Universidad de Castilla-La Mancha

Escuela Superior de Informática

Supervised Learning

GitHub

Authors:

Josue Carlos Zenteno Yave
Sergio Silvestre Pavón
Javier Álvarez Páramo
Sergio Martín-Delgado Gutiérrez

1 Introduction

At this document we are going to explain our solution to the given challenge which states the following rules:

Given a data set of 30.000 policy insurances claims we have to predict Workers Compensation using that realistic data.

The data set contains the following variables:

- **ClaimNumber**: Unique policy identifier

- **DateTimeOfAccident**: Date and time of accident

- **DateReported**: Date that accident was reported

- **Age**: Age of worker

- **Gender**: Gender of worker

- **MaritalStatus**: Martial status of worker. (M)arried, (S)ingle, (U)unknown.

- **DependentChildren**: The number of dependent children

- **DependentsOther**: The number of dependants excluding children

- **WeeklyWages**: Total weekly wage

- **PartTimeFullTime**: Binary (P) or (F)

- **HoursWorkedPerWeek**: Total hours worked per week

- **DaysWorkedPerWeek**: Number of days worked per week

- **ClaimDescription**: Free text description of the claim

- **InitialIncurredClaimCost**: Initial estimate by the insurer of the claim cost

- **UltimateIncurredClaimCost**: Total claims payments by the insurance company. * This is the field you are asked to predict in the test set.

2 Preprocessing

At this section all the considerations that have been taken to normalize and clean the data of the data-set are going to be explained. This explanations are also a part of the own colab file so, if you want to read them there, you can do it easily.

2.1 Missing Values

We have removed every entry of the pandas dataframe that has not all the values. By doing this we have reduced the size of the data set.

2.2 One Hot Encoding

The strategy implemented by One Hot Encoding is to create a column for each different value that exists in the feature we are encoding and, for each record, mark the column to which that record belongs with a 1 and leave the others with 0.

3 Baseline

At this section we are going to explain all the considerations that we have taken into account to build the very first try of a model by modifying the data set, reducing the features, splitting the data in two small data sets on for training and the other for testing the model.

This explanations are also a part of the own colab file so, if you want to read them there, you can do it easily. Colab file.

3.1 Feature Selection

For this part we have decided to remove the following features:

- *Gender*, *MaritalStatus*, *PartTimeFullTime* as they were redundant.

- *ClaimNumber*, *DateTimeOfAccident*, *DateReported*, *ClaimDescription* as they were non numeric values.

An algorithm has been developed and you can find it in the corresponding section into the Colab file.

3.2 Train and Test split

For this section we have splitted the data set into two smaller ones, the first, that includes the 80% of the data, which is destinated to train the model and the other 20 % will be saved in tghe second data set and this one will be destinated to test the model.

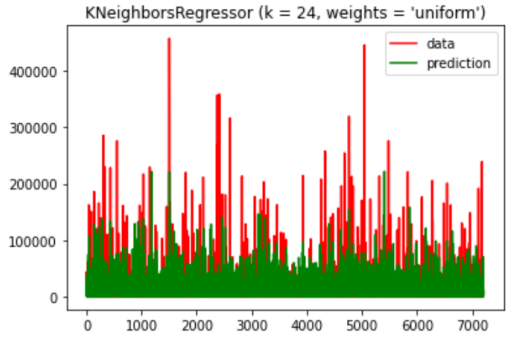
An algorithm has been developed and you can find it in the corresponding section into the Colab file.

3.3 KNN

At this part of the colaboratory file we are ready to start training the model, for this purpose we have decided to use KNN by using cross validation and a euclidean metric.

In statistics, the k-nearest neighbors algorithm (KNN) is a non-parametric classification. It is used for classification and regression. In both cases, the input consists of the k closest training examples in a data set. The output depends on whether KNN is used for classification or regression but, in our case we use the KNN regression as the output is the property value for the object. This value is the average of the values of k nearest neighbors.

