PREDICTING EARTHQUAKE DAMAGE IN NEPAL

Miguel Ángel Barberán, April 2018

SUMMARY

This report introduce an analysis of data regarding the damage produced by the earthquake in Ghorka District of Nepal happened in 2015. This analysis has been performed by taking in consideration a dataset of 10.000 rows and 38 features.

At first, some basic data exploration steps took place. After the computation of diverse statistics and visualizations to dive deeper into the data understanding, several potential relationships between the building characteristics and their damage grade based on these characteristics were noticed. Once the exploration stage was finished, the next step was then to determine the potential value of each feature aiming to solve a classification problem.

To achieve this purpose, the goal was to identify the potential statistical weight of each feature in relation with the "damage grade" target variable. The approximation here was made by looking closely at some statistics that help us with this endeavor like the correlation coefficients of the features.

During the process, a number of features were pruned from the dataset, and another number was transformed into more valuables and useful features for the purposes of a correct modeling.

As a final preparation step, the target variable was oversampled in order to balance the number of samples of each target category to classify, and avoid bias and skewness in the modeling process.

Finally, three classification models were created, tuned, and compared in terms of performance and accuracy to determine which one was helping best to solve our classification problem.

At the end of the present analysis, the following conclusions were reached:

1. Among the initial given dataset, containing 38 features, some of these features were contributing very poorly or not at all to the modeling process.
2. A high level of noisy data was identified
3. Important majorities of these features were contributing the most, or reached their maximal statistical importance when combined together in distinct ways.
4. A correct classification of the classes in the target variable was not possible without some resampling techniques and procedures.

The most significant features noticed were:

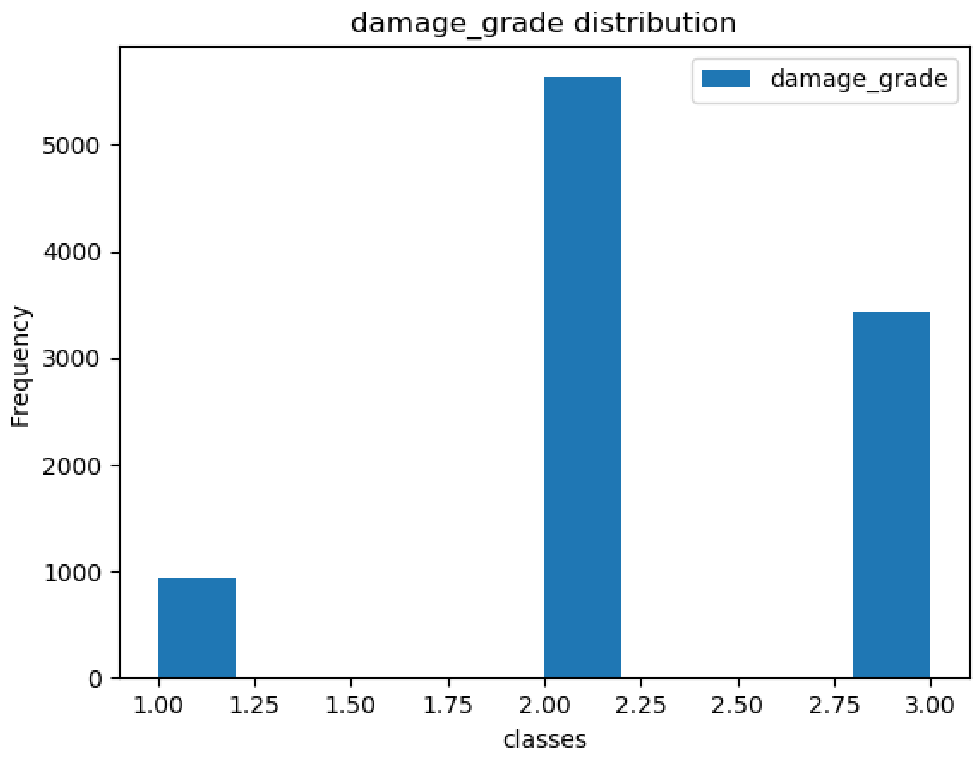
- **Geo_level_1_id:** Feature containing the information about the geographic region in which the buildings exist. From largest (level 1), to most specific sub-regions (level 3).
- **Count_floors_pre_eq:** A feature containing the count of floors number in each building before the earthquake.
- **Age:** the age of the buildings in years.
- **Area:** the plinth area of the buildings in squared meters.
- **Height:** height of the buildings in meters.
- **Superstructure set:** A set of binary features holding information concerning the materials used to build the superstructure of the buildings.
- **Count_families:** number of families that live in the buildings
- **Secondary_use set:** a set of features with information indicating if the buildings were used for any secondary purposes.

COLUMN	**MEAN**	**STD**	**MEDIAN**	**MIN**	**MAX**
geo_level_1_id	7.13	6.22	6.0	0	30
geo_level_2_id	296.93	279.39	219.0	0	1411
geo_level_3_id	2678.61	2520.66	1937.5	0	12151
count_floors_pre_eq	2.14	0.73	2.0	1	9
age	25.39	64.48	15.0	0	995
area	38.43	21.26	34.0	6	425
height	4.65	1.79	5.0	1	30
has_superstructure_adobe_mud	0.08	0.28	0.0	0	1

has_superstructure_mud_mortar_stone	0.76	0.42	1.0	0	1
has_superstructure_stone_flag	0.02	0.17	0	0	1
has_superstructure_cement_mortar_stone	0.01	0.13	0	0	1
has_superstructure_mud_mortar_brick	0.06	0.25	0	0	1
has_superstructure_cement_mortar_brick	0.07	0.25	0	0	1
has_superstructure_timber	0.25	0.43	0	0	1
has_superstructure_bamboo	0.28	0.28	0	0	1
has_superstructure_rc_non_engineered	0.19	0.19	0	0	1
has_superstructure_rc_engineered	0.01	0.11	0	0	1
has_superstructure_other	0.01	0.11	0	0	1
count_families	0.98	0.42	1	0	7
has_secondary_use	0.10	0.31	0	0	1
has_secondary_use_agriculture	0.06	0.25	0	0	1
has_secondary_use_hotel	0.02	0.16	0	0	1
has_secondary_use_rental	0.006	0.07	0	0	1
has_secondary_use_institution	0	0.02	0	0	1
has_secondary_use_school	0	0.02	0	0	1
has_secondary_use_industry	0	0.02	0	0	1
has_secondary_use_health_post	0	0.01	0	0	1
has_secondary_use_gov_office	0	0.01	0	0	1
has_secondary_use_use_police	0	0.01	0	0	1
has_secondary_use_other	0	0.07	0	0	1
damage_grade	2.24	0.61	2	1	3

By looking into these statistics, it can be quickly noticed that the behavior of the features that our model is going to be based on is not as good as we would like. The first oddity that takes our attention is the fact that there is an important quantity of binary encoded features, more than the half of the dataset, and the first thinking could be that these binary features in such a great number could introduce quite a good amount of noise, and this could affect the performance of our model. By this moment, it is known that something has to be done with these conflictive features.

