Overview: Insurance claims are important financial matters that may appear in anyone's daily life at any time. Insurances are purchased to protect any property or valuable things in case they are damaged in an accident. There is also life insurance that is the compensation to loved ones after someone's death to give financial protection.

Auto insurance is the most frequently purchased insurance among all to protect the vehicle, passengers, pedestrians and other properties those could be damaged in car accidents. Driving without proper insurance protection is the violation of law. Insurance is purchased to cover upto a maximum amount of the damage, but when claims are filed, insurance companies decide on the actual damage within the limit. That decided amount after assessment are paid to compensate for the loss.

This project is about deciding insurance claims in dollar value. There are features given associated with the incidents and claims are predicted based on that. The dataset are taken from one of the Kaggle's competition launched by "Allstate" insurance company to predict the insurance claims. The link is given below: https://www.kaggle.com/c/allstate-claims-severity

Problem Statement: It is very hard to know the absolute claim value just by knowing some informations. That's why machine learning can help by building a model from prior occurrences and using it to predict in future. It is not needed to know the absolute value to know the severity of the claim. Rather relative values are sufficient to know. There are total 130 features given in the dataset (excluding 'loss value' and 'id'). 'Loss' in dollar amount need to be predicted using the features given. Among the features, there are 116 categorical features and 14 continuous values features. In short, categorical features are the ones to have only some allowed discrete values while continuous features may have any value.

Values of categorical features are in letters like ‘A’, ‘B’, ‘C’ etc. So to use them to learn a model, these features need to be encoded to digits using sklearn label encoder.
As there are 130 features, it will be needed to know if all the features are important to predict the insurance claim or if only selected features help to learn the model quickly or accurately. This will be done using sklearn pipeline module. As it will be attempted to predict the insurance claim which can have any dollar amount, so this is a supervised learning regression problem. That’s why sklearn regression algorithms need to be attempted to get the model.

Metrics: The coefficient of determination or “R squared” will be used as performance metric for this regression problem. The definition of this metric is:

$$R^2 \equiv 1 - \frac{SS_{\text{res}}}{SS_{\text{tot}}}.$$

where,

$$SS_{\text{tot}} = \sum_i (y_i - \bar{y})^2,$$

and

$$SS_{\text{res}} = \sum_i (y_i - f_i)^2 = \sum_i e_i^2$$

Here, y_i is the actual output and f_i is the prediction from model. y_bar is the mean.

As this is a regression problem with the goal to predict the insurance claim as accurately as possible, “R squared” will measure how close the model’s outputs are compared to actual value. Closer the value to ‘1’, better model it will be. For a good model, SS_{res} will be very small compared to SS_{tot} and R^2 will be close to 1, as shown in the above equation. So, in case of this project, if insurance claim predicted by the model is exactly equal to the actual value for all examples, R^2 value will be 1. If it is 0, then the model will be equivalent to just averaging all the

claims values, that means it will fail to predict any variance. So that model will fail and there will be no use of the model. So goal is to increase R^2 value as high as possible and have the model to give as much variance as possible.

Data Exploration: There are 116 categorical features those will be encoded using label encoder. Also there are 14 continuous valued features. Feature scaling is already done for these continuous valued features, as their values are between 0 and 1. The statistics for insurance claim or 'loss' are given below:

Total number of instances in the dataset: 188318
Maximum loss: $121,012.25
Minimum loss: $0.67
Mean loss: $3,037.34
Median loss: $2,115.57
Standard deviation of loss: $2,904.08

So both the mean and median values are between $2000 to $3100, but the maximum loss is $121,012.25. Also the standard deviation is $2,904.08. So most of the claims are within few thousands, there are only few with very high dollar value.

