POLITECNICO DI TORINO

Department of Structural, Geotechnical and Building Engineering

Master of Science in Civil Engineering

MACHINE LEARNING METHODOLOGIES TO ASSES DEBRIS EXTENT AFTER EARTHQUAKE

Supervisor:

Prof. Gian Paolo Cimellaro

UCLA Supervisor:

Prof. Henry Burton

Dissertation of:

Luca Rampini 223989

Academic Year 2017/2018

Alla mia famiglia

ACKNOWLEDGEMENTS

First and foremost, I would like to thank Professor Gian Paolo Cimellaro who gave me the chance to write my master thesis and the opportunity to have this remarkable work experience in a prestigious and important university such as UCLA. I will always be grateful to him also for the advices and the knowledge that directly derives from his collaboration.
I am deeply indebted to Professor Burton to give me a great chance of learning, to guide me in all the process necessary to finish my work and to host me at UCLA; I really appreciate his competence, kindness and availability for all the period that I have been in Los Angeles.
I would also like to extend my thanks to Professor Taciroglu, to give me the possibility to work with his research group, that turned out a continuous source of knowledge.
A huge thank you goes to Antonio and Matteo, who have helped me a lot to improve my knowledge about Machine Learning and the use of Python in general; and made my experience in Los Angeles crisper.

Tutta la mia gratitudine è rivolta ai miei genitori, Brunella e Roberto, per avermi dato non solo la possibilità di perseguire i miei studi al Politecnico, ma anche questa preziosa opportunità di studio all'estero senza far mai mancare il loro supporto.
Un ringraziamento speciale va a mio fratello Giacomo, sempre pronto a dispensare preziosi consigli e che ha sempre rappresentato per me un modello da seguire.
Voglio estendere i miei ringraziamenti anche ai miei amici di sempre, Mirko, Silvio, Nicola, Simone, Andrea, Antonio, Matteo, Nicodemo, Fabio, Paride, Emanuele, Giacomo, Niko e Gabriele; i quali mi sono stati vicini, nonostante la distanza, e che mi hanno reso in larga parte la persona che sono oggi.
Un sentito ringraziamento ad Alessandro, Mattia, Stefano e Daniele, per avermi fatto passare gli ultimi mesi a Torino con una sintonia tale da credere che fossimo sempre vissuti insieme.
Un sentito grazie va inoltre a Miriam, Claudia, Martina, Aurelio, Ludovico, Pierpaolo, Fabio, Desideria, Giulio, Francesca, Vittoria, Simone, Livio, Arturo, Enrico e a tutti gli altri ragazzi che ho avuto la fortuna e l'onore di conoscere in

questi anni a Torino, per avermi accompagnato in questo viaggio rendendolo sicuramente più piacevole ed elettrizzante.
Un caloroso grazie lo devo anche ad Alessio, per avermi sopportato per ben due anni come coinquilino e per essersi rivelato un amico sempre pronto ad aiutare in ogni momento.
Un grazie speciale anche a Mattia R. per aver sempre creduto in me e per la sua amicizia che, sono sicuro, non verrà mai a mancare.

ABSTRACT

Each significant seismic event is an opportunity to assess the performance of our built environment. After such events, a large quantity of varied information is often collected within a short time period.

Therefore, a quick and reliable method for detecting damage to buildings is required for disaster response with the intent of reducing both human and economic losses.

In the last few years, the increasing advancement of technologies and Computer Science has permitted the adoption of Machine Learning techniques in various fields, included Civil Engineering and Urban Resilience.

One of the most influence parameters in the evaluation of seismic loss is the extension of the debris and their associated effects on critical infrastructure. Debris accumulation can result in disruption of the road network and compromise rescue operations. This implies an overall increase in the average number of people who have difficulty evacuating, with a significant risk that some residents cannot evacuate at all.

Starting from the newest Machine Learning algorithms and using Python programming language, the purpose of this study is to collect different features and data from the main seismic events that occurred in the past several decades and use them to forecast the extension of building-rubble after earthquakes. This type of model would ultimately allow civil protection agencies to plan their rescue operations while reducing the risk of getting stuck by extended rubble.

Finally, the PHI Challenge sponsored by PEER, gives a chance to introduce Deep Learning methodologies for Image Recognition tasks related to Earthquake Reconnaissance. This topic indeed, is one of the most popular argument in the state of the art that connects Machine Learning techniques and Civil Engineering.

SINTESI

Ogni evento sismico di una certa rilevanza rappresenta una opportunità per valutare le prestazioni dei nostri edifici e delle nostre infrastrutture. Infatti, dopo tali eventi, una grande quantità di informazioni è raccolta entro breve tempo.
Pertanto, una rapida ed affidabile metodologia è necessaria per valutare velocemente i danni strutturali, allo scopo di ridurre le perdite umane ed economiche.
Negli ultimi anni, la rapida crescita delle tecnologie e della Computer Science, ha permesso l'utilizzo di tecniche di Machine Learning nei più svariati campi, incluso l'Ingegneria Civile e la Resilienza Urbanistica.
Uno dei parametri più influenti per la valutazione delle perdite, dovute ad eventi sismici, è l'estensione dei detriti e gli effetti associati ad essa sulle infrastrutture di importanza strategica. L'accumulo di macerie può infatti provocare l'interruzione della rete stradale, o parte di essa. Tutto ciò implica un generale aumento delle persone che incontrano difficoltà ad evacuare determinate zone, con l'effettivo rischio che alcune persone non riescano affatto ad evacuare in tempo.
Utilizzando gli algoritmi di Machine Learning più aggiornati e il linguaggio di programmazione Python, lo scopo di questa tesi è di raccogliere dati ottenuti dagli eventi sismici precedenti, per predire l'estensione dei detriti. Questo modello può, come fine ultimo, permettere agli enti di Protezione Civile di pianificare gli interventi di salvataggio, riducendo il rischio di essere bloccati dalle macerie durante il percorso.
Infine, la PHI Challenge organizzata dal PEER, fornisce l'occasione giusta per introdurre le tecniche di Deep Learning applicate al riconoscimento automatico di Immagini da dividere in categorie legate all'Ingegneria Sismica. Tale argomento infatti, risulta essere uno dei più popolari e discussi nello stato dell'arte attuale della ricerca che collega Machine Learning all'Ingegneria Civile.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1
INTRODUCTION

After every disaster, a huge number of images are collected by reconnaissance teams formed by professional engineers, academic researchers and graduate students.
All of them are charged with collecting perishable data to be used for:

- learn as much as possible about the nature of the event and the extent of the consequences;
- identify potential gaps in existing research or in the application of engineering knowledge;
- make recommendations regarding the need of further investigations and/or changes to codes and laws.

All this collected data can be used to reduce social and economic losses that follow strong ground motions. Indeed, after a seismic event, often a large amount of debris is generated that might be a critical obstacle to emergency operations and evacuations.
Starting from existing collections of data, the purpose of this thesis is to forecast the extension of the debris and, eventually, to determine whether a road is blocked or not.
In this way, it is possible to lead a procedure that can predict the best path for rescue operations and increase the possibility for a better action in term of human security and time safe.
Using the methodologies of modern Machine Learning (ML), it is possible to evaluate the accuracy of our predictions and increasing constantly the precision adding more and new data that could be collected in the future events.

Because most data collected by reconnaissance teams are pictures, it was considerate appropriate obtain evaluations about the amount of rubble from them.
Indeed, this parameter is not directly reported inside the reports that are compiled around the world and the future availability of new data, taken directly on field, could improve significantly the results of this study.
Once the extension of debris is extracted from the dataset (as shown in Chapter 2), the most famous and important ML algorithms is used to forecast a prediction.
ML indeed, represents one of the most powerful and modern instruments to make predictions and is widely used around the world in different fields.
The first result of the present research is to forecast, for new data, the amount of rubble, giving some inputs as magnitude, height, material and distance from epicenter.
Finally, it is described another important aspect that connects ML and Earthquake resilience: Image Recognition (IR). This aspect continues to take the field in the Earthquake Engineering investigations and proof of this is the challenge sponsored by PEER. Using Google Cloud Project (GCP) and the deep learning most modern models, the concerned thesis covers also this aspect.

1.1 Python, tools and modules

Running ML algorithms required a good knowledge of programming language and Computer Science. One of the powerful and widely used programming language is Python, an open-source project that spreads out in the last few years.

The power of this language lies on his flexibility due to the versatility of his modules and tools. Unlike MATLAB, indeed, a lot of functionalities could be downloaded as different packs and installed separately from the main program.

This allows the availability of a huge number of different packages, easy to install and to use for every purpose. In the list below is listed the modules used in this research and what is their function:

- Scikit-learn: this module is constantly being developed and improved, thanks to a very active community. It contains several state-of-art ML algorithms, as well as comprehensive documentation about each algorithm.
- Numpy: surely one of the fundamental packages for data analysis in Python. It contains functionality for multidimensional arrays, high-level mathematical functions such as linear algebra operations.
- Scipy: is a collection of functions for scientific computing. It provides special mathematical functions and statistical distribution.
- Matplotlib: is the primary scientific plotting library in Python. It provides functions for making publication-quality visualizations such as line charts, scatter plots, and so on.
- Tensorflow: is a powerful library used for train the Convolutional Neural Network (CNN) models for the PHI Challenge
- Keras: is a high-level neural networks API, written in Python and capable of running on top of TensorFlow. All ingredients within CNN mentioned in previous paragraphs are available in Keras, one only need to call those function in a reasonable order.

1.2 Layout of the Thesis

The present work is divided into seven chapters, described below:

- *Chapter 2* shows how the dataset is built and how the extension of the debris is calculated for each sample.

- *Chapter 3* presents the description of all the ML algorithm that are used to train, test and validate the data.

- *Chapter 4* describes how the training and preprocessing operations are conducted and visualizes the dataset to give important insights for how to use them.

- *Chapter 5* summarizes the testing phase and give the results in terms of accuracy for each ML algorithm.

- *Chapter 6* describes the PHI Challenge organized by the Pacific Earthquake Engineering Research (PEER) about Image classification for detecting damage in buildings from pictures taken after seismic event.

- *Conclusions* at the end resume what has been done in the present work, running through the main features. This chapter gives an interpretation about the results as well as final consideration about how improves them and how could be conducted further researches on this field.

CHAPTER 2
DATA COLLECTION

This chapter describes in detail the process of create the dataset used for train and test the ML algorithms. The sources of the images are mentioned as well as the methodology used to extracting information from that.

2.1 Dataset sources

In a typical seismic evaluation mission, a group is dispatched to a region where a seismic event has taken place. The teams may follow established guideline, that can be vary nation by nation, to provide a well-informed dataset, where it is possible to obtain important information as location of the damaged buildings, severity of the event, materials involved in the damaged and so on.
In the recent years many databases, publicly available, come out and they are usable for every kind of research and activities.
The dataset used for this thesis comes from these main sources:

- EERI clearinghouse: The Earthquake Engineering Research Institute (EERI) is a national, non-profit, technical society of engineers, geoscientists, architects, planners, public officials, and social scientists. EERI members include researchers, practicing professionals, educators, government officials, and building code regulators. The objective of the Earthquake Engineering Research Institute is to reduce earthquake risk by (1) advancing the science and practice of earthquake engineering, (2) improving understanding of the impact of earthquakes on the physical, social, economic, political, and

cultural environment, and (3) advocating comprehensive and realistic measures for reducing the harmful effects of earthquakes.

