8. There is an optional module commented out in order to write the predicted values to a csv file named **result.csv**, which I will be sending as a part of submission. If you uncomment and run it, it will produce a file named **result.csv** under output section of the notebook.

## References:

1. <http://xgboost.readthedocs.io/en/latest/python/python_api.html>
2. <https://www.kaggle.com/c/allstate-claims-severity/forums/t/24480/one-hot-coding-for-categorical-variables?forumMessageId=140006#post140006>