Another insight that can be extracted from just looking at this table, is that the target variable's mean is 2.24, knowing that its max is 3, and its min is 1, it can be guessed with more or less precision that that the 3 classes presents in this variable are skewed. This is confirmed by the histogram of the target variable, which we are going to look at. But we will look into that in the next epigraphs.



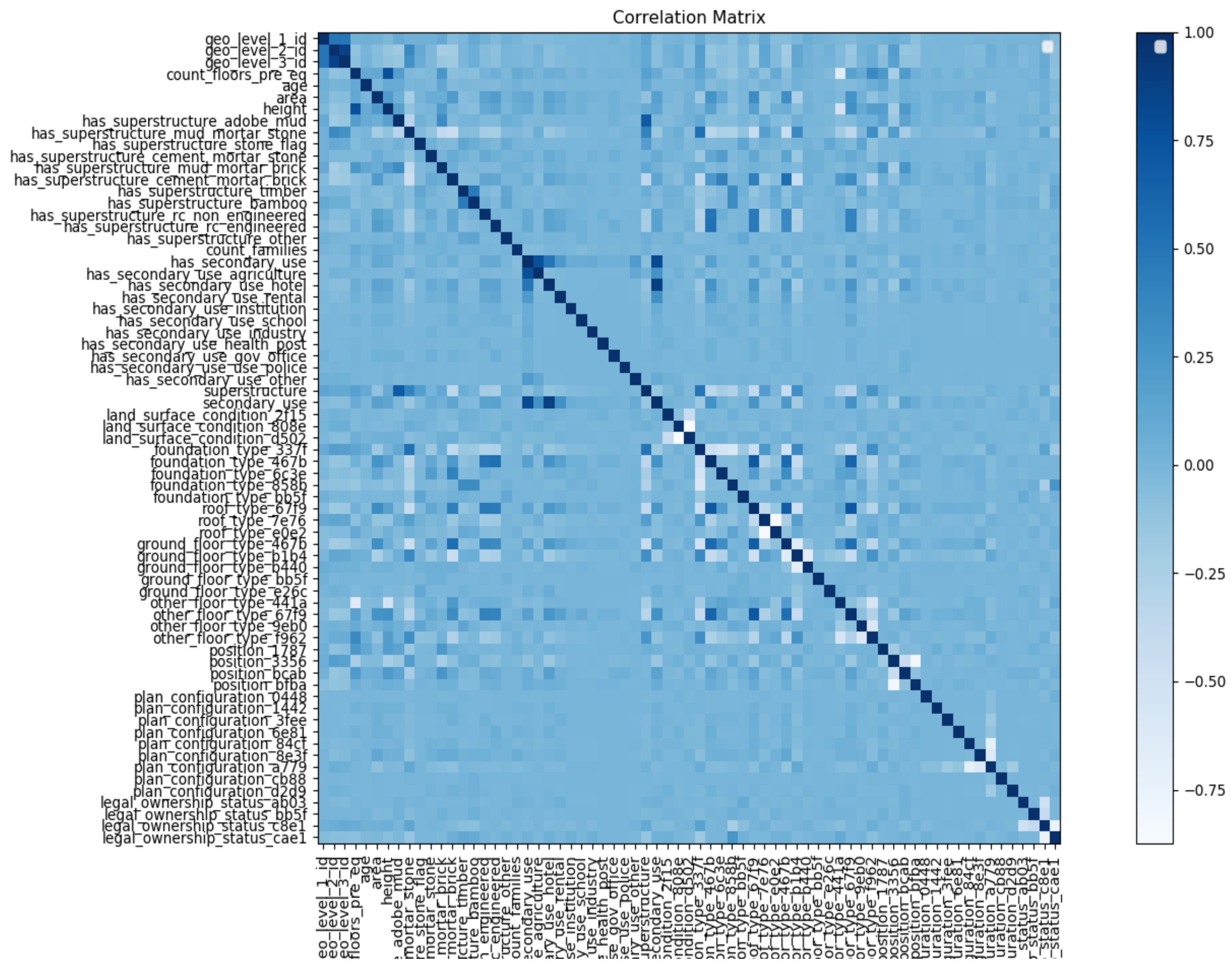
Apart of these numerical features, there are also eight categorical "string" type features, which also need to be dealt with as transformed features in order to make them suitable for our model.

The categorical features are the following:

- **land_surface_condition:** the surface condition of the land where the building was built.