- Datacenterhub.org: as written on the internet site: “Too often, the results of years of engineering and scientific inquiry are stored on media that become outdated, broken, misplaced, or are simply not accessible. Even when these data are accessible, it can be difficult to interpret them. DataCenterHub seeks to alleviate this problem by providing a simple, standardized yet flexible platform to preserve and share data. In the future, this platform will offer data visualization tools and the ability to compare directly data from different sources. We plan to archive each uploaded dataset on Purdue’s institutional repositories (FORTRESS and PURR) to ensure the longevity of the data contributed to DataCenterHub. While funding is available, we shall also maintain a more readily accessible copy of all data on DataCenterHub.
- Db.concretecoalition.org: This database contains over 50 case studies on concrete buildings damaged in over 20 earthquakes. Each case study provides detailed information about the building and photographs of damage.
- GEER: The Geotechnical Extreme Events Reconnaissance (GEER) Association was formed as an outgrowth of grassroots efforts to investigate and document the geotechnical impacts of the 1989 Loma Prieta Earthquake, 1994 Northridge Earthquake, and 1995 Kobe Earthquake. Following these earthquakes, members of the geotechnical earthquake engineering community responded with ad hoc reconnaissance teams that relied on past personal and professional relationships.

These databases provide thousands of images (around 15000), but the purpose of the thesis is to study only pictures where the extent of the rubble is clearly visible and could be measured and compared with other elements present inside the photographs (as shown in the instance below). For that reason, a big effort was required to select manually each picture from these sources to build the dataset necessary for the concerned research.

Figure 2.1 Instance template of the images forming the dataset

Following this approach, a database of 198 photographs after past earthquakes events is gathered from these sources and each picture shows an entire building (or, at least, a good portion of it) that was damaged or collapsed by an earthquake event and his extension of rubble after damage.

The pictures cover an amount of 14 different earthquakes as shown in Table 2-1:

Table 2-1 Number of photographs for each earthquake event

Earthquake	**Pictures**
Central Italy	38
Cephalonia (Greece)	6
Christchurch (New Zealand)	9
Ecuador	38
India	6

Loma Prieta (US)	5
Mexico Central	20
Nepal	38
Northern Iran	1
Northridge (US)	2
North western Armenia	5
South Napa Valley (US)	6
Southern Taiwan	26
Turkey	1

In the following paragraphs, a brief description about the earthquakes as well as some useful considerations, are provided for each earthquake (except for Loma Prieta, Northern Iran, Northridge and Turkey, where less than 6 samples are taken from these earthquakes).

2.2 Earthquakes

2.2.1 Central Italy

The Central Italy earthquake it happened in two different events, the first on 24 August 2016 and the second on 26 October 2016. The magnitude of these Earthquakes was around 5.7. The main part of building hit by the ground motion was built in masonry and dates from more than one hundred years ago.

Figure 2.2 Torre dell'Orologio in Poggio Renatico (FE)

2.2.2 Cephalonia

In 2014 a shake hit the island of Cephalonia with a Magnitude of 6. The general damage was not too heavy, but some buildings lost some debris and create rubble on the streets.

Figure 2.3 Damaged building in Cephalonia

2.2.3 Christchurch

An earthquake with a magnitude of 6.3 hit the city of Christchurch in New Zealand on 22 February of 2011. This event caused a lot of building damage, including the Cathedral, and an amount of 185 victims.

Figure 2.4 Cathedral of the Blessed Sacrament Catholic in Christchurch

2.2.4 Ecuador

The 16th of April 2016 a 7.8 Magnitude seismic event occurred in Ecuador. It is one of the most tragic events for the nation since 1987 and the number of damage and victims is high. A lot of buildings, both in masonry and in reinforced concrete, were damaged, as well as some infrastructures.

Figure 2.5 Concrete building damaged in Ecuador

2.2.5 India

The 2001 India earthquake, occurred in 26th January, had a magnitude of 7.7. It caused more than 10'000 victims and destroyed a lot of buildings and infrastructures.

Figure 2.6 Hotel damaged during the Earthquake occurred in India

2.2.6 Mexico Central

A 7.1 of magnitude earthquake hit the central part of Mexico on 19 September 2017. The earthquakes caused damage in the states of Puebla and Morelos and around 370 people was killed during the seismic event.

Figure 2.7 A building collapse during the Central Mexico earthquake

2.2.7 Nepal

In 2015, on 25th of April, a seismic event with a magnitude of 8.0 hit the nation of Nepal. It was the worst natural disaster from 1934 and caused thousands of victims and around $10 billion of total damage. Almost all the buildings were made in masonry, so the amount of debris was very huge in this case.

Figure 2.8 Damage in Nepal building after the seismic event

2.2.8 South Napa Valley

On 24th August 2014, in the south of Napa, an earthquake of 6.0 magnitude was occurred. The event was the biggest in the San Francisco Bay Area since the 1989 Loma Prieta earthquake.

Figure 2.9 Damaged building in South Napa after earthquake

2.2.9 Southern Taiwan

A 6.4 magnitude earthquake occurred in the Southern Taiwan on 6th February 2016. Tainan was the city most hit by the event with a lot of building damage.

Figure 2.10 Removal of debris caused by a damaged building after the seismic event in Southern Taiwan

2.3 Dataset features

Quite possibly the most important part in ML process is understanding the data you are working with and how it relates to the task you want to solve.
In a larger context, the algorithms and methods in ML are only one part of a greater process to solve a problem, and it is always good to keep the big picture in mind.
Indeed, no ML algorithm will be able to make a prediction on data for which it has no information.
For this reason, is extremely important feed the dataset with several features that allows to treat the problem and forecast a solution.

Looking at the purpose of this thesis, the following attributes have been kept in consideration because strictly connected with the resultant amount of debris that has been collected for each sample:

- Material: the buildings considered in this study are made in reinforced concrete or in masonry. The behavior of the two materials is quite different in term of extension of debris.
- Stories: tallest buildings could produce a bigger amount of rubble and could be interest see if there is a relation between height and extension of debris.
- Year of construction: construction technologies and methodologies can vary during the years because of both different regulations and construction technique.
- Magnitude: stronger is the ground motion and more are the possibilities to damage a building.
- Distance from epicenter: as the magnitude, buildings close to epicenter suffer from a stronger ground motion.
- Height: This parameter influences the seismic force (and, in consequence, the damages) that is felt by the structure since the Natural Period depending strongly from this value.

The complete table with all the pictures' information is shown in the appendix 8.1.

2.3.1 Debris extension

The purpose of this thesis is to forecast the extension of debris after earthquake based on different parameters of the buildings. Unfortunately, this information is not reported inside the report that usually are written after every event.
A method to evaluate the rubble's dimension from pictures is requested as explained in the next paragraphs.
Starting from a photograph, it is identified an object or a dimension, which the measure is well known; in the instance below, which correspond to the sample id.061 of the dataset, the standard width of a car is used as reference.

Figure 2.11 The width of a car is used as reference measure

Using PhotoFiltre, a photo editing program, it is possible evaluate the coordinates of the pixel of the two extreme points of a diagonal in the red square (both x axis and y axis).

Same operation is done to calculate the extension of the debris inside the picture.

Figure 2.12 Evaluation of the debris extension

Using a proportion (equation (1.1)), is possible to obtain an estimation of the rubble's amount along one axis (x-axis in the instance):

$$(1.1)\, d = p \cdot \frac{P}{D}$$

Where:

- $P = x_{1_r} - x_{2_r}$ (x_{1_r} and x_{2_r} are the pixel coordinates (x or y) of the reference element)
- $p = x_{1_d} - x_{2_d}$ (x_{1_d} and x_{2_d} are the pixel coordinates (x or y) of the debris extension)
- D is the value in meters of the principal measure of the reference element
- d is the value in meters of the debris extension

Completing the instance, the parameters used to evaluate the rubble is summarize in the Table 2-2:

Table 2-2 Parameters to evaluate the extension of debris in the instance id.061

Coefficient	**Value**
x_{1_r}	502
x_{2_r}	334
x_{1_d}	720
x_{2_d}	371
p	349
P	168
D	1.8 m
d	3.629 m

Once the extension of the debris is evaluated for all possible pictures, the estimation that is obtained is divided by the height of the building.

This operation is called “normalization” and is very common when samples, used for a ML study, are collected. This process is very useful for a series of reason that are listed here:

- Makes training data less sensitive to the scale of features: in this study the height of a building heavily influences the general amount of the rubble.
- Minimize the variance of the dataset
- Consistency for comparing results across models: if other studies decide to cover this argument with other methods, with normalization is easier to compare results.
- Makes optimization well-conditioned: most of the ML optimizations are solved using gradient descent and the speed of convergence depends on the scaling of features.

For completeness, all the parameters used to calculate the extension of the debris are reported in the appendix section 8.2.

2.4 Dataset

Once the features and the methodologies required to build the dataset are shown, it is appropriate to summarize for each feature how many samples are available.
Indeed, not for all the pictures it was possible to collect every parameter because the reports written all around the world are made by different teams and people, with different methodologies and instruments.
Therefore, the next Table 2-3 is helpful to understand and see, for each feature, how many samples are collected.

Table 2-3 Quantity of samples for each feature

Feature	**Quantity of samples**
Material	198
Stories	198
Year of construction	43

Magnitude	198
Distance from epicenter	170
Height	198

If there is a greater interest in the values and the composition of data, the complete dataset is available in the appendix section 8.1.

CHAPTER 3
MACHINE LEARNING METHODOLOGIES

3.1 Machine Learning

Arthur Samuel in 1959 defined Machine Learning as:

> *"Field of study that gives computers the ability to learn without being explicitly programmed"*

As the definition suggest, ML is about extracting knowledge from data. It is a research field at the intersection of Artificial Intelligence (AI), statistics and Informatics.

From the beginning of Computer Science and programming the idea behind ML was always well known by the scientists in the field of Informatic Engineer.

However, the limited capacity both in term of calculation power and storage data capacity of the computers never allowed a practical and fruitful use of this camp.

Only recently, ML methodologies has become ubiquitous in everyday life, thanks to the great improvement registered in the field of Informatic and computers.

From detecting spam, recognizing people inside photos and automatic recommendations of which movies to watch, many modern website and company have ML algorithms at their core.

The most successful kinds of ML methodologies are those that automate decision-making processes by generalizing from known examples, which is known as *supervised learning*. The concerned study provides the algorithm with pairs of input

and desired outputs in order to allow the algorithm to find a way to produce a desired output given an input.

The concerned study is treated as a supervised learning problem with a regression task, where the answer to be learned is a continuous value.

3.2 Machine Learning algorithms

There are a lot of algorithms available in literature, so it is important to understand which one could be useful for the purpose (and is also important to have a deep comprehension of the data by visualizing them, as it will be described in Chapter 4).

For this reason, Scikit-learn provides the chart shown in Figure 3.1 that help to choose the correct algorithm based on the dimension of the dataset and the problem that is dealt with.