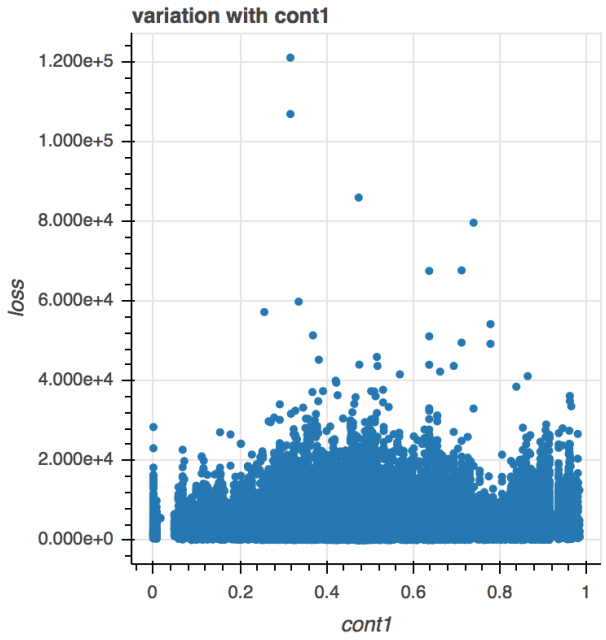
The first row of dataset are below as an example of the entries in the dataset:

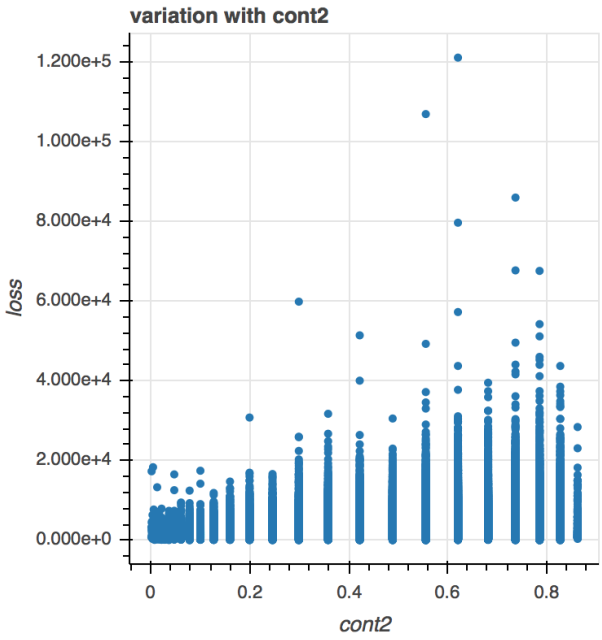
id,cat1,cat2,cat3,cat4,cat5,cat6,cat7,cat8,cat9,cat10,cat11,cat12,cat13,cat14,cat15
,cat16,cat17,cat18,cat19,cat20,cat21,cat22,cat23,cat24,cat25,cat26,cat27,cat28,c
at29,cat30,cat31,cat32,cat33,cat34,cat35,cat36,cat37,cat38,cat39,cat40,cat41,cat
42,cat43,cat44,cat45,cat46,cat47,cat48,cat49,cat50,cat51,cat52,cat53,cat54,cat55
,cat56,cat57,cat58,cat59,cat60,cat61,cat62,cat63,cat64,cat65,cat66,cat67,cat68,c
at69,cat70,cat71,cat72,cat73,cat74,cat75,cat76,cat77,cat78,cat79,cat80,cat81,cat
82,cat83,cat84,cat85,cat86,cat87,cat88,cat89,cat90,cat91,cat92,cat93,cat94,cat95
,cat96,cat97,cat98,cat99,cat100,cat101,cat102,cat103,cat104,cat105,cat106,cat10
7,cat108,cat109,cat110,cat111,cat112,cat113,cat114,cat115,cat116,cont1,cont2,c