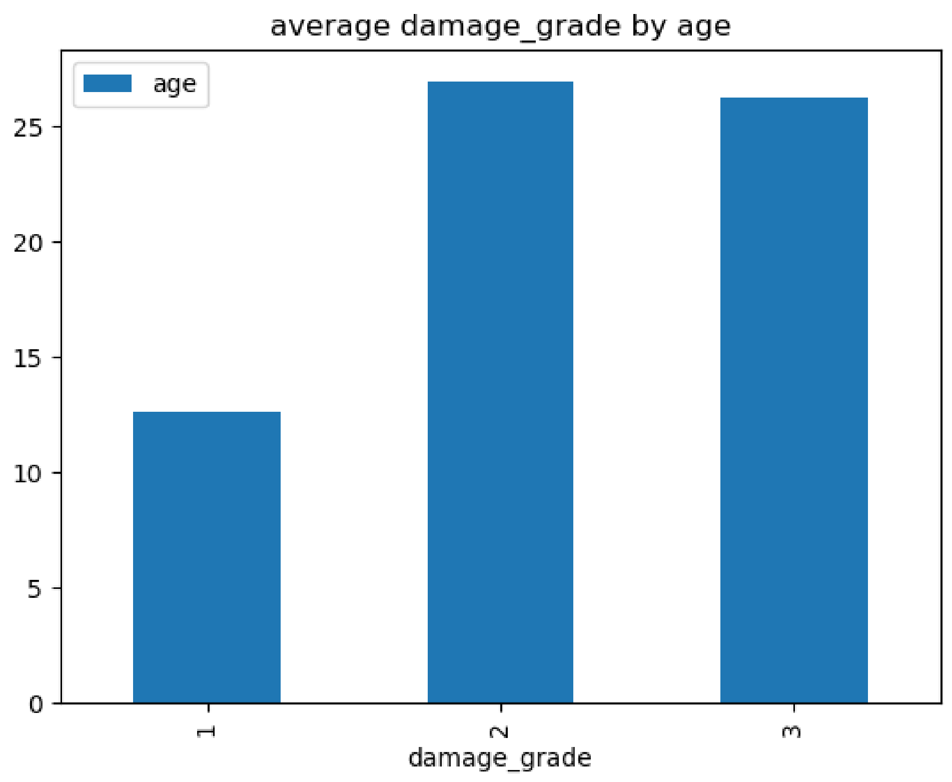
- **Foundation_type:** type of foundation used while building.
- **Roof_type:** type of roof used while building.
- **Ground_floor_type:** type of the ground floor.
- **Other_floor_type**: type of constructions used in higher than the ground floors (except of roof).
- **Position:** position of the building.
- **Plan_configuration:** building plan configuration.
- **Legal_ownership_status:** legal ownership status of the land the building was built.

As a way to determine in which measure these features could be helpful during the modeling process, a transformation was made, by binary encoding these features they were properly set up for the model's algorithms to be used. Again, the correlation coefficients of the features were examined with attention:

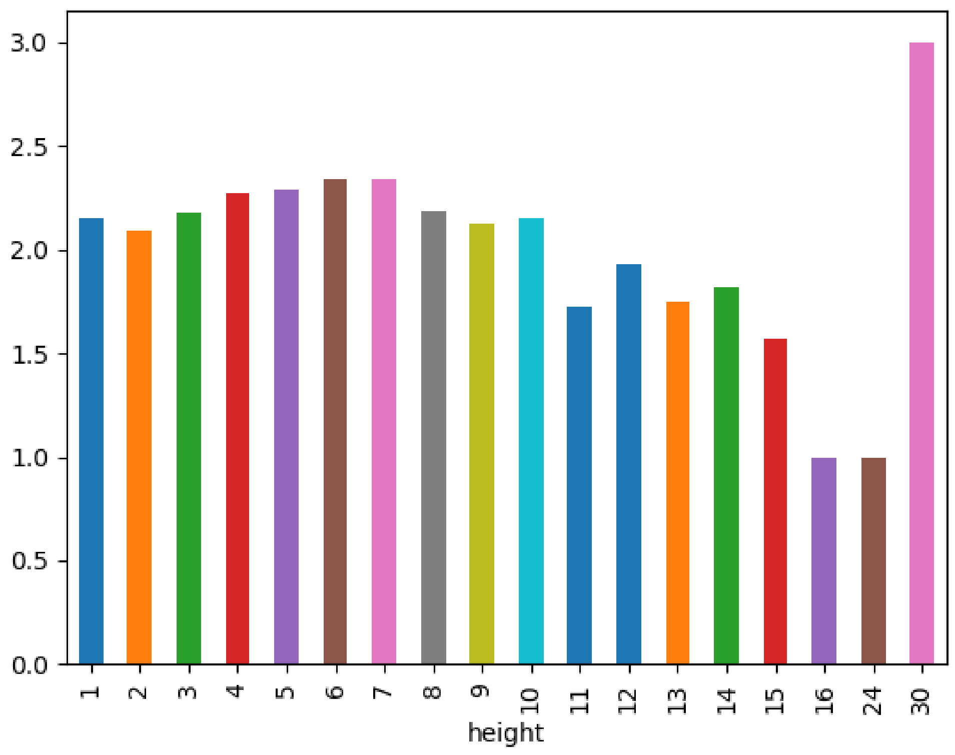


As shown in the figure, It is noticeable that a good quantity of noisy data exists. In order to efficiently model the relationships between the given features and the target variable, this noisy data must be dealt with somehow. The first step here was to categorize all the text “string” features and transform them into numeric values, so is it possible to better observe the correlation coefficients with the target variable, and start pruning those features which are most likely to add noise to our model.

Another interesting discovery was that as observed in the next figure, the damage grade was more likely to be higher in buildings which were older in terms of age:



However, the damage behavior is not the same when it is explored regarding the building heights:



By observing the plot it can be stated that building heights are not strongly correlated to building's damage grade caused by earthquakes. In fact, it is clear that the damage is almost normally distributed amongst the buildings, perhaps a bit skewed, with the exception of some of the highest buildings, that suffered the highest damage level, and also the lowest damage.

Correlations and relationships.

After the data exploration stage was finished, a deeper exploration of the correlations and relationships in the data was conducted.