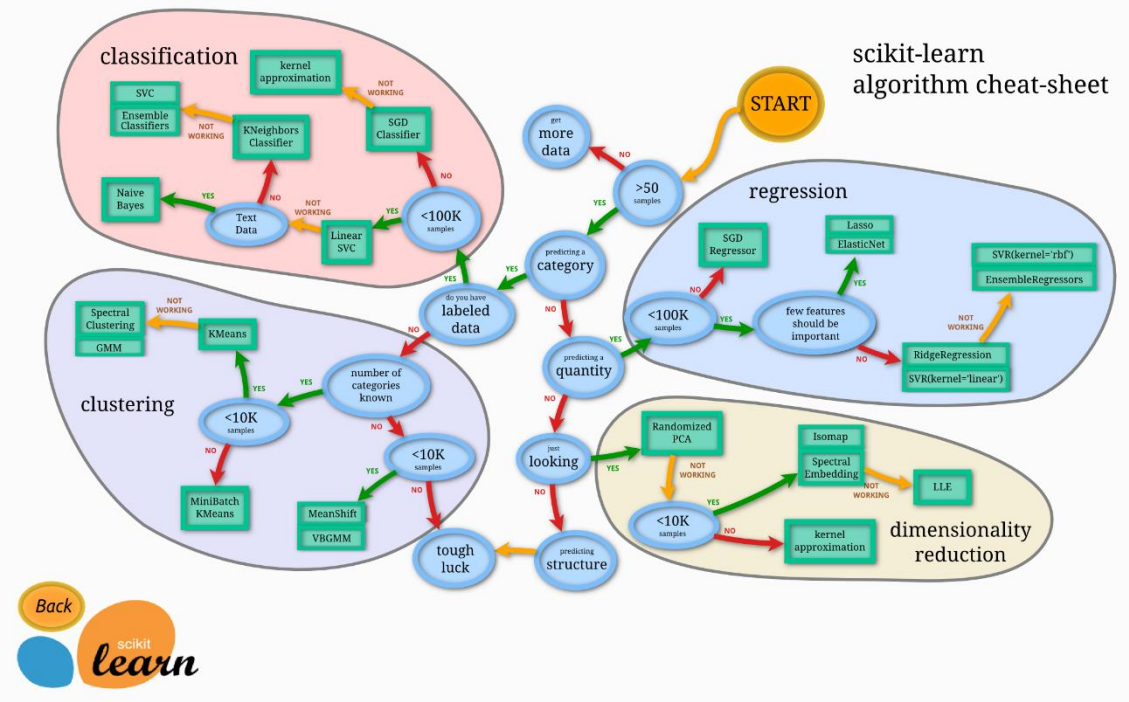


Figure 3.1 Scikit-learn diagram to choose the correct algorithms.

In the next paragraphs, all the different typologies of algorithms are introduced and explained as well as strengths and weakness for each method.

3.2.1 k-Nearest Regressor

The k-Nearest Neighbours (KNR) algorithm is arguably one of the simplest. It makes prediction for a new data point starting from the closest data in the training datasets, its “nearest neighbours”.

The k stands for how many samples are used to evaluate the prediction; in the following Figure 3.2 and Figure 3.3 is shown how change the prediction varying the value of k from 1 to 3.

In the first case, the target value is the same of the closest data. In the latter, the prediction is given by the mean of the relevant neighbours.

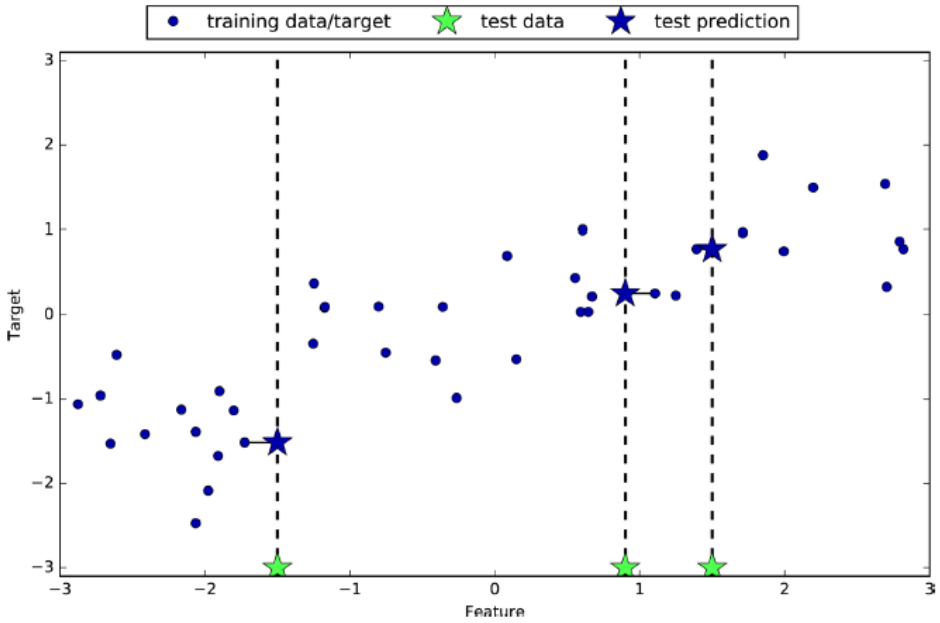


Figure 3.2Predictions made by k=1

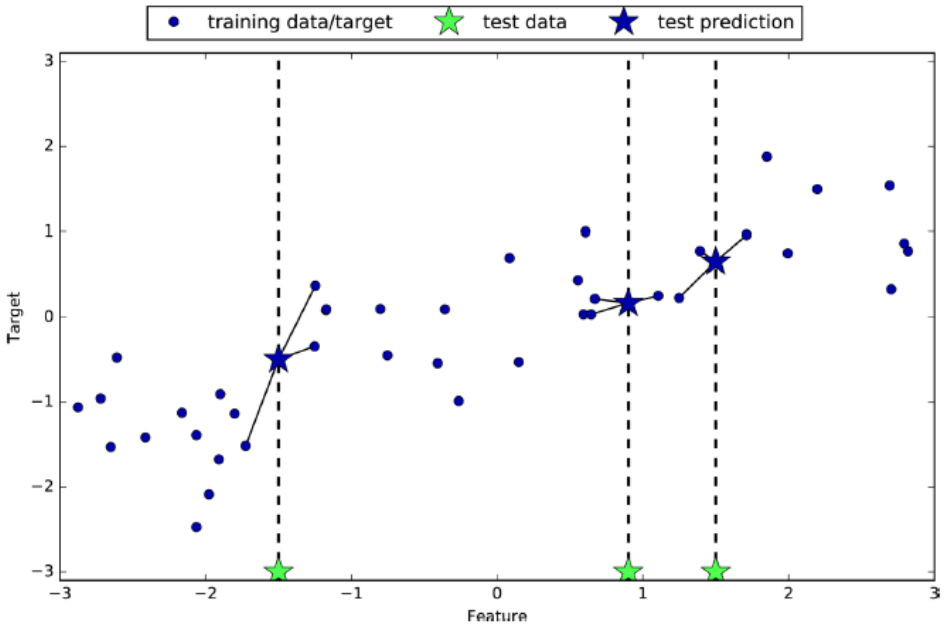


Figure 3.3 Predictions made by k=3

In conclusion, there are two important parameters for the KNR algorithm: the k value and how you measure distance between data points. Usually, the number of neighbours used is not more than 5 (to avoid overfitting, as it will be discussed in the next Chapter) and the lengths are measured by the Euclidean distance.

One of the strengths of KNR is that model is very easy to use and understand, and often give good result without a lot of adjustment.

On the other hand, with a large training set, the prediction can be slow, and the reliability of the result is less accurate.

3.2.2 Linear models

Linear models are the oldest method and is widely used for his simplicity and robustness.

The general prediction formula in this case, is represented by the following (3.1) equations:

$$(3.1)\ y = w_0 \cdot x_0 + w_1 \cdot x_1 + \ldots + w_n \cdot x_n + b$$

Where x_0 to x_n denotes the features and w and b are parameters of the model that are learned.

In conclusion, linear models try to predict labels approaching the training data as a straight line.

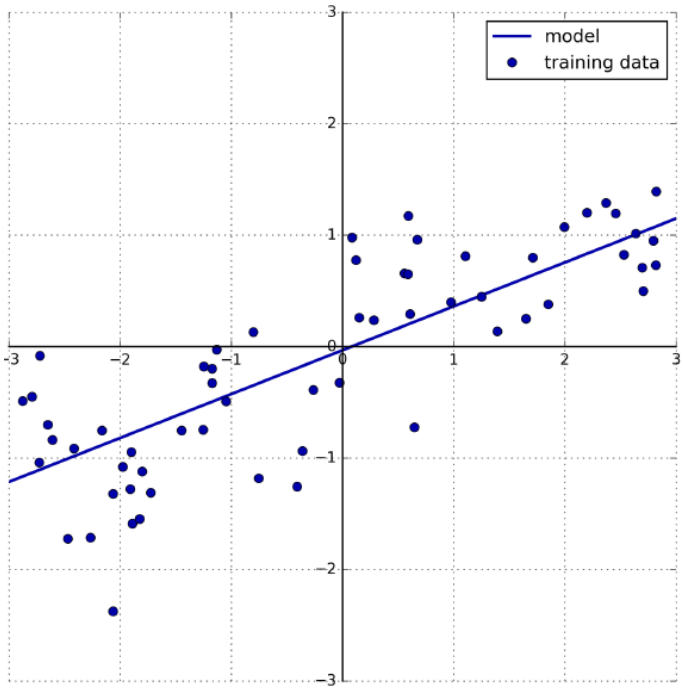


Figure 3.4 Predictions of a linear model

In literature, there are a lot of different linear models and their difference lies in how the parameters w and b are learned from the training data.

In this concerned study, the most popular linear regression models are used to predict the extension of debris.

3.2.2.1 Ridge regression

In ridge regression, the coefficients (w) are chosen not only so that they predict well on the training data, but also to fit an additional constraint.

The target is to have the magnitude of coefficients to be as small as possible and so, all entries of w should be close to zero. In other words, this means each feature should have as little effect on the outcome as possible (which translates to having a small slopes), while still predicting well. This is a clear example of "regularization", where the model is restricted to avoid overfitting.

How much simplicity is placed in the model is decided by the alpha parameter; increasing alpha, indeed, forces coefficients to move more toward zero, which decreases training set performances but might help generalization.

3.2.2.2 Lasso

The last linear model that is presented is Lasso. As with Ridge regression, also lasso restricts coefficients to be close to zero, but in a slightly different way. Indeed, in this case some of the coefficients are exactly zero. This means some features are completely ignored by the model; and this be a form of automatic feature selection. Also lasso has a regularization parameter alpha, that controls how strongly coefficients are pushed toward zero.

3.2.2.3 Elastic Net

Elastic net is a hybrid of ridge regression and lasso regularization. Like lasso, elastic net can generate reduced models by generating zero-valued coefficients. Empirical studies have suggested that the elastic net technique can outperform lasso on data with highly correlated predictors.

3.2.2.4 Linear models' strengths and weaknesses

Linear models are very fast to train, and fast to predict. They scale to very large datasets and work well with sparse data. Another strength of linear models is that they make it relatively easy to understand how a prediction is made, using the formulas we saw earlier for regression and classification. Unfortunately, it is often not entirely clear why coefficients are the way they are.

3.2.3 Decision Trees

Decision trees are widely used models for regression tasks. Essentially, they are learning a sequence of if/else questions that gets us to the true answer most quickly. In the ML settings, these questions are called "tests".

To build a tree, the algorithm searches over all possible tests and finds the one that is most informative about the target variable (see Figure 3.5).