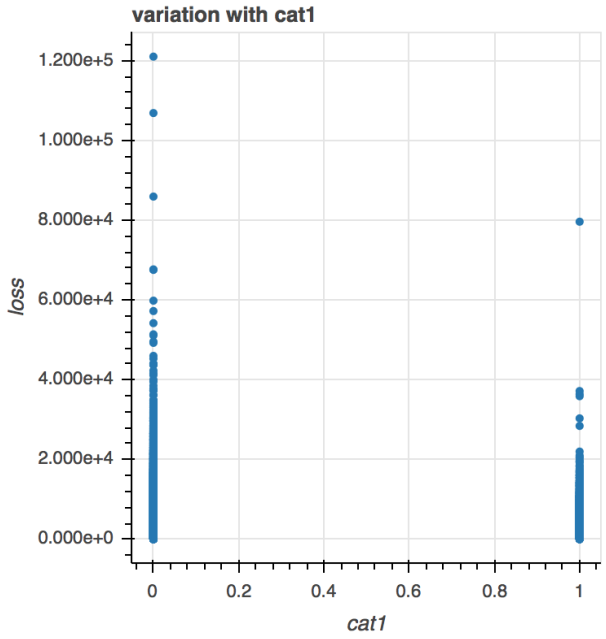
ont3,cont4,cont5,cont6,cont7,cont8,cont9,cont10,cont11,cont12,cont13,cont14,los
s
1,A,B,A,B,A,A,A,A,B,A,B,A,A,A,A,A,A,A,A,A,A,A,B,A,A,A,A,A,A,A,A,A,A,A,A,A,A
,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,A,
D,B,B,D,D,B,D,C,B,D,B,A,A,A,A,A,D,B,C,E,A,C,T,B,G,A,A,I,E,G,J,G,BU,BC,C,AS,
S,A,O,LB,0.7263,0.245921,0.187583,0.789639,0.310061,0.718367,0.33506,0.302
6,0.67135,0.8351,0.569745,0.594646,0.822493,0.714843,2213.18

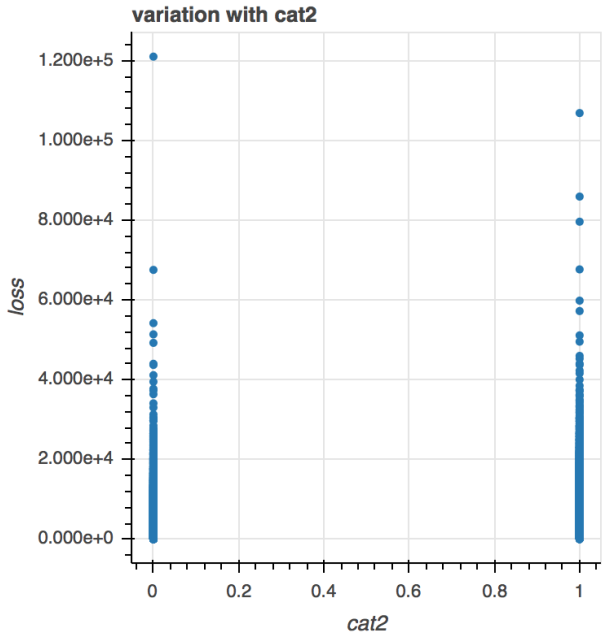
There are no missing value in the dataset. Also the meaning of features are not revealed in the dataset. That means they are encoded and it is not possible to make any decision intuitively.

Exploratory Visualization: The variation of ‘loss’ with continuous features ‘cont1’, ‘cont2’ and categorical features ‘cat1’, ‘cat2’ are shown below.