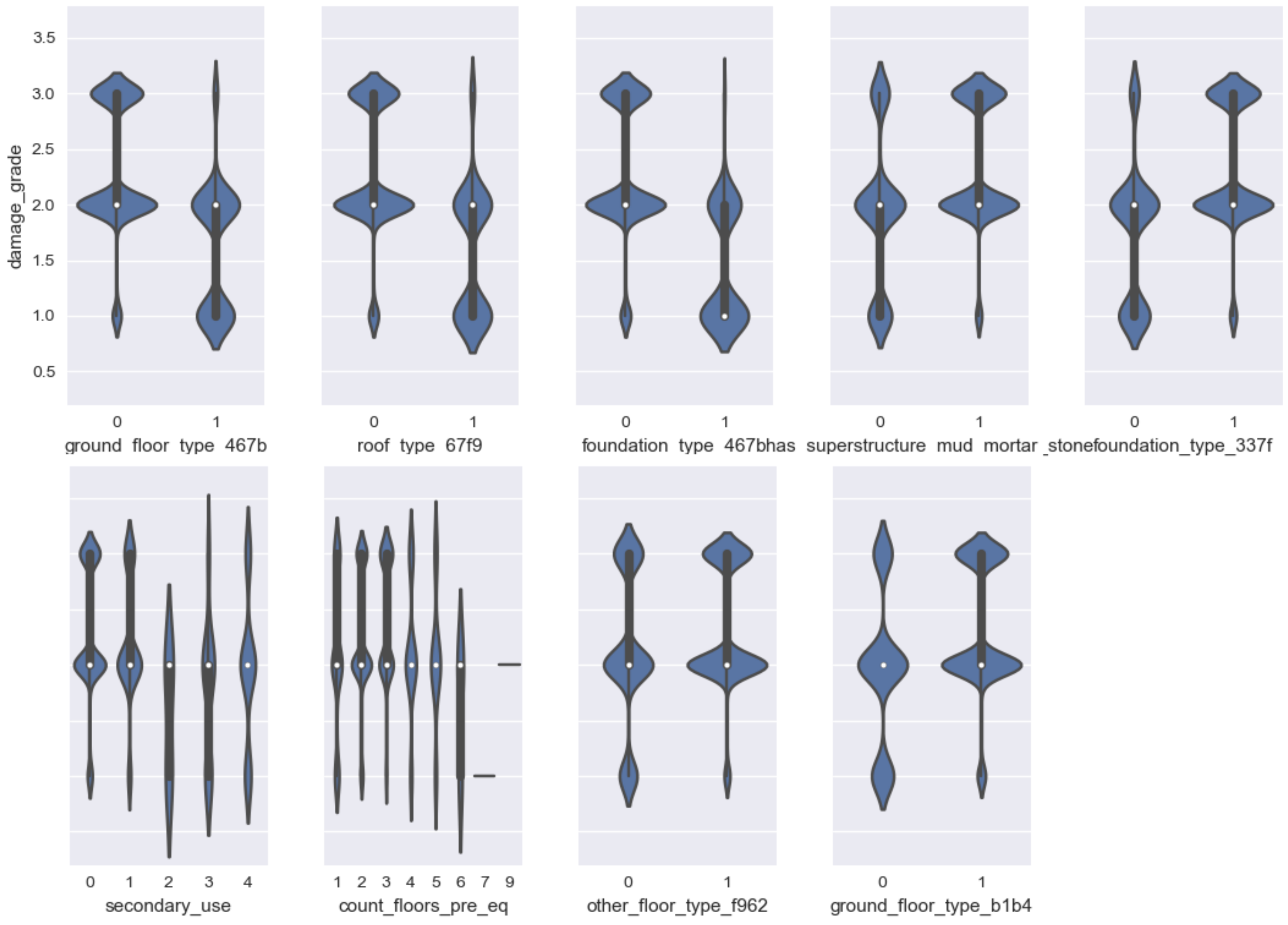
In addition to the correlation matrix shown above, it was also useful to investigate the Pearson's correlation table of the target variable to the other features. This way it was possible to prune and select those features which were more important and had a superior weight to predict the target variable.

This are the chosen features at this stage based on their correlation coefficients with the target variable “damage_grade”:

Features	**damage_grade**
ground_floor_type_467b	-0.314422628
roof_type_67f9	-0.272906081
foundation_type_467b	-0.262965728
has_superstructure_cement_mortar_brick	-0.233397655
other_floor_type_67f9	-0.220604189
has_superstructure_rc_engineered	-0.179761203
foundation_type_858b	-0.171043844
has_superstructure_rc_non_engineered	-0.156371249
foundation_type_6c3e	-0.133998557
area	-0.114775002
other_floor_type_441a	-0.1141705
secondary_use	-0.108705097
has_secondary_use	-0.087286354
has_secondary_use_hotel	-0.085268699
has_superstructure_timber	-0.08055793
has_superstructure_cement_mortar_stone	-0.073337428
has_superstructure_bamboo	-0.072327307
legal_ownership_status_cae1	-0.066896584
geo_level_1_id	-0.064084324
has_secondary_use_rental	-0.063367062
plan_configuration_8e3f	-0.051512445
position_bcab	-0.044988466
has_secondary_use_school	-0.041652487
has_secondary_use_other	-0.031927369
foundation_type_bb5f	-0.030599472
has_secondary_use_institution	-0.029295632

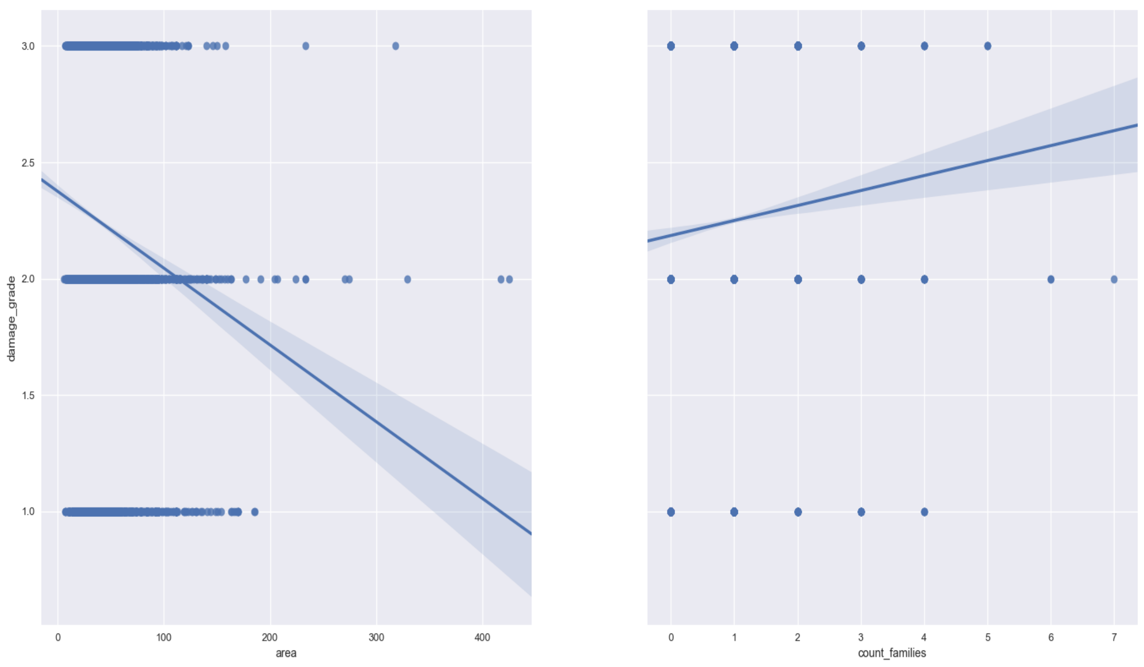
plan_configuration_3fee	-0.028273553
plan_configuration_6e81	-0.023307911
has_superstructure_other	-0.019515346
has_secondary_use_gov_office	-0.017306099
has_secondary_use_agriculture	-0.013333012
has_secondary_use_industry	-0.011503873
legal_ownership_status_bb5f	0.010122177
plan_configuration_cb88	0.012275868
other_floor_type_9eb0	0.024389893
legal_ownership_status_ab03	0.029224505
height	0.031728002
legal_ownership_status_c8e1	0.031735925
position_bfba	0.036427599
age	0.038219188
plan_configuration_a779	0.039981969
count_families	0.044518191
has_superstructure_stone_flag	0.050476528
has_superstructure_adobe_mud	0.055370726
roof_type_e0e2	0.066212647
roof_type_7e76	0.077309175
count_floors_pre_eq	0.112295661
other_floor_type_f962	0.157632656
ground_floor_type_b1b4	0.225436368
superstructure	0.272854598
has_superstructure_mud_mortar_stone	0.277533169
foundation_type_337f	0.338973928