The results that this model obtains are not the best ones as it can be seen in the following pictures:



And the obtained errors are the following ones:

```
MAE: 7404.008356131998
MAPE: 0.8814212795553249
MSE: 463051675.183564
R^2: -0.5772671784943437
```

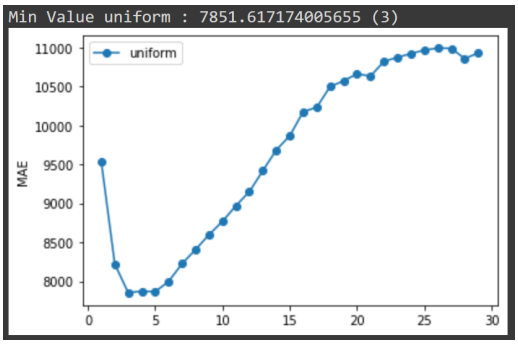
An algorithm has been developed and you can find it in the corresponding section into the Colab file.

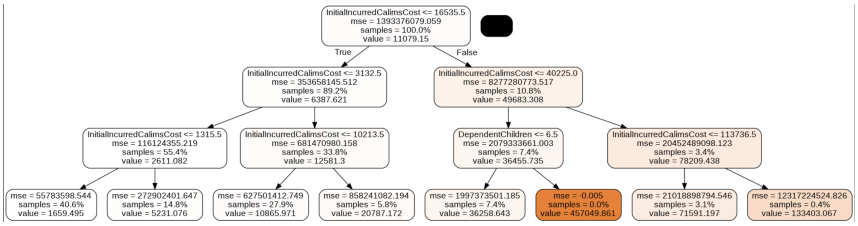
3.4 Decision Trees

Once we have tried using KNN, now it is time to try again but using decision trees.

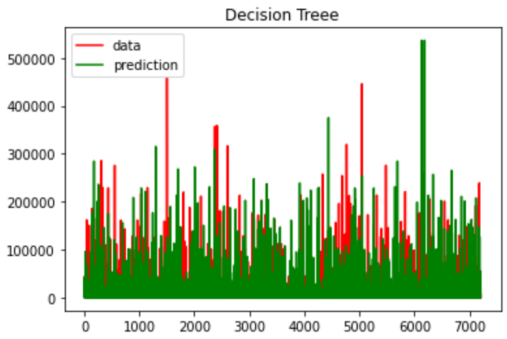
The goal of the algorithm is to predict a target variable from a set of input variables and their attributes. The approach builds a tree structure through a series of binary splits (yes/no) from the root node via branches passing several decision nodes (internal nodes), until we come to leaf nodes. There is no single decision tree algorithm. Instead, multiple algorithms have been proposed to build decision trees.

In our case we have set a maximum depth of 3 because we have calculated the Mean Absolute Error and it was the value that has been declared as the best one as it has the lower error as you can see in the following picture:





And results are:



And the obtained errors are the following ones:

```
MAE: 9769.402037785574
MAPE: 1.1930000900796487
MSE: 1004462852.1608009
R^2: -0.22763511486098964
```

An algorithm has been developed and you can find it in the corresponding section into the Colab file.

4 Optimized Model

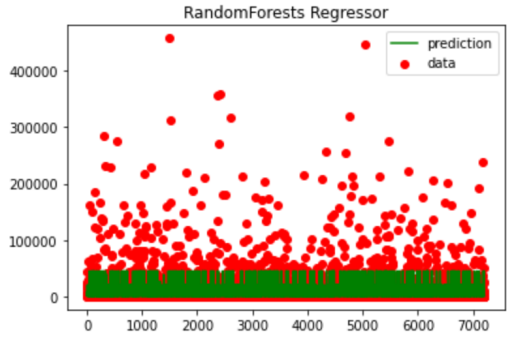
At this section we are going to explain all the considerations taken into account to try to improve the result obtained in the very first try of building a model. Now we are going to try to use "better" models that are supposed to obtain better results.

This explanations are also a part of the own colab file so, if you want to read them there, you can do it easily. Colab file.

4.1 Random Forest

At first we are going to try by using Random Forest which is an ensemble learning method for classification, regression and other tasks that operates by constructing a multitude of decision trees at training time. For our case we are going to use it in the regression way where the mean or average prediction of the individual trees is returned.