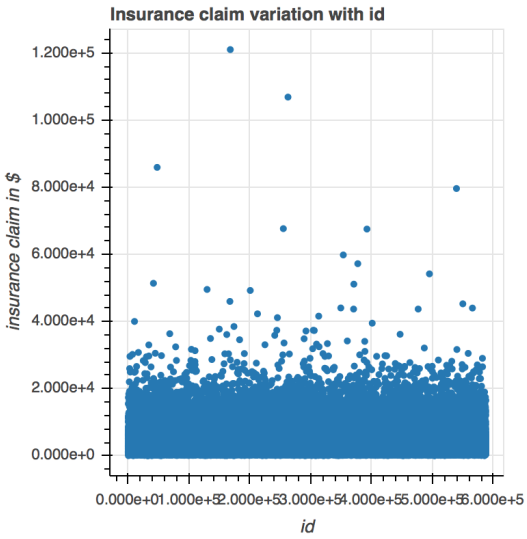






It is clearly seen from the above figures that, none of these features have clear variance with the insurance claim. So the combinations of features have to establish a model. As there are many features in the dataset, not all the diagrams are presented here. But feature selections have to be attempted to get important features.

Another observation here is the plot of all claims in the dataset.



From the above plot, it is interesting to see that most of the claims are within $20,000, There are some between $20,000 to $40,000. But very few are more than $30,000. So all the entries above $30,000 will be removed from the dataset considering them as noise, as there are not many of them, so they will be very hard to predict.

Algorithms and Techniques: LabelEncoder from sklearn will be used to encode the categorical features. Then Linear Regressor will be used to select features. After feature selections, Random Forest Regressor, Linear Regressor, K Nearest Neighbour Regressor and Polynomial Regressor will be tried to learn the model. These algorithms are chosen as they are fast to create the model as the dataset is very large and many iterations will be possible to run to get the best model within reasonable time. Also, non-linearity in the dataset will be modeled by algorithms like random forest and polynomial regressor, that might be present in the dataset.

Random Forest algorithm 'K' number of different trees. A random subset of training examples are picked each time a tree is created. When trees are created, random attributes are picked up instead of all attributes. To predict the value for unknown output, each tree is used to evaluate and average is taken for the output. Linear regression draws a line to reduce mean squared error(MSE). So the coefficients of each feature are chosen and a line is drawn reducing MSE. K Nearest neighbour takes the average value of nearest K examples in the training set and that becomes the value of unknown output. Polynomial regressor converts each feature to a certain degree, like 2,3, etc. instead of 1, as in linear regression. For example, if X1,X2 are features, in polynomial regression, $X1^2$, $X2^2$, X1X2 are also used as features with X1, X2. Then coefficients of new features are also obtained to create the model.

The best model will be the final model for this project based on the best R^2 score. As the features of this problem don't reveal any real life characteristics, instead it is presented in an encoded way, it's not possible just intuitively tell if the insurance claims vary linearly or nonlinearly.

Benchmark: In the dataset, insurance claim or 'loss' is provided for each claim filed. So "Insurance Claim" in dollar will be the output of the model. Closer the model output to actual output, better it is. That means higher the R^2 value is, better the model is. The goal of the project is to have a model that can predict 50% variance of the insurance claim value from the mean value of all the data. That means R^2 value should be at least 0.5. Also the purpose of the project is to predict the claims relatively so that they can be grouped properly. If all claims predictions are shifted in the same direction, it can be tolerated.

Data Preprocessing: Claim values more than $30,000 are removed as they are very few in number. Now there are total 188255 data points, that means 63 data points are removed. Feature selections are done using Linear regression by varying numbers to see if reducing any feature gives significant improvement to model performance or if it gives, how many important features improve the performance of the model. Before doing feature selection, label encoding are done using sklearn so that data can be used in the model.

OneHotEncoder would be preferable than LabelEncoder. But there are total 116 categorical features and each of them has at least several values. So if they are converted using one hot encoder, total number of features will blow up and the run time for this large dataset will be very large. So, total number of features with one hot encoder will be around 1000 compared to 130 with label encoder. That's why one hot encoder was not used.

Separate model will be created after removing the entries from the dataset with very high "loss" value, as there are not many in the dataset like these and they can be considered noise, but second model is created including all, as in reality they may appear.

Implementation: All the categorical features are converted to digits from letters one by one. Because here are mixes of both categorical and continuous valued features in the dataset, each feature was taken care individually and then finally merged using python script.