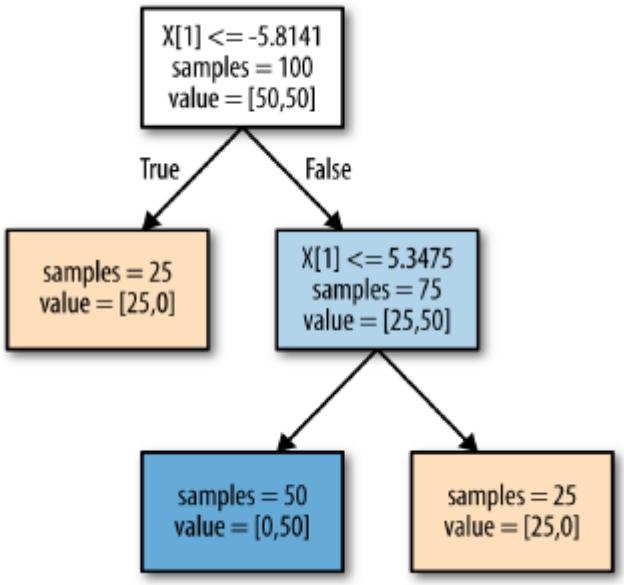


Figure 3.5 Decision tree learned on the data

Typically, using pure leaves (means that all the training data are at the end of the tree) leads the model to overfit the training data. To prevent that, there are two commons strategies: stopping the creation of the tree early (also called "pre-pruning") or building the tree but then removing or collapsing nodes that contain little information (also called "post-pruning" or just pruning).

Decision trees have two advantages over many of the algorithms that are discussed: the resulting model can easily be visualized and understood by nonexperts, and the algorithms are completely invariant to scaling of the data.

The weakness of this method otherwise, still lies in the risk of overfitting data providing poor generalization, despite the use of pre-pruning or post-pruning.

Therefore, in most applications, the ensemble methods are usually used in place of a single decision tree.

3.2.3.1 Random Forests

As we just observed, a main drawback of decision trees is that they tend to overfit the training data. Random forests are one way to address this problem. A random forest is essentially a collection of decision trees, where each tree is slightly different from the others. The idea behind random forests is that each tree might do a relatively good job of predicting but will likely overfit on part of the data. If we build many

trees, all of which work well and overfit in different ways, we can reduce the amount of overfitting by averaging their results.

To implement this strategy, we need to build many decision trees. Each tree should do an acceptable job of predicting the target and should also be different from the other trees. Random forests get their name from injecting randomness into the tree building to ensure each tree is different. There are two ways in which the trees in a random forest are randomized: by selecting the data points used to build a tree and by selecting the features in each split test.

3.2.4 Support Vector Regression (SVR)

In the SVR it is defined as hyper plane the line that will predict the continuous value that is the subject of the research.

On the other hand, two boundary lines are situated at the distance 'ε' from the hyper plane.

The SVR algorithm counts the number of data that are inside the boundaries and the best fit line is the one that contain the maximum number of data.

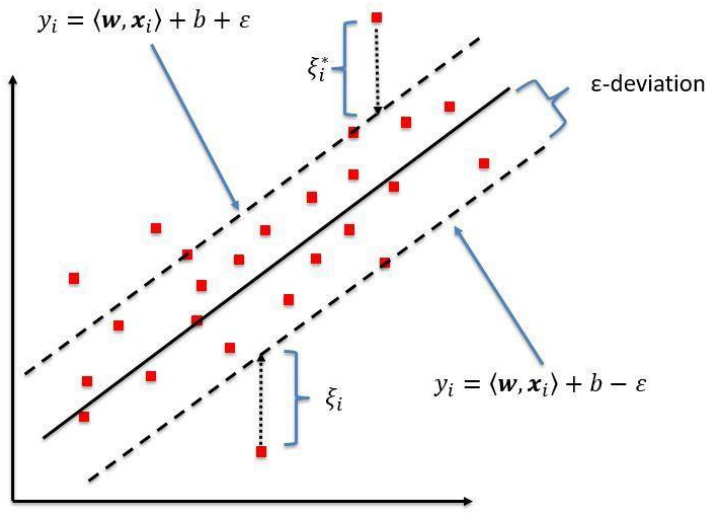


Figure 3.6 Instance of SVR fit line and boundaries

3.2.5 Multilayer Perceptrons (MLP) Regressor

In the last few years a new methodology spreads out and it is so important that is considered a new field of ML, under the name “deep learning”. The main task for this method is Image Recognition where, thanks to the huge improvements in terms of both hardware power and number of data, in the last decade impressive results are reached.

MLP is a Convolutional Neural Network (CNN) that try to simulate the same procedures that occur in the neurons of the human brain. Each processing units indeed, behaves like neuron and therefore calculate a weighted sum of its inputs. Then, uses an activation function to form the single output value.

Usually, in a CNN we have an input layer where the features are collocated, then other layer called hidden, where the weights are calculated step by step and finally the output value.

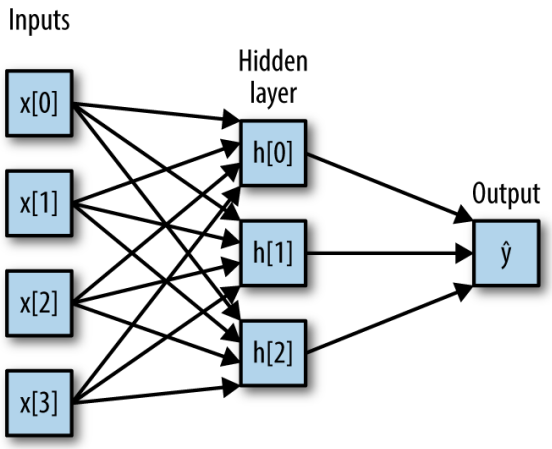


Figure 3.7 Illustration of a multilayer perceptron with a single hidden layer

CHAPTER 4
DATA VISUALIZATION AND PREPROCCESING

4.1 Data visualization

The data representation is one of the main tasks of a data scientist and ML practitioners trying to solve and it is known as feature engineering.
Indeed, visualize the data in an efficient way can have a bigger influence on the results than the parameters tuning.
The purpose of this research is to estimate the extension of debris (EOD), so, in the next charts, is shown this quantity along the y-axis, meanwhile, in the x-axis are represented 5 different features (one for each graph).
In each graph, the data are divided in two categories: masonry and reinforced concrete building, as indicated in the label.

4.1.1 Stories-EOD

The number of stories of a building is not a continuous number, but a discrete one. For this reason, all the sample are situated only in the integer positions of the chart below.

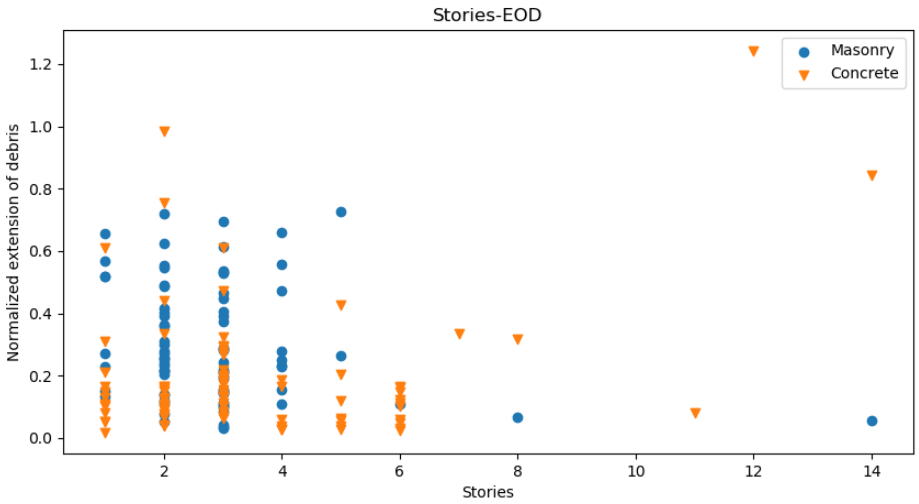


Figure 4.1 Data representation: Number of stories-EOD

4.1.2 Year-EOD

The year of construction is an interesting feature but unfortunately is very difficult to find this kind of information. Indeed, the number of samples of this graph is the smallest.

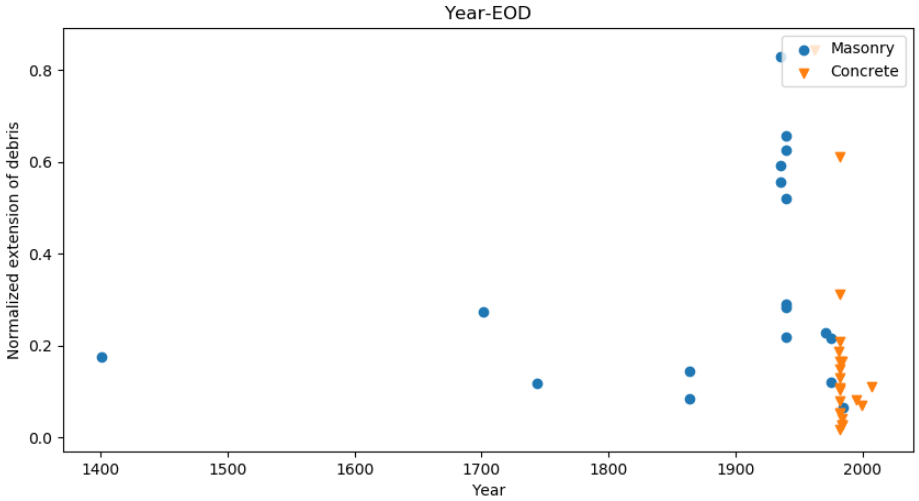


Figure 4.2 Data representation: Year of construction-EOD

4.1.3 Magnitude-EOD

The data are grouped in 11 x coordinates because these are the magnitude of the different earthquakes.

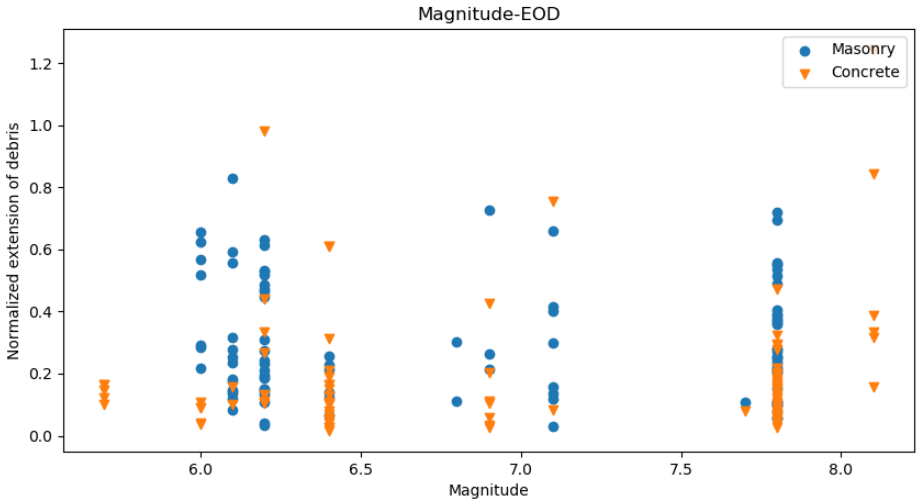


Figure 4.3 Data representation: Magnitude-EOD

4.1.4 Distance from Epicenter-EOD

The distance from epicenter is expressed in kilometers and is measured taking the geographic coordinates of building and epicenter (so it's measured in the surface).