And results are:



And the obtained errors are the following ones:

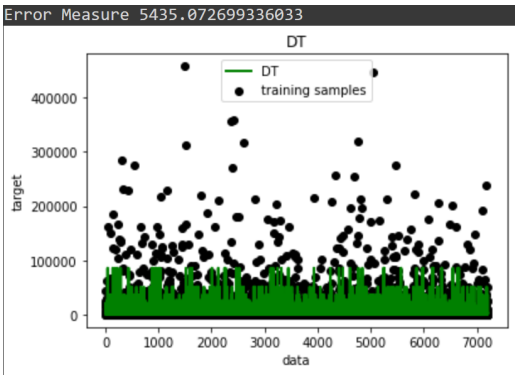
```
MAE: 5915.318501652456
MAPE: 1.3803510745618623
MSE: 503701719.37342286
R^2: -4.315977401509709
```

An algorithm has been developed and you can find it in the corresponding section into the Colab file.

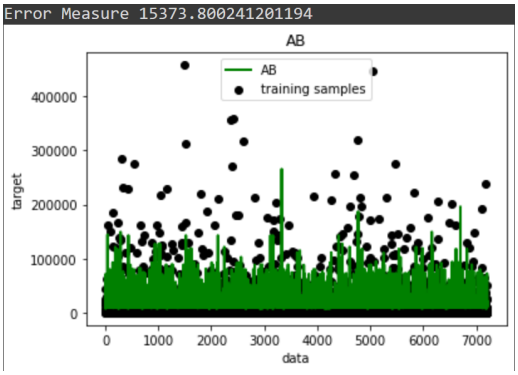
4.2 Boosting

Once we have tried using Random Forest now we are going to try again but in this case we are going to fit the model by using three regressors: decision tree regressor, adaptive boosting regressor (adaboost) and gradient boosting regressor.

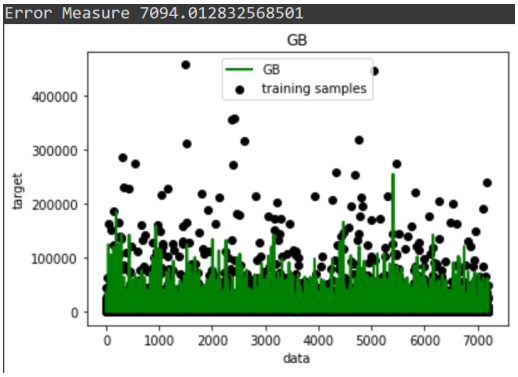
- Decision tree regressor could be useful to begin the building of the model and then apply Adaboost on it and later gradient boosting.



- Adaptive Boosting that consists of creating a sequence of different simple predictors in a way that the second is an improved version of the first one and then the third one is an improvement of the second one and so on so on.



- Gradient boosting is a machine learning technique used in regression and classification tasks, among others. It gives a prediction model in the form of an ensemble of weak prediction models, which are typically decision trees. When a decision tree is the weak learner, the resulting algorithm is called gradient-boosted trees; it usually outperforms random forest. A gradient-boosted trees model is built in a stage-wise fashion as in other boosting methods, but it generalizes the other methods by allowing optimization of an arbitrary differentiable loss function



An algorithm has been developed and you can find it in the corresponding section into the Colab file.

4.3 Hyperparameter Optimization Random Forest

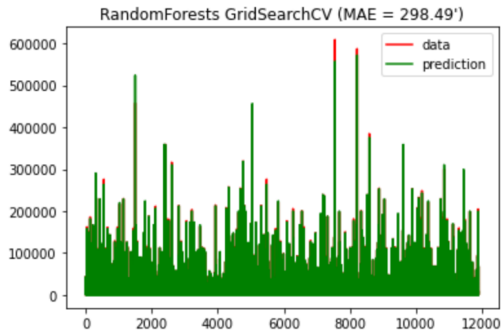
At this section we are going to make use of Hyperparametrization.

First we declare the parameters for the regressor. We are going to use four different estimators or trees, in a random forest context (8,16,24,32).Also we have established three different maximum depths or levels in tree context (6,5,4).We have set cross validation to 2 because by default its value was 5 and this has been done with Grid search that is a parameter fitting approach that allows you to methodically construct and evaluate a model for each combination of specified algorithm parameters in a grid. And finally we have trained every combination to see which one was the best one. The best one was the one with less error.

The selected one was:

```
RandomForestRegressor(max_depth=6, n_estimators=8, random_state=0)
```

Finally we have used the previously trained model to predict the testing data set obtaining the following results:



With the following errors:

```
MAE: 298.4868923240643
MAPE: 0.2039058437434235
MSE: 868204.0391195125
R^2: 0.9988003549718651
```

An algorithm has been developed and you can find it in the corresponding section into the Colab file.