For feature selection, “selectkbest” filter was used from sklearn and Linear model regressor was used in the sklearn pipeline function to select best “K” features. The value of “K” was found by running it for few times and modelling the train data. But to select “K” features, 1% of the total data was used. It was tried using 10% of the data for feature selection, but it didn’t give any significant improvement on performance. So only 1% of the data were used.

Remaining features other than best “K” features are excluded from remaining 99% of the data using the help of “get_support()” function from pipeline. If a feature is selected, it has “True” entry from the function return and “False” entry from the function return if not selected. Using these values, remaining data were cleaned up using python code.

The remaining data are divided into training and test set. 80% of data will be used for modeling using regressor algorithm and remaining 20% of the data are used for testing the performance of the model created.

For selecting the value of ‘K’, it was tried to use 50,60,70,80,100,110. But after 100 it didn’t improve. So K=100 was chosen.

Linear Regression with best 100 features was attempted. It was giving r^2 value of 0.49.

Decision tree regressor was attempted next using cross validation and shuffle-split. The depth of tree was chosen from 8,9,10,11,12,13,14 and 15. The best tree was giving r^2 value of 0.449.

K-nearest neighbours regressor was attempted next. The value of nearest neighbour was chosen from 5,10,15,20,25 and 30. All the models were produced and only the best model was picked based on the r^2 value from the test set. Finally the r^2 value achieved using this algorithm was 0.35.

Random Forest algorithm with 10 trees in the forest was attempted next. The r^2 value was found to be 0.47. The maximum number of levels were not selected for the trees in the forest. So it could overfit. Also the number of trees were not tried to vary. This has some room for improvement.

Refinement: Because random forest was very close to be the best solution in the initial stage and there were some known parameters to iterate for trying to improve the solution, it was picked for trying to improve the model.

The initial value was 0.47. The goal should be to improve as much as possible. Also it has be in mind that this is a very large data set with lot of features. So each run takes long time to get the results.

In the process of iterations, the number of features that gave good result was found to be 125. So instead of 100 features used in the initial stage, 125 best features were chosen out of 130 features.

Next improvement was using polynomial features. Degree =2 was used instead of default Degree=1 in the function “PolynomialFeatures” in sklearn. It creates all possible second order features from first order features. So number of features increase to a very high number compared to initial 130 features. In this case it took 125 best features and converted them to all possible second order features. The intuition to attempt polynomial features is the nature of the problem. In case of insurance claim, interaction between features highly likely, for example, the door of a luxurious car damager are more expensive than door of a basic car. So if there are features like brand of car and part of body damaged, they will give better

results if combined. Also some features may vary polynomially instead of linearly. For example, the value of a car is reduced very significantly in first 2 years compared to after 5-6 years. So it is worth attempting polynomial regression. After converting 125 features to second order polynomial features, total 8000 features were created.

In the random forest algorithm, maximum number of features to be used, can be given. Started from 100 features, number of features were increased by 25 each time to get the improvement of increasing number of features. Once it was 325, it seemed nearly saturated, then 335 features were chose.So 335 was found to be the best number for improving the model within reasonable run time from the 8000 features after second order polynomial conversion.

The number of trees in the forest was increased to 110 from 10 in the initial model. Although it increases the run time significantly, but to improve the model it was necessary.

The maximum depth of the tree was set to be 22. This reduces the r^2 value on training set but improve the value on test set. There was an attempt with lower value of depth, for example, 16, it reduces the r^2 value on training set but also reduces on test set slightly. So maximum depth of 22 for tree was chosen.

The final r^2 value was found to be 0.55 on test set. Initially maximum achievable r^2 value was 0.49. Both the results are on 20% of the dataset. So the testing is robust enough.

Model Evaluation and Validation: The model was evaluated on 20% of the data and r^2 value was found to be 0.55. That means 55% of variance was explained

by the model on the 20% unseen data. Because Random Forest algorithm gave best r^2 value among all, it was chosen as the final model.