Some of the most importantly correlated of the chosen features are represented in the next figure:

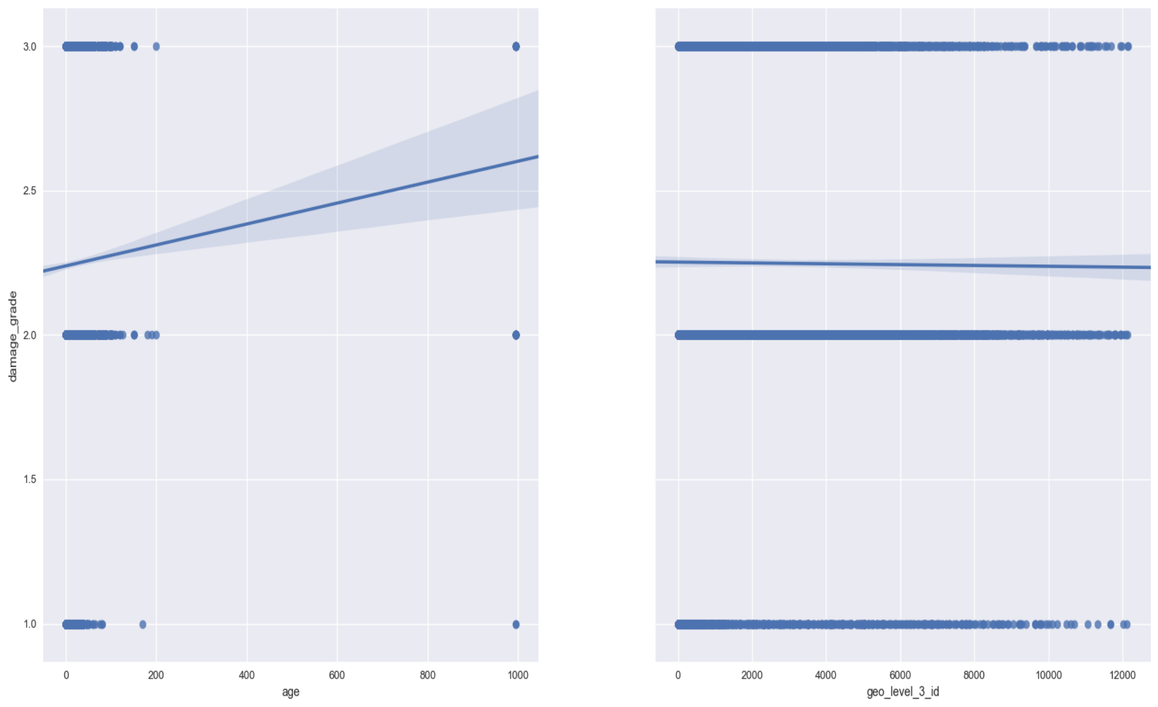


Notice that the "ground_floor_type_457b" feature has a high incidence in the lowest damage levels. It is also noticeable that the rest of features are widely spread amongst all damage levels. It is also interesting to see that the "ground_floor_type_b1b4" feature is related with those buildings that took the highest damage level; however, those buildings that did not have this ground floor type have lower damage levels. One more finding here is that those buildings, which had between one and five floors, were cumulating medium to high damage level, and those with more floors suffered only low damage level.

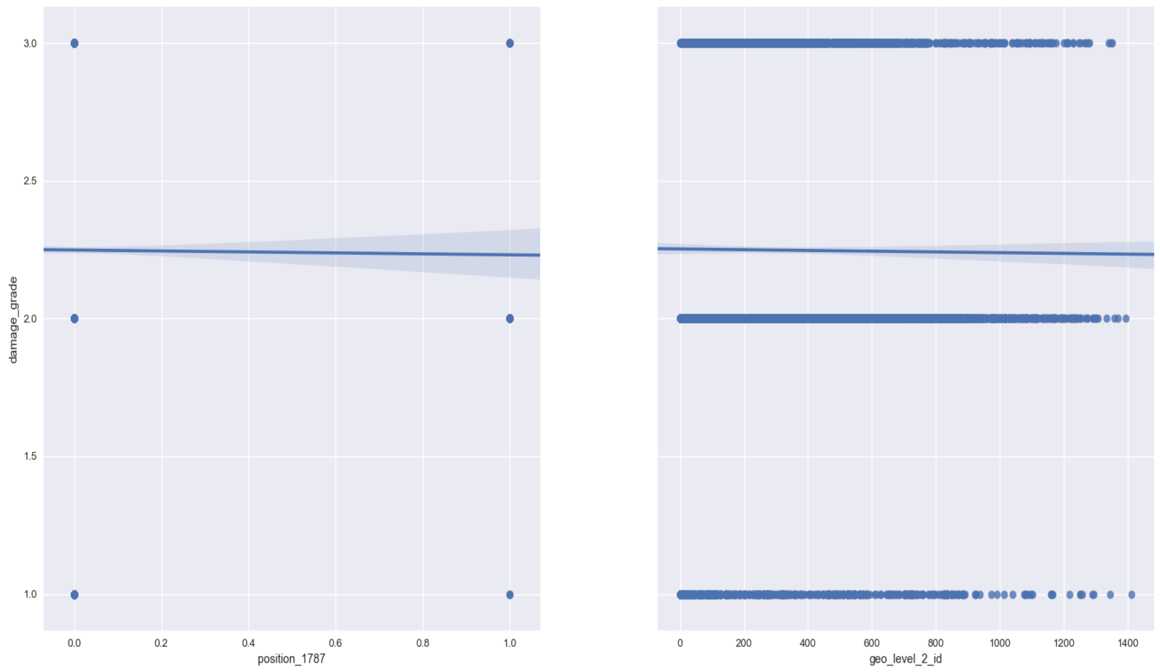
More complex relationships can be identified in the next figure:



This figure shows the relationship between the “area” and “count_families” features, and the target variable. Notice that the relation between the three is inversely proportional. Buildings with smaller areas tends to suffer lower damages but buildings were more quantity of families live in took more damage. This does not seems a very normal thing, since the buildings with more capacity to host people should usually be bigger and taller, which means that they should have a bigger area as well, but in this case, this contradictory relationship was found. Maybe it should be understood that in this city most of the buildings were people live in with their families are small, and the biggest buildings could be commercial buildings, offices, etc…



In this plot, another interesting observation can be made. By looking at it, it can be quickly guessed that older buildings received higher damage levels, but it is realized that the geo level 3 information was not significant when it comes to determine if one region or another is to be more affected by earthquakes damages.



The same point is stated here. A single glance is enough to realize that the building position and geo level 2 are almost irrelevant to damage grade took by the buildings during the earthquake. Perhaps a very slight tendency can be seen, but It does not look important enough to give this features an extraordinary importance.

Data Transformation

Once the data was explored and apparent relationships were identified, a transformation process was run on the data in order to reduce irregularities and undesired behaviors during the modeling process. The first step to accomplish was to analyze which features could be helpful to achieve the classification task, and how to deal with the categorical features In text format.

The first goal was accomplished by examining the correlation coefficients, and by running a method called RFE or Recursive Feature Elimination. A parallel analysis was performed by running PCA and using decision trees and forests to determine features importance.

Next, a conversion process was made to deal with those above mentioned text formatted features, the chosen way to do this was to binary encode the features and later analyze which of them were more correlated to the target variable.

In addition to these steps, some feature engineering operations were elaborated. After analyzing those many features in the “has_superstructure…” as well as in the “has_secondary_use…” sets, it was noticed that these feature values by themselves, did not have a significant weight in the modeling process, in fact, it was noted that their individual weight was very low, but as a set, it had a totally different value: the weight as a set was superior and was more correlated to the target variable. So it was decided to add all the individual features of these two groups, and combine them into two sets containing the numerical values of adding their respective features.

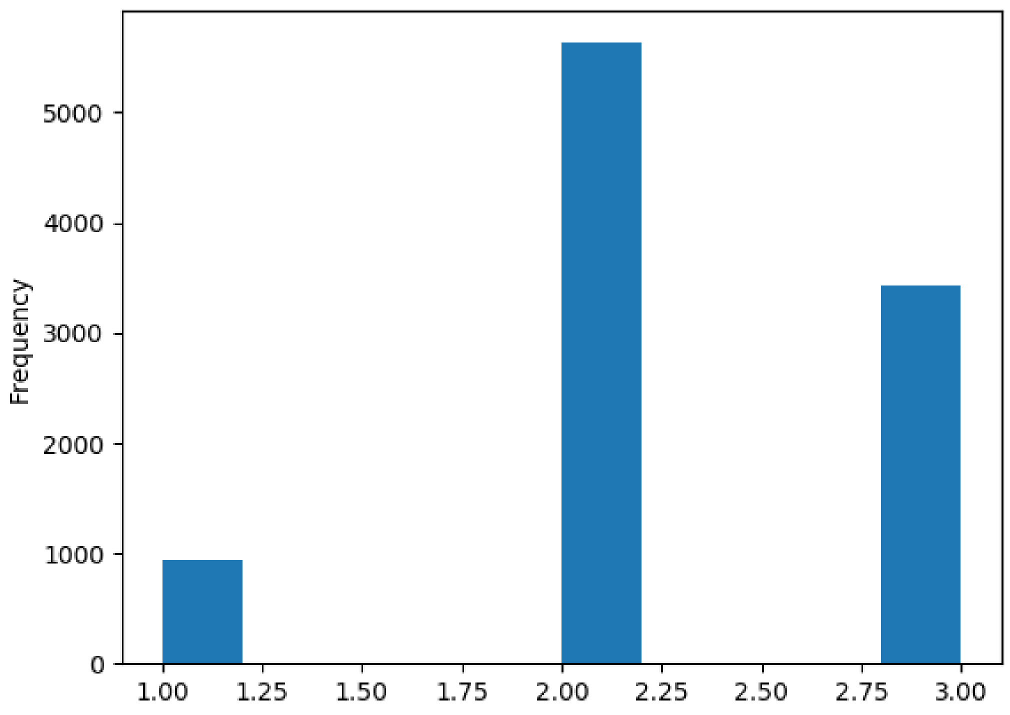
As a result, a boost of the correlation and variance contained in the features was observed.

Finally, these were the chosen features to keep:

1. geo_level_1_id,
2. count_floors_pre_eq,
3. age,
4. area,
5. height,
6. count_families
7. has_secondary_use
8. has_secondary_use_agriculture
9. has_secondary_use_hotel
10. has_secondary_use_rental
11. has_secondary_use_institution
12. has_secondary_use_school
13. has_secondary_use_industry
14. has_secondary_use_gov_office
15. has_secondary_use_other
16. superstructure
17. secondary_use
18. foundation_type_337f
19. foundation_type_467b

20. foundation_type_6c3e
21. foundation_type_858b
22. foundation_type_bb5f
23. roof_type_67f9
24. roof_type_7e76
25. roof_type_e0e2
26. ground_floor_type_467b
27. ground_floor_type_b1b4
28. other_floor_type_441a
29. other_floor_type_67f9
30. other_floor_type_9eb0
31. other_floor_type_f962
32. position_bcab
33. position_bfba
34. plan_configuration_3fee
35. plan_configuration_6e81
36. plan_configuration_8e3f
37. plan_configuration_a779
38. plan_configuration_cb88
39. legal_ownership_status_ab03
40. legal_ownership_status_bb5f
41. legal_ownership_status_c8e1
42. legal_ownership_status_cae1

However, the data preparation was not done yet. An exploration of the target variable's distribution was made, in order to determine in which measure the target variable's samples were more or less adapted for the main machine learning purposes required by this classification problem:



This is the actual distribution of the target variable “damage_grade”. It is clearly noticeable without a doubt that no matter which model or algorithm we decide to create, it will always be skewed, as it will learn much more from the damage level 2 class than from any of the other two classes.

To help overcome this issue, a first model based on the Random Forest classification algorithm was created, in order to have a first measure of what f1 score looked like before starting to make changes in the target variable, and to have the possibility of making comparisons between results, and to perform try-error steps.

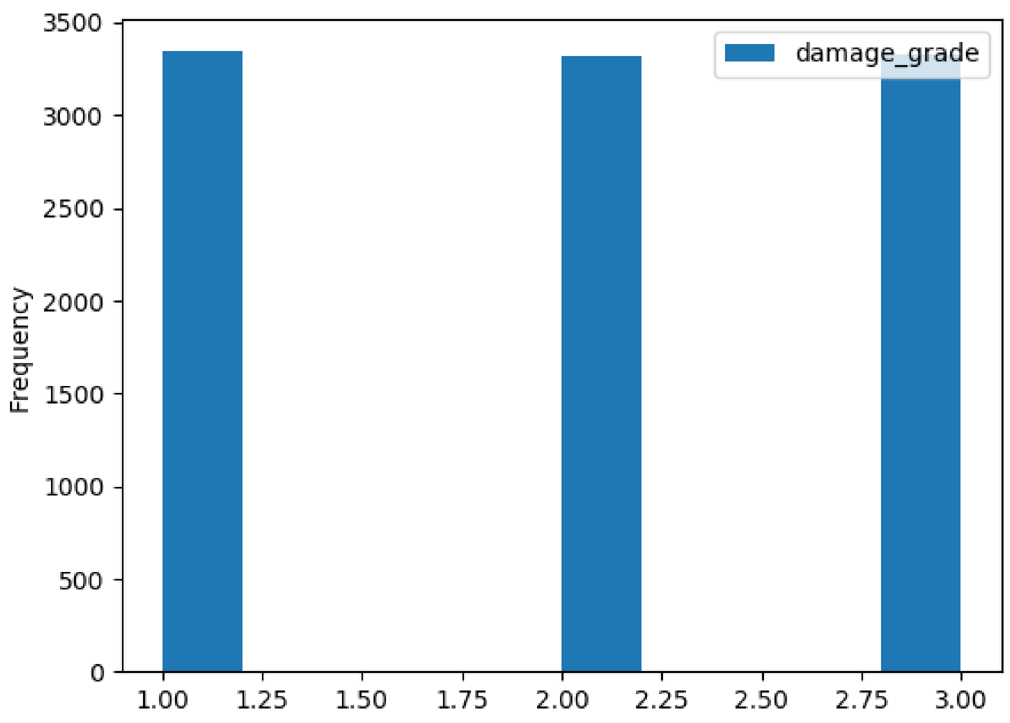
The model was train with the 70% of the data, after splitting the data into separate train and test sets, 70%-30% respectively, using the stratified shuffle split method, that especially helps in the resolution of machine learning problems where the target variable is unbalanced in some of the classes. The first run of this model yielded an accuracy of 55%, which was just confirming the sampling issues observed in the target variable’s distribution.

So in this case, a resampling need was identified, and to help in the completion of this task, some of the most relevant sampling algorithms were compared, as well as the new distributions achieved on the target variable, and the new model results.

The main resampling algorithms considered in the process of solving this problem were:

- All SMOTE’s variants
- ADASYN
- Random Over Sampling (ROS) method

A first test was run with the regular resampling kind of the SMOTE(Synthetic Minority Over-Sampling Technique) algorithm:



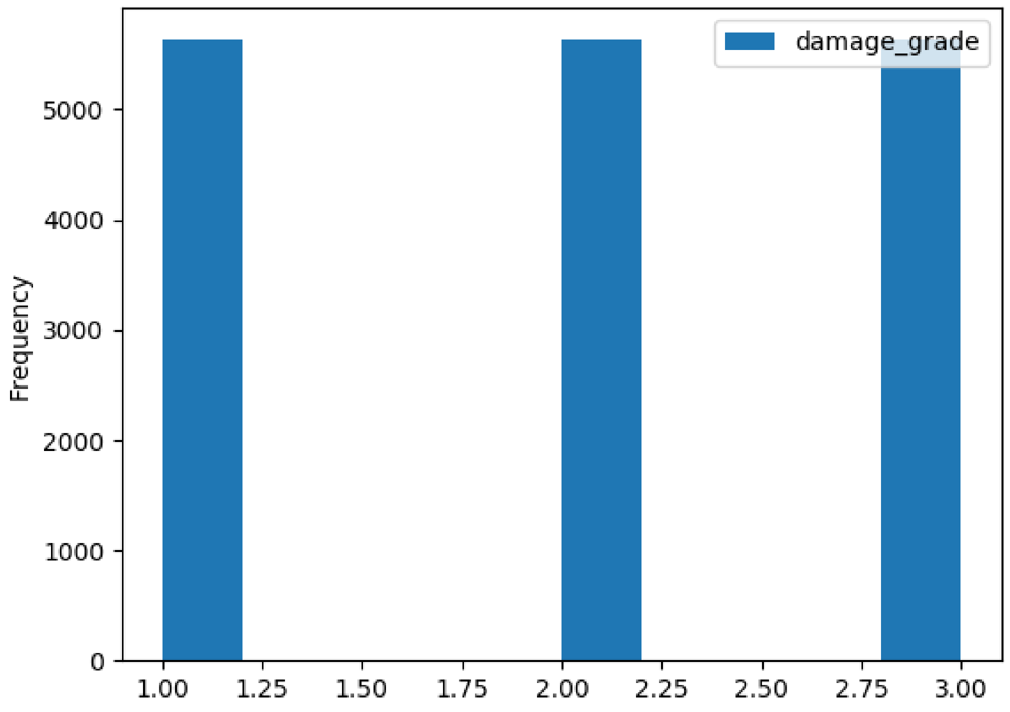
This is what the distribution results looked like after the resampling process. It was clearly better behaved in terms of class distribution, and more adapted to obtain a better machine learning performance.

The next step was then to rebalance the data in terms of samples, given that any over-sampling techniques will synthetically create new samples in our data, this means more rows than in the original dataset, which can lead to an overfitting issue, as the model would be learning from more samples than the originally given. This is desperately needed to keep away at any cost. Fortunately in this case, this was an easy task to solve, and the best way to do it was to take a random sample of the same row size as in the original dataset, which was 10000.

Once this step was complete, everything was set up to run a first model test, and see what the new results looked like after the resampling process was done, and the data was rebalanced to its original size.

A first result of 71% overall accuracy score was obtained in the first test, using a Random Forest Classifier model, with the default hyper parameters configuration, which was much better than the initial result, even if it could be surely improved. This confirmed the path to follow in order to successfully solve this machine learning problem.

The next steps were basically iterations between the over-sampling techniques. As a conclusion for this resampling experiment, it was noted that the ADASYN, and Random Over Sampling methods, did not produced the best results. The final resampling method chosen was the SVM variant of the SMOTE oversampling algorithm, which scored the best in the model tests with an accuracy of 75% percent.



This are the results obtained by the SVM SMOTE variant. This method proved to be the best in terms of performance due to the way it works, resampling those samples that are near to the decision boundary, making them easier to distinguish for any classification algorithm. At the same time it oversamples the minority classes, it also reduces the probability for each of the classes of being misunderstood by the classification algorithms used.

A final note for this part would be the fact that in addition of the over-sampling techniques, other under-sampling techniques were also tested and compared among each other. Also, combinations of over-sampling and under-sampling techniques were applied in the search of the best performance results, but in the end, as stated above, the SVM variant of the SMOTE algorithm was included in the final tests and results.

Model Testing and optimization

Now that the data was finally set up, model testing, comparison and optimization steps were made in order to polish the results and obtain the best possible score for this problem, based on the chosen approach.

At this stage, several potentially helpful algorithms were tested and their results compared.

Regression algorithms were quickly discarded as they were yielding the poorest results, none of them above a 50%-60% in accuracy. The same happened with neural networks algorithms. Tests were made with Keras, and Multi Layer Perceptron Classifiers, and none of them scored above the 50%in overall accuracy terms. Support Vector Machines were especially slow in the training phase, and their score was not superior to 60%.

After several try-error steps to compare potentially useful algorithms, four were found best for this classification problem. The chosen algorithms were:

1. Random Forest Classifier
2. Extra Trees Classifier
3. Gradient Boosting Classifier
4. XGB Classifier

All of these algorithms scored above 75% in overall accuracy. Once the best performing algorithms were found, the model optimization process began. The chosen approach to fulfill this task was to run a random grid search for hyper parameters tuning, since a full grid search, needed much more time and computation power.

Cross Validation

During the hyper parameters tuning phase, and to be sure that the best possible result was obtained, a cross validation process was executed. The two variants of cross validation were tested: Kfold Cross Validation, and Stratified Cross Validation, for each of the fours tested models. The parameter range used for the random grid search used for each of the models were:

1. Random Forest Classifier: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, bootstrap.
2. Extra Trees Classifier: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, bootstrap, criterion.

3. Gradient Boosting Classifier: n_estimators, max_depth, min_samples_split, min_samples_leaf, max_features, criterion, learning_rate, war,_start, oob_score.
4. XGB Classifier: max_depth, n_estimators, silent, learning_rate, objective, booster, min_child_weight, scale_pos_weight.

All of these random grid searches were made over 100 search iterations, this number was increased in the case of the XGB Classifier and the Gradient Boosting Classifier. The fastest ones in terms of computation were the Random Forest Classifier, and the Extra Trees Classifier. Gradient Boosting Classifier and XGB Classifier were extremely slow to complete the random search.

Finally, each model's scores were compared. The f1 micro averaged score of each model was:

1. Gradient Boosting Classifier: 80.76%
2. Random Forest Classifier: 78.96%
3. Extra Trees Classifier: 78.44%
4. XGB Classifier: 75.10%

In the end, the models were really close in terms of their scores, however, the decision was easy to be made, and Gradient Boosting Classifier was chosen as the final model due to its best performance. In order to improve the results, an experiment was made, with each of the four models: Each of the models were nested into One Vs Rest Classifiers, and Bagging Classifiers. Performance tests and iterations over all the possible combinations were made, even another random grid search for the Bagging Classifier, in order the determine its best parameters, and fitting it to the four previous estimators. The computation was long, but it was a worthy effort since the results were indeed improved. The Gradient Boosting Classifier was again the best performer, obtaining a score of 82.36%, followed by the Random Forest Classifier.

In terms of cross validation, the method providing the best results was the Stratified Cross Validation method, as it is best to face imbalanced data problems.

The final score obtained in the validation stage at the Data Driven site was 66.92%.

Conclusions

This analysis has proved that the building's damages caused by the earthquake of 2015 in Nepal can be predicted from the building characteristics. The main features that must be paid attention to are:

- ground_floor_type_467b
- roof_type_67f9
- foundation_type_467b
- has_superstructure_cement_mortar_brick
- other_floor_type_67f9
- has_superstructure_rc_engineered
- foundation_type_858b
- has_superstructure_rc_non_engineered
- ground_floor_type_b1b4
- superstructure
- has_superstructure_mud_mortar_stone
- foundation_type_337f

there are some other secondary features as foundation_type_6c3e, are, other_floor_type_441a, count_floors_pre_eq and other_floor_type_f962, that have also an impact in the damage prediction.

Another important aspect that this analysis have proved is that imbalanced data needs to be processed to make our models capable of correctly learn from the target variable to make predictions that are more accurate. Sometimes this processed must be approached by making iterations and by testing several methods and then compare the results. The same applies to the algorithm selection when facing any machine learning problem. It is critical not to keep the first tried algorithm, but instead, make a deeper approximation and research considering all the possible ways to solve the problem we are facing, and then choose the best adapted for our purposes.

A domain expertise is always the best friend when solving any machine learning problem, in case of a lack of expertise in the work domain is suffered, spending time in research is always helpful to better understand and comprehend the data.

Finally, attached to this better understanding of the data is the fact that the ability to correctly visualize the data is also very important for the purposes of a better analysis.