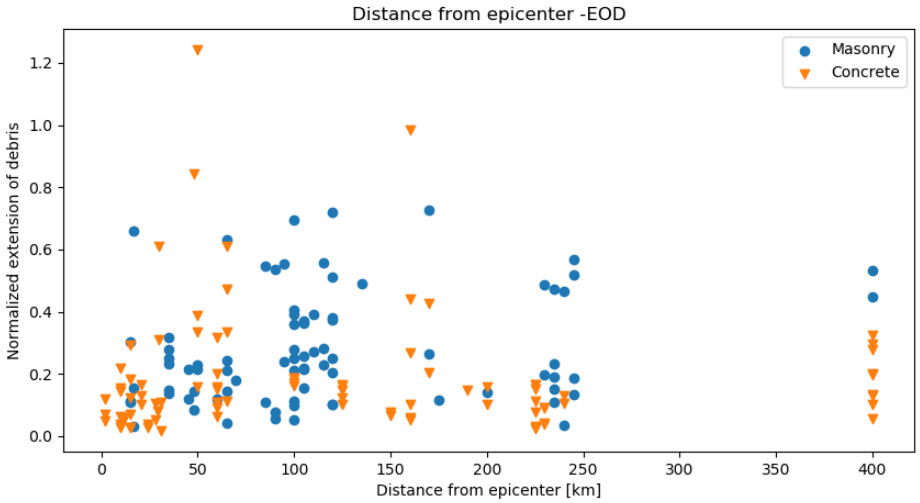


Figure 4.4 Data representation: Distance from epicenter-EOD

4.1.5 Height-EOD

This is the features with more and well distributed data.

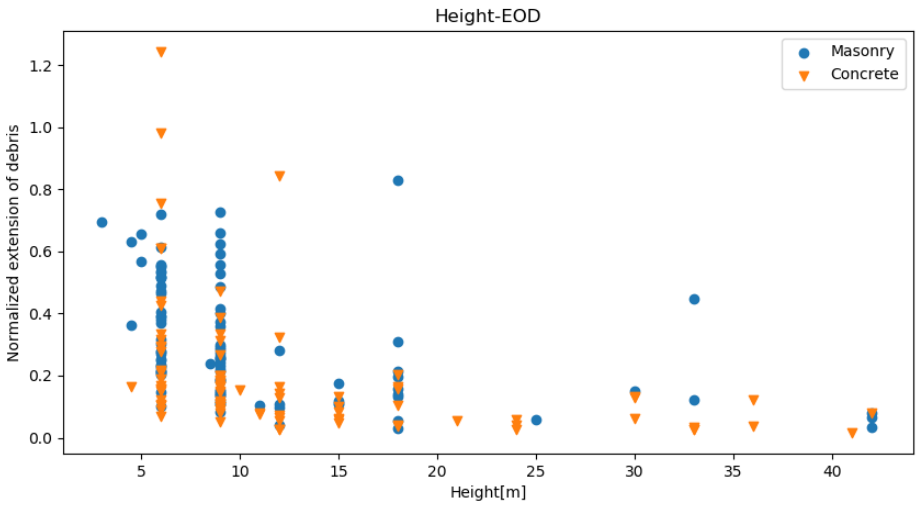


Figure 4.5 Data representation: Height-EOD

4.1.6 Logarithmic values

Visualize the data help to understand if a correlation between features and target is possible. Looking to this data it is possible to say that linear correlation will not fit properly the dataset so the accuracy of the most part of the algorithm will not be high. Therefore, to improve the accuracy of our forecast, it is appropriate to use the logarithmic values of our data with the purpose to get a better linearization of the problem. In Figure 4.6 it is shown the new distribution of the data and it is clearly visible a higher linearization than the previous chart in Figure 4.5.

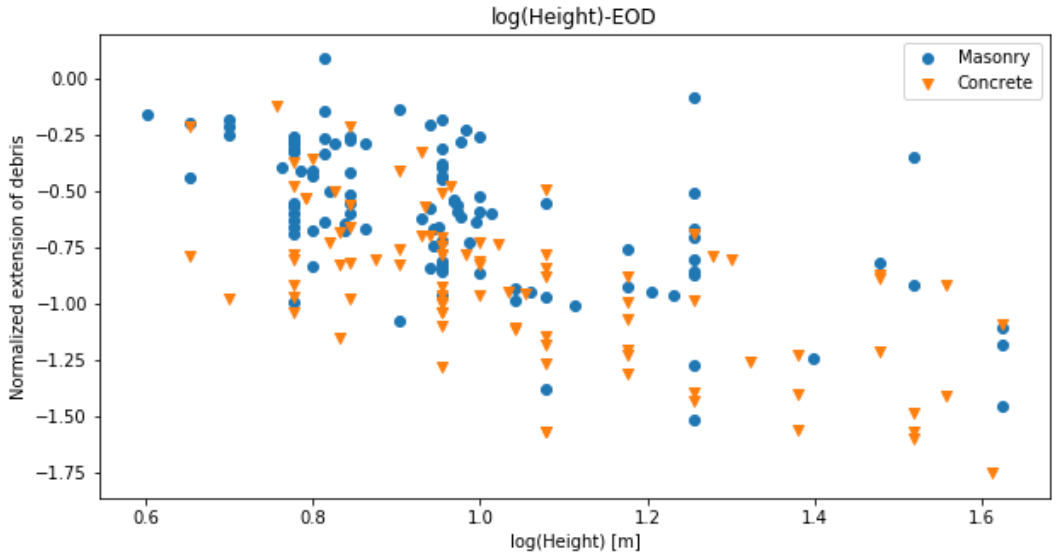


Figure 4.6 Data representation: Height-EOD with logarithmic values

4.1.7 t-Distributed Stochastic Neighbor Embedding (t-SNE)

Since our problem is a multifeatured regression in 6 dimensions is useful utilize some tools that allow a sort of 2D visualization of the data in order to make more comprehensive the distribution of them. In 2008 an algorithm called t-SNE was developed to visualize high dimensional data using Barnes-Hut approximations.

Using this algorithm for the dataset related to this research, the result is shown in Figure 4.7.

Looking at the graph it is possible to say that the R-squared value hardly could has a high score because for the same x-value on the graph we have more than one data in the y-axis (forward is explained better how works the R-squared coefficient).

Its useful repeat that the graph has only a visual importance, therefore the values in the axes have not any meaning for the concerned problem.

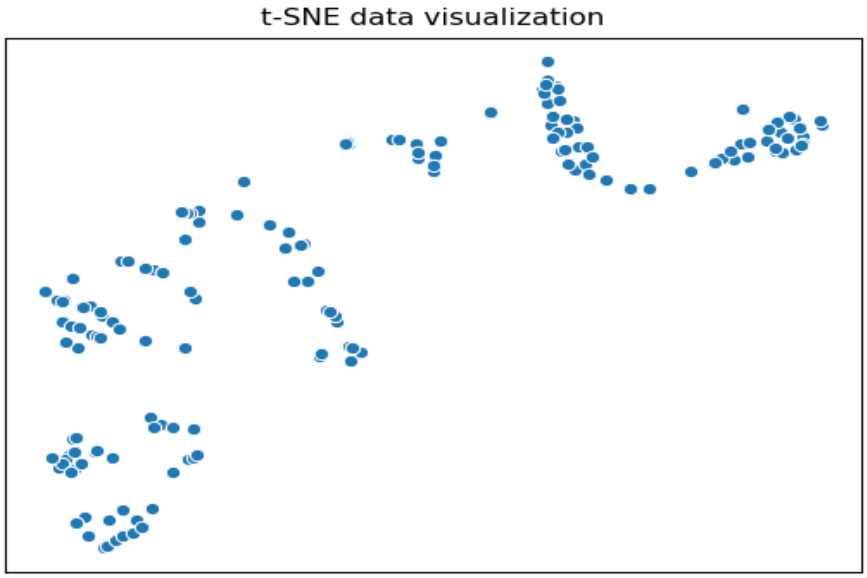


Figure 4.7 t-SNE data visualization

4.2 Preprocessing data

Once the data are collected is important to scale them to avoid overfitting or other problems. Overfitting, or high variance, is when an algorithm, which it is used to fit a training set, can have a high accuracy for the train set but a lower one for the predictions. In other words, overfitting occurs when the algorithm may fit the training set very well but fail to generalize to new examples.
In Scikit Learn there are many possibilities to scale the data but in this research the following are used:

- StandardScaler; this preprocessing ensures that for each feature the mean is 0 and the variance is 1, bringing all the features to the same magnitude. In this way it is possible to avoid odd data points, called outliers, that could create problem for the accuracy.
- MinMaxScaler; this procedure shifts the data such that all features are exactly between 0 and 1. For a two-dimensional dataset this means all the data is contained within the maximum and the minimum value of the feature.

CHAPTER 5
DATA TESTING AND RESULTS

The chapter is dedicated to the comparison of all the results obtained, in order to show which parameter are used to evaluate the accuracy and the reliability of the prediction.

5.1.1 Data tuning

For each different algorithm, several parameters or coefficients are needed to tune in order to obtain the best result in terms of R-squared and MARD.
In the next paragraphs the parameter necessary to find the best fit regression are described. For further information it is recommended to consult the documentation of Scikit-learn since in the concerned research only the parameters and the options modified are described but, for each algorithm, there are more coefficients to tune in general.

5.1.1.1 KNR

For the KNR algorithm the parameter to tune is just one, that is “n_neighbors”. This represents the number of neighbors taken into consideration to evaluate the predictions for the X test. A value of 5 it is used to obtain the best results.

5.1.1.2 Linear Models

For all the 3 linear models (Ridge, Elastic Net and Lasso) the parameter to tune is one, that is alpha. It gives an idea of how many samples are not considered by the

model to avoid overfitting. Since the dataset is not huge the values of alpha are quite small.

Table 5-1 Alpha values for linear models regressor

Algorithm	Alpha
Ridge	0.1
Elastic Net	0.01
Lasso	0.01

5.1.1.3 Decision Tree

The coefficients used for Decision Tree are 3:

- Min_samples_split: gives the minimum number of samples required to split an internal node and it is set to 25.
- Min_samples_leaf: is the minimum number of samples required to be at a leaf node. A value of 10 it is used, it means that a split point at any depth will only be considered if it leaves at least 10 training samples in each of the left and right branches.
- Presort: setting this option to 'True' the data are presorted to speed up the finding of best splits in fitting. This is widely used for small dataset.

5.1.1.4 Random Forest

For the Random Forest algorithm, the parameters tuned are 3:

- Max_depth: indicates the maximum depth of the tree. It is set to 10 to avoid overfitting.
- N_estimators: gives the number of trees in the forest. A value of 20 it is used.
- Min_sample_split: has the same meaning as in the decision tree algorithm and it is set to 40.

5.1.1.5 SVR

The options tuned for the SVR are 3. C and epsilon, respectively 0.1 and 0.001, are parameters to produce no penalty associated in training loss function for points that are predicted within a certain distance (evaluated using C and epsilon) from the

actual value. Finally, the Kernel type ‘rbf’, which stands for Radial Basis Function, is used.

5.1.1.6 MLP Regressor

CNN required a lot of parameters to tune due to their nature of multilayer connective framework. In the concerned research the parameters set are 7:

- Alpha = 0.1; is the regularization parameter
- Batch_size = 10; is the number of samples included in the minibatches to train the model.
- Hidden_layer_sizes = 1000; is the number of neurons in the *i*-th hidden layer
- Learning_rate = ‘constant’;
- Learning_rate_init = 0.001;
- Activation = ‘relu’; defines the activation function for the hidden layer. ‘relu’ stands for Rectified Linear Unit Function.
- Solver = ‘adam’; defines the solver for weight optimization. ‘adam’ refers to a stochastic gradient-based optimizer proposed by Kingma, Diederik and Jimmy Ba.

5.2 Data testing

Once the data are preprocessed the next step is to tune the parameter for each algorithm in order to obtain the best accuracy for the task. But how is measured the accuracy?

For regression problems doesn’t exist a unique parameter to evaluate quality of our prediction, therefore in the concerned research are used the two most used parameters: R-squared and Mean Absolute Error (MARD).

5.2.1 R-squared

The R-squared value, also known as coefficient of correlation, provides a measure of how well future samples are likely to be predicted by the model valuating how much the scatter points are distant from the regression fit line calculated by the algorithm.

Best possible score is 1.0 and it can be negative (that means, for instance that the model is predicting descending values while the true data are growing). Furthermore, a constant model that always predicts the expected value of y, disregarding the input features, would get a R-squared score of 0.

The statistic definition of the coefficient of correlation is the following:

If y_i is the predicted value of the *i*-th sample and y_i is the corresponding true value, then the score R^2 estimated over $n_{samples}$ is defined as:

$$(0.1) \quad R^2(y, y) = 1 - \frac{\sum_{i=0}^{n_{samples}-1} (y_i - y_i)^2}{\sum_{i=0}^{n_{samples}-1} (y_i - \bar{y})^2}$$

Where $\bar{y} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} y_i$ (0.2).

5.2.2 Mean Absolute Relative Distance (MARD)

MARD is the average vertical distance between each point and the regression line. MARD is also the average horizontal distance between each point and the best fit line.

Therefore, lower is the value of MARD and better are the predictions.

In statistic the definition of MARD is the following:

If y_i is the predicted value of the *i*-th sample, and y_i is the corresponding true value, then the MARD estimated over $n_{samples}$ is defined as:

$$(0.3)\ MARD(y, y) = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} \frac{\left| y_i - y_i \right|}{y_i}$$

5.3 Results

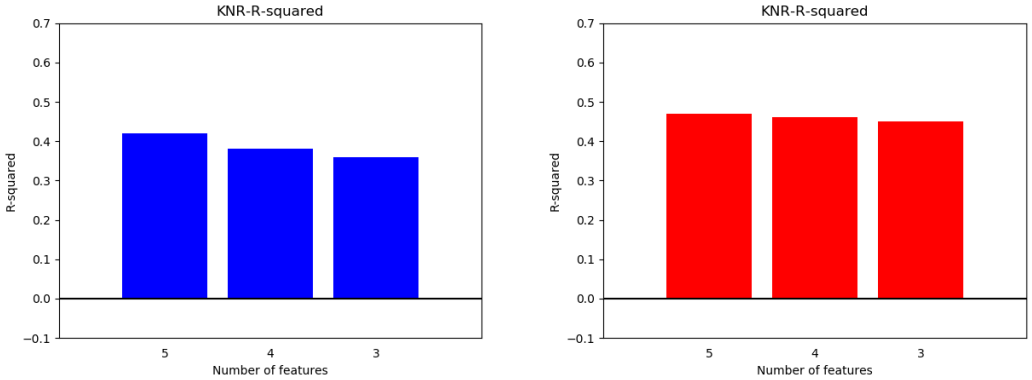
In the next paragraphs are shown the R-squared and MARD values for each algorithm. In the charts, shown for each algorithm, 3 colors are used, where;

- Blue bars refer to the score with a test size data about 15% of the dataset;
- Red bars refer to the score with a test size data about 20% of the dataset
- Green bars refer to the score with a test size data about 33% of the dataset

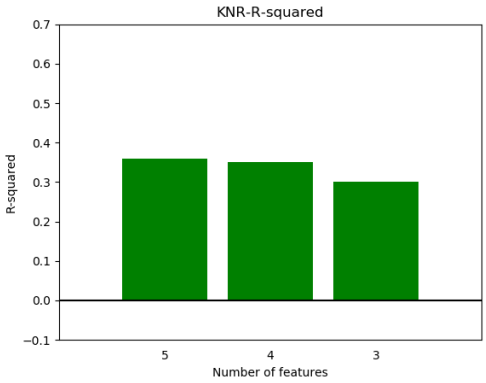
These quantities are the most common when an algorithm is trained.
Furthermore, each algorithm is run with a different number of features. That because the number of samples are different if we consider this 3 groups of multiple features. Indeed, as shown in Table 2-1, if we consider all the features, the dataset it is composed by 43 features; if the "year of construction" is ignored, the dataset is formed by 170 samples and if also the "distance from epicenter" is ignored, all the dataset and his 198 samples are used.

5.3.1 KNR

Using the parameters described in 5.1.1.1, the KNR algorithm gives the following R-squared scores, shown in Figure 5.1.



(a) (b)

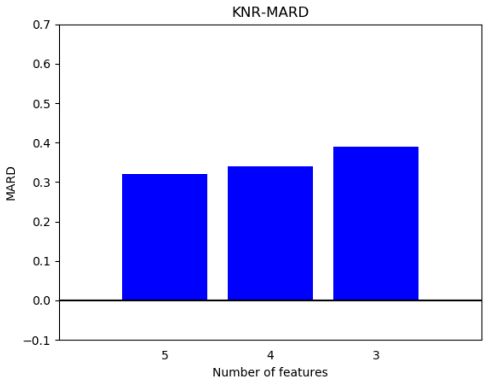


(c)

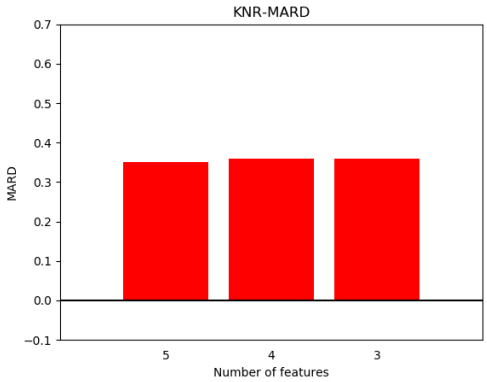
Figure 5.1 R-squared score for testing set about (a)15% (b)20% (c) 33%

The best score of 0.32 is obtained for the case (a), considering 3 features.

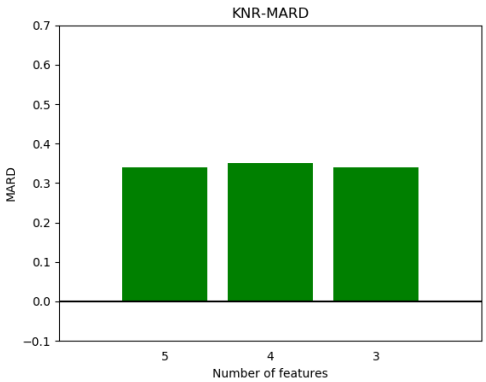
In Figure 5.2 are shown the MARD score for each case. The best score is reached for the case (b) with 3 features.



(a)



(b)

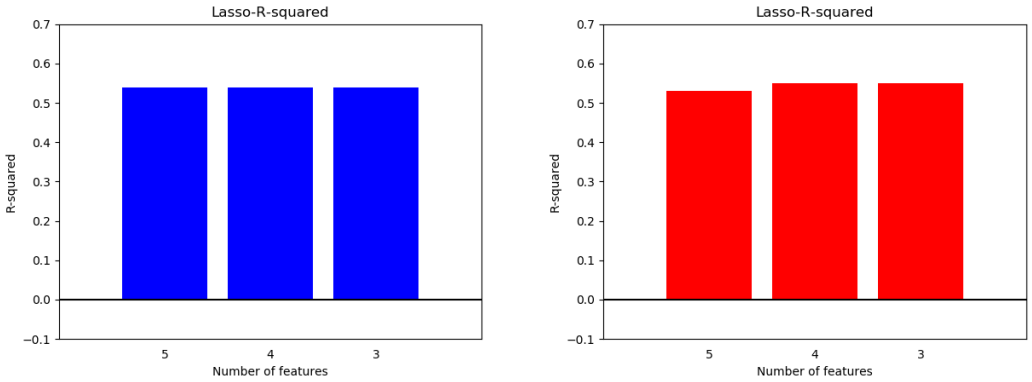


(c)

Figure 5.2 MARD score for testing set about (a)15% (b)20% (c) 33%

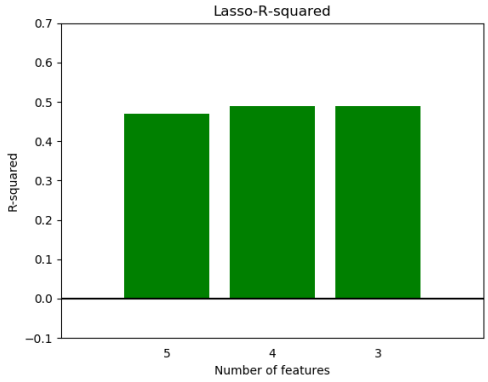
5.3.2 Linear models

Linear models present, except for very few percentages of difference, the same scores both in terms of R-squared and MARD. In Figure 5.3, Figure 5.4 and Figure 5.5 are shown the R-squared scores for Lasso, Elastic Net and Ridge algorithms.



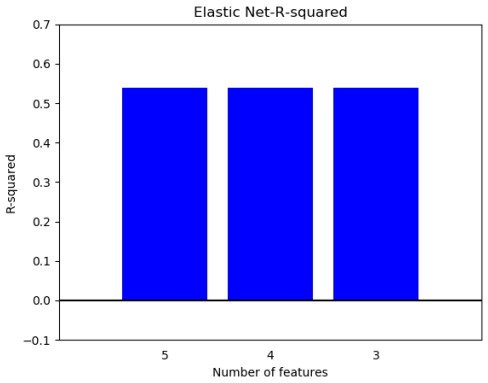
(a)

(b)

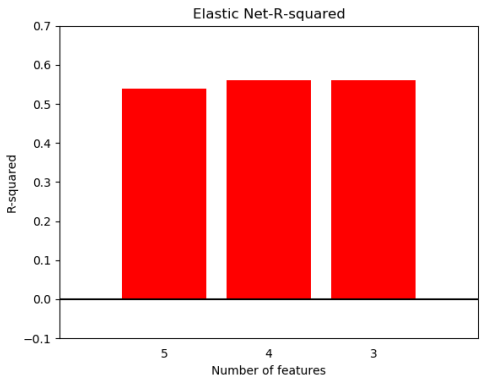


(c)

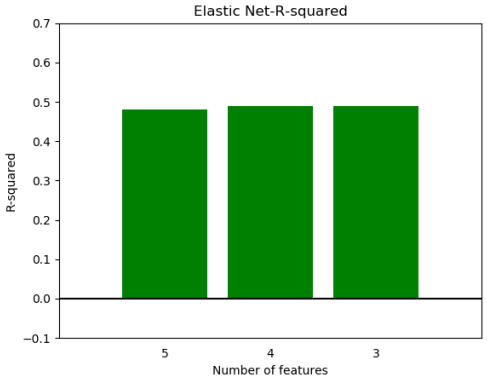
Figure 5.3 Lasso-R-squared score for testing set about (a)15% (b)20% (c) 33%



(a)

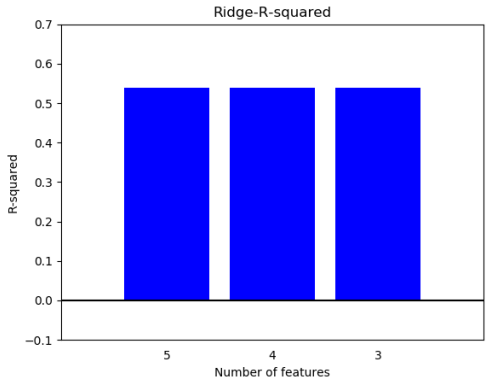


(b)

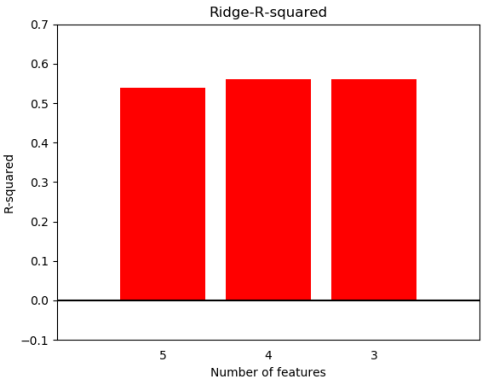


(c)

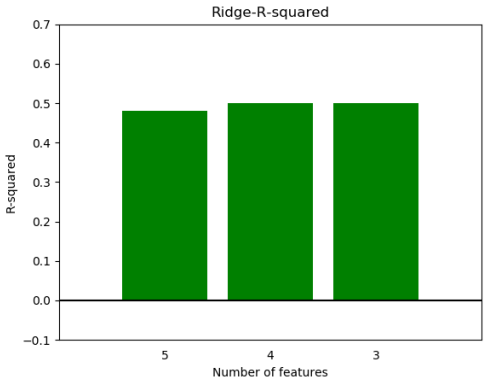
Figure 5.4 Elastic Net-R-squared score for testing set about (a)15% (b)20% (c) 33%



(a)



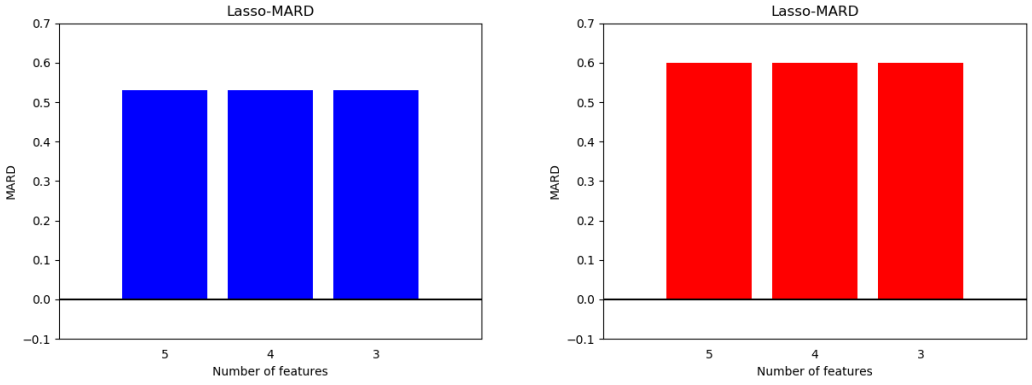
(b)



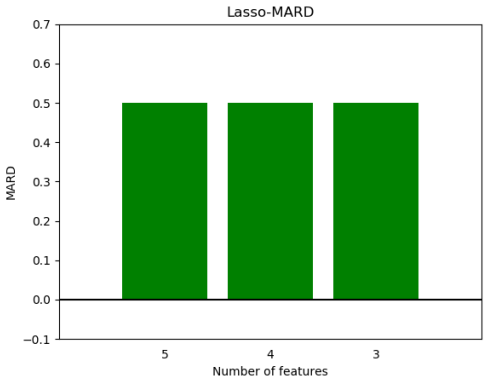
(c)

Figure 5.5 Ridge-R-squared score for testing set about (a)15% (b)20% (c) 33%

Linear models do not give good results in terms of R-squared and also the MARD scores are worse as it is possible to see in Figure 5.6, Figure 5.7, Figure 5.8.



(a) (b)



(c)

Figure 5.6 Lasso-MARD score for testing set about (a)15% (b)20% (c) 33%

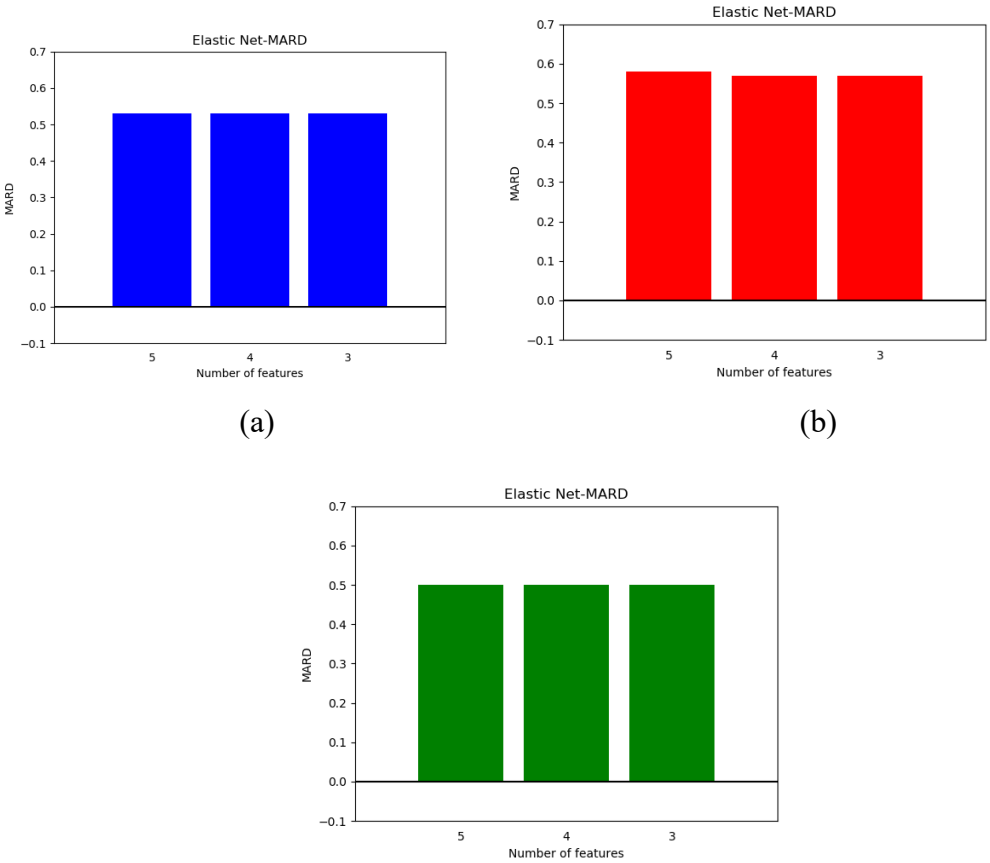
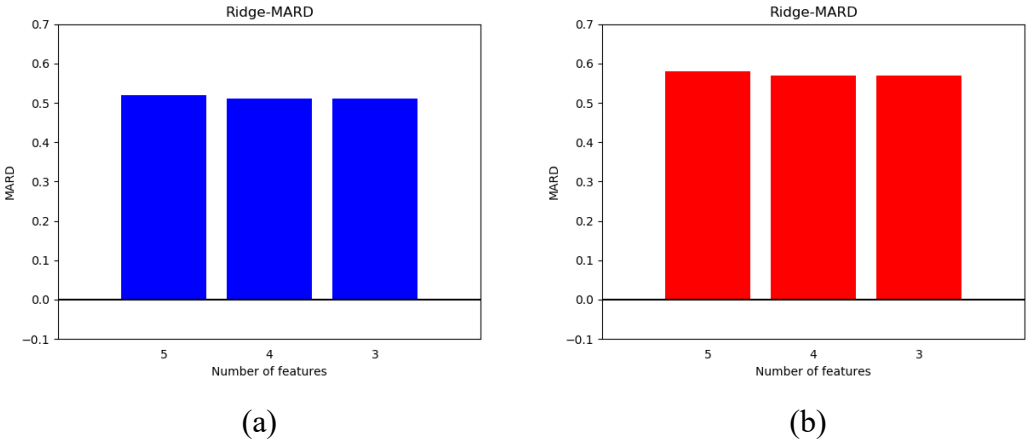
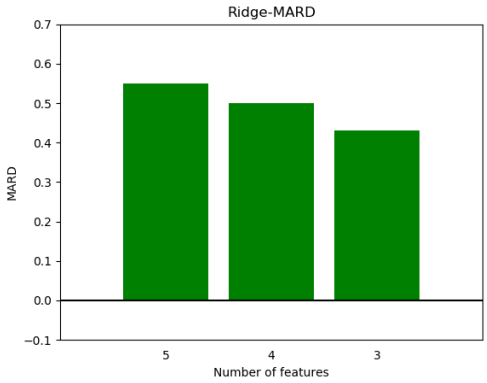


Figure 5.7 Elastic Net-MARD score for testing set about (a)15% (b)20% (c) 33%



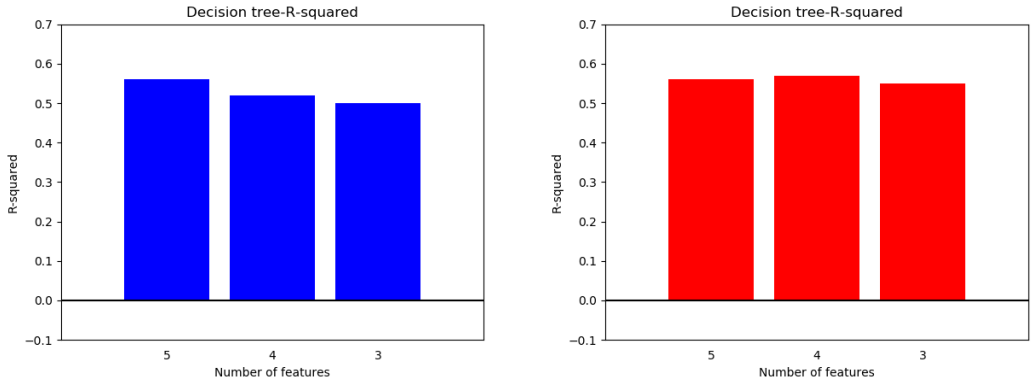


(c)

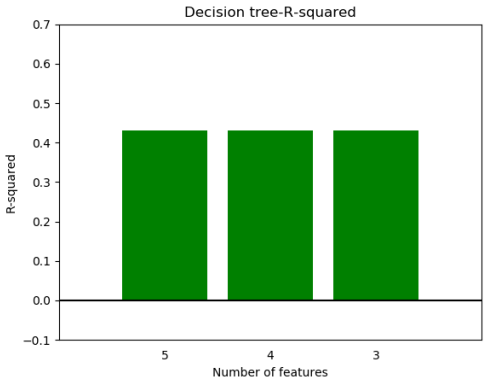
Figure 5.8 Ridge-MARD score for testing set about (a)15% (b)20% (c) 33%

5.3.3 Decision Tree

With decision tree algorithm the scores are better as it is shown in Figure 5.9.



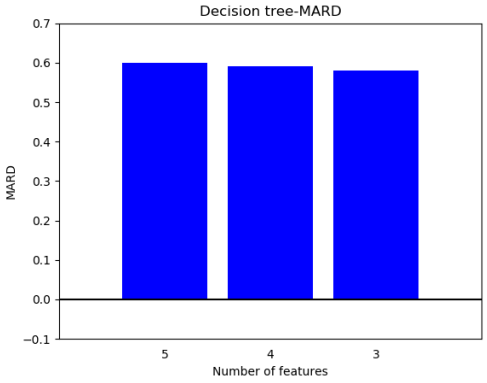
(a) (b)



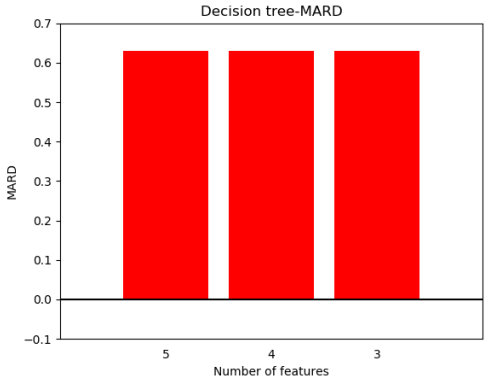
(c)

Figure 5.9 Decision tree-R-squared score for testing set about (a)15% (b)20% (c) 33%

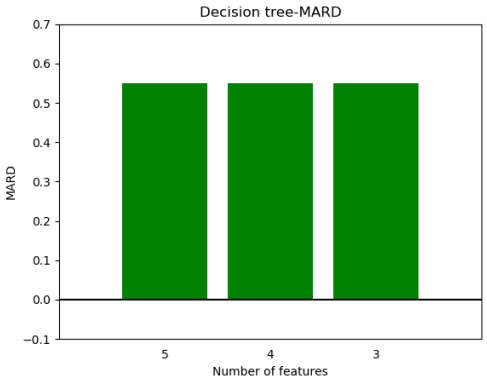
The best score is in the case (a) with all the features. The same case gives also the best result in terms of MARD with a value of 0.23 as shown in Figure 5.10.



(a)



(b)



(c)

Figure 5.10 Decision tree-MARD score for testing set about (a)15% (b)20% (c) 33%

5.3.4 Random forest

Random Forest gives the best scores both in terms of R-squared and MARD. Indeed, the R-squared score reach a total of 0.52 while the MARD score is 0.22. In the following Figure 5.11 and Figure 5.12 are shown the overall scores for each case.

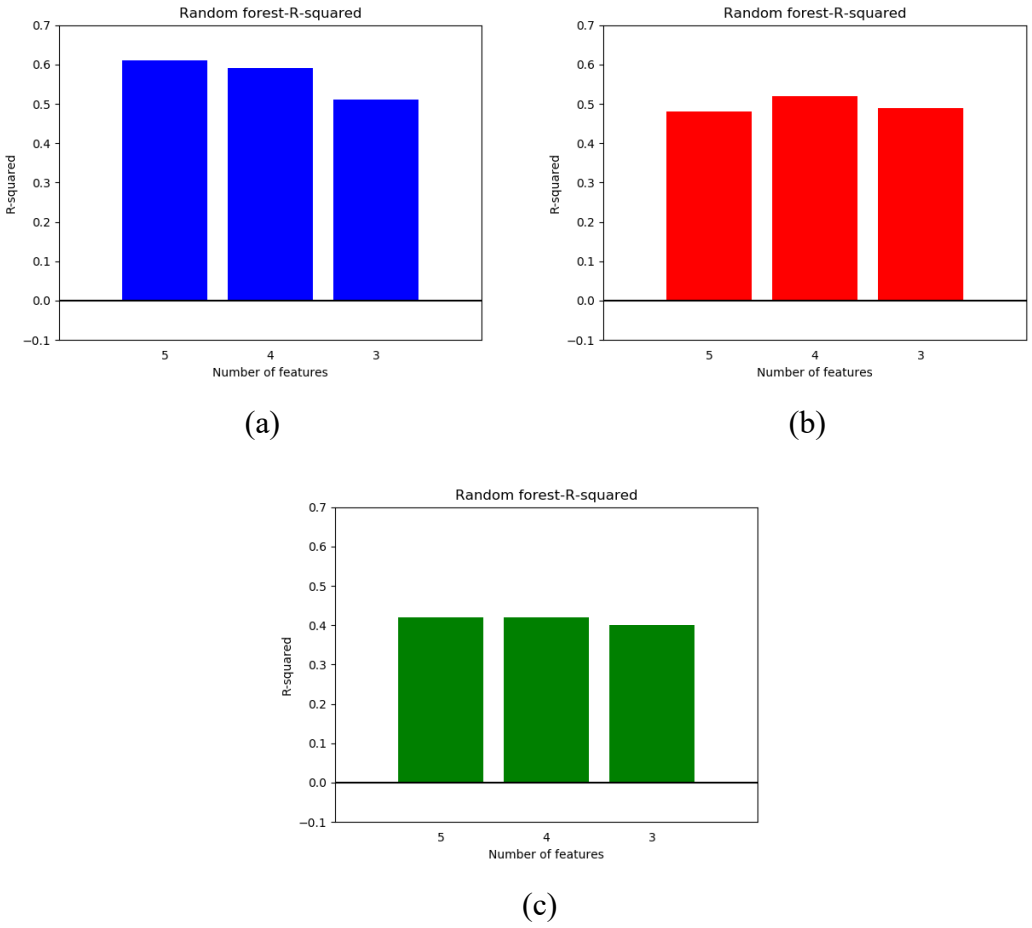


Figure 5.11 Random forest-R-squared score for testing set about (a)15% (b)20% (c) 33%

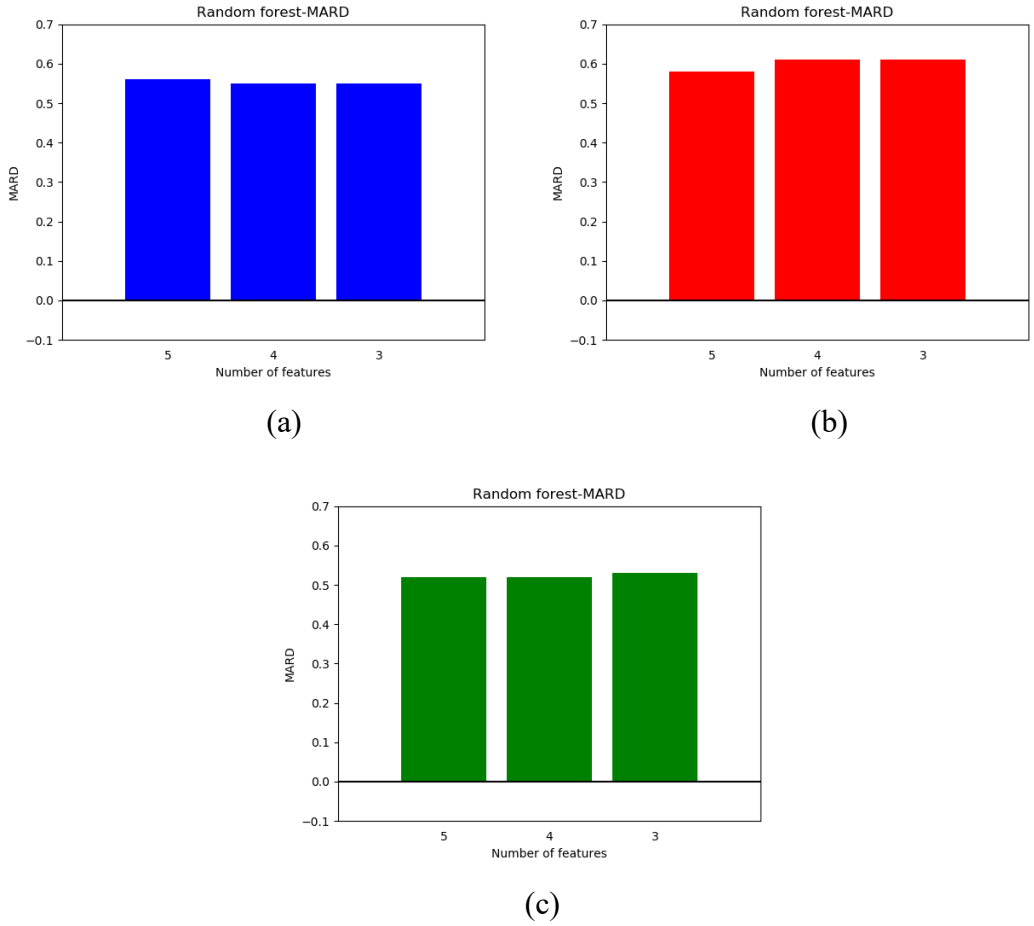
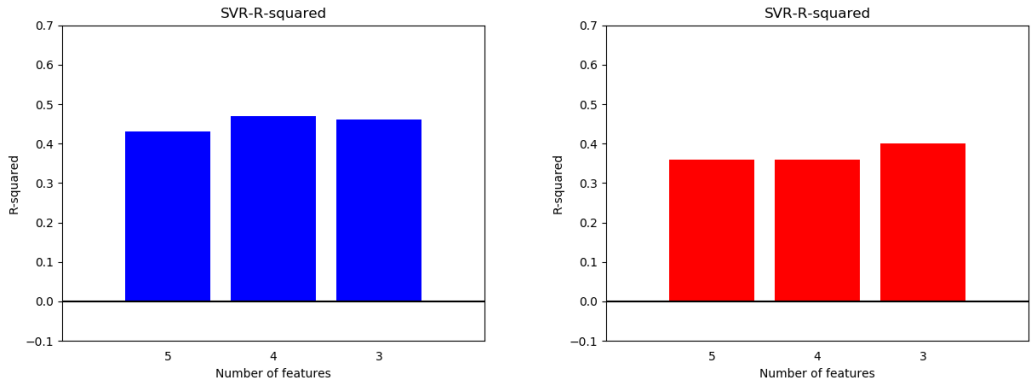


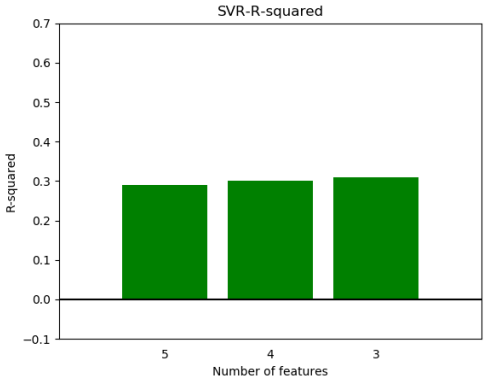
Figure 5.12 Random forest-MARD score for testing set about (a)15% (b)20% (c) 33%

5.3.5 SVR

In the following graphs are reported the results for the SVR algorithm. In Figure 5.13 are shown the R-squared scores.



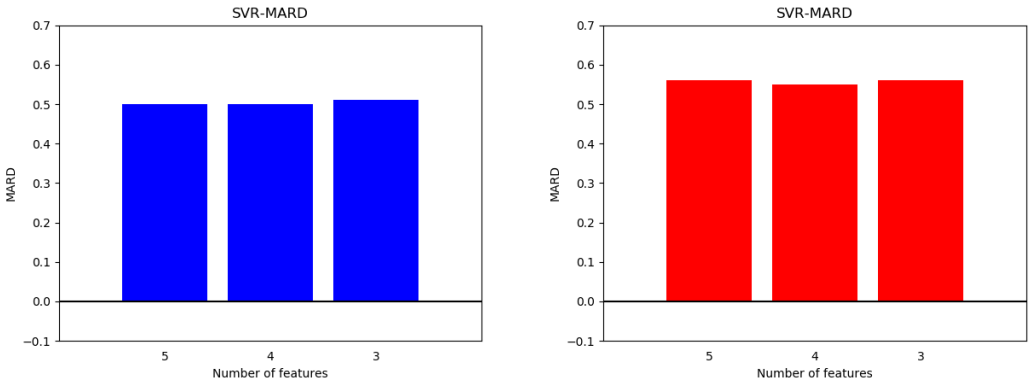
(a) (b)