Because the test set is very large, some entries from the test set are presented in the following table.

Entry number	Actual Claim $	Model Output $	Difference in $
10	2977.26	3922.69	-945
15	1157.76	1475.28	-318
20	980.81	1795.84	-815
25	601.81	1526.92	-925
30	1486.42	1881.32	-395
35	2183.46	2891.43	-708
40	708.57	1415.50	-707
45	**4855.51**	**3055.83**	**1800**
50	1018.39	1879.89	-861
55	2679.03	3407.81	-728
60	2202.42	3220.29	-1018
65	1574.62	1368.75	206
70	**1833.75**	**4154.05**	**-2321**
75	812.17	1504.11	-692
80	2271.21	2593.79	-322
85	1966.07	1662.27	304
90	1380.15	2465.42	-1085

95	**3269.49**	**4550.65**	**-1281**
100	1157.45	1769.36	-612

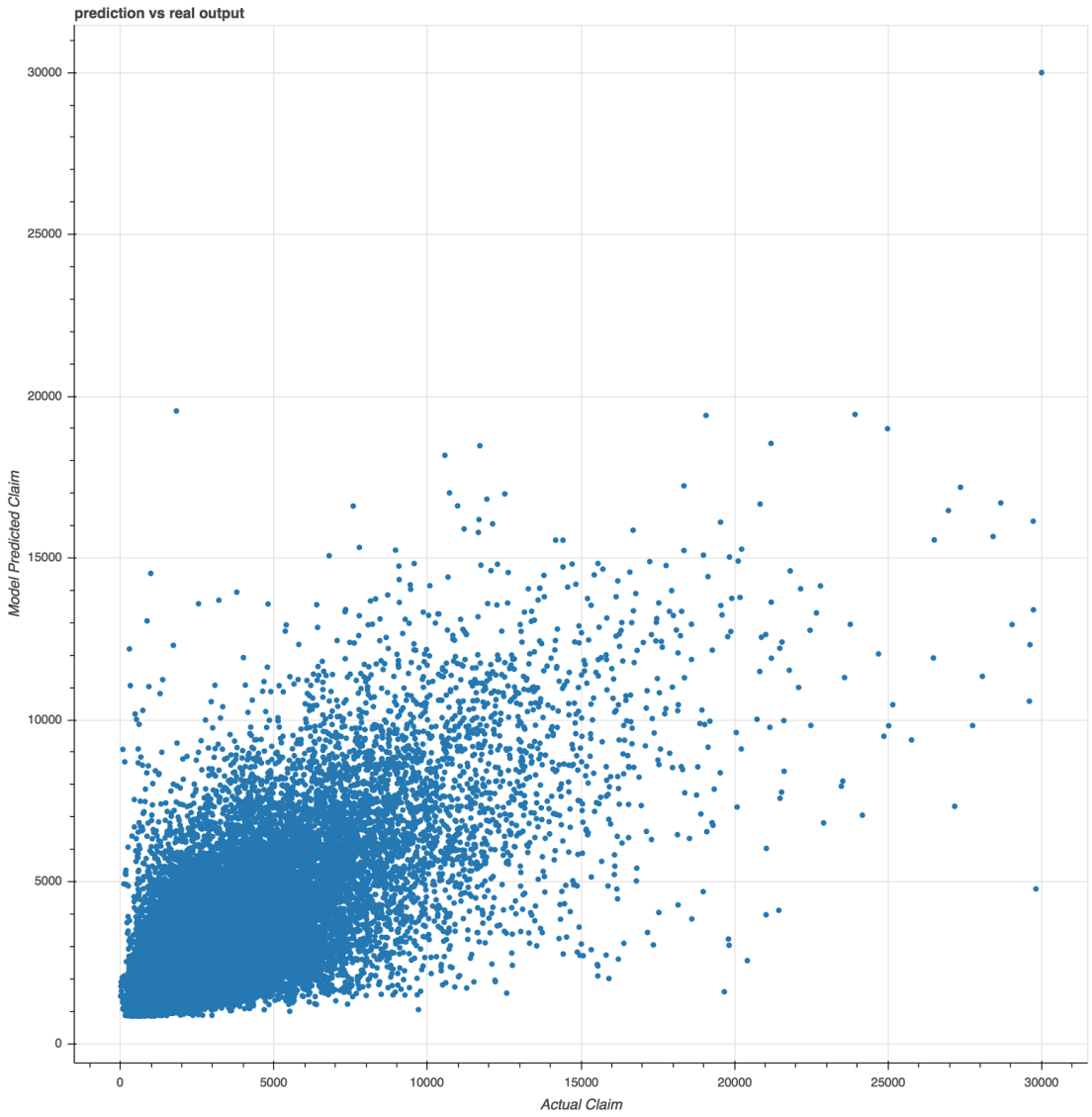
Table: Actual and model output comparisons from test data

Out of 19 examples above, 3 of them are totally off. Those are 45,70 and 95. Rest of them are mostly over valued by the model in the range of few hundred to $1000. Few of them are valued less but they are very close. It is not possible to predict exactly the claim values but closer it is, better the model is. But in the majority of the cases the model is missing the actual number by -$500 to -$1000. Because the goal is to predict the loss approximately and treating the cases based on their claim prediction, missing the actual value by a consistent amount, works very well, as relatively they are well predicted. Also there can be cases who are very hard to predict or doesn't show much characteristics and can be considered noise.

The robustness of the model was tested by using 70% of the data as training set and remaining 30% as the test data. The r^2 score was almost same. So small perturbations in the training data didn't change the model.

Justification: The purpose of the project was to predict the loss values relatively so that based on prediction the customers can be taken care properly. If absolute value is different, it could be found after evaluation practically. In that sense, the model is justified because the r^2 value for test set is more than 0.5 and the predictions are off by few hundred to $1000 in most cases. That means if the losses are $500 and $1000, they can be evaluated as $1000 and $1500 respectively without harming the purpose of creating the model.

Free-Form Visualization: The actual and model output values are plotted in the figure below.



From the figure it is seen that there are fewer examples for higher claim value and they are spreaded more from the diagonal (45 degree line with X or Y axis). If it is close to diagonal, the value is predicted more accurately. But if it is far from the diagonal, it is predicted wrongly. It can be observed from the above plot that the points are mode spreaded and all over the places when the claim value is very high. As there are not many examples, model couldn't do a great job for predicting them and they happened to be really wrong.

Reflection: This project was initially started with using some algorithms using their basic settings. From all the algorithms tried, the better one(random forest) was

chosen to further improve the model. There were many iterations to get a reasonable solution to this problem. The dataset is very large and have many features as expected in case of an insurance claims. That's why it was challenging to find a model to predict the claims reasonably well. The iterations were not automated as it runs take very long time and after many iterations and time passed, the attempt could be proven wrong. That's why the parameters were changed carefully each time and changed again to look for improvements.

There was a challenge with the features that they were encoded. So it was not possible to predict which feature means what and if they could seem to be important based on the common sense other than using the algorithms.

To predict the severity of the claims from the features given, the model can be used satisfactorily in most frequently occurred cases.

Improvement: The model seems to show improvement with non-linear features (order =2). The third order was also attempted but because of dataset being very large and number of features being very large, the computational resource available were not able to proceed within reasonable amount of time. So it would be worth trying to improve the model further. Also non-linear regressor like "Support Vector Regressor" could be tried to improve the model that was not possible because of run time being very long.

Because of run time for modeling the data being very large, the model was not created by iterative runs using the benchmark. Instead it was manually improved by sequential runs. So changing the benchmark has no impact on the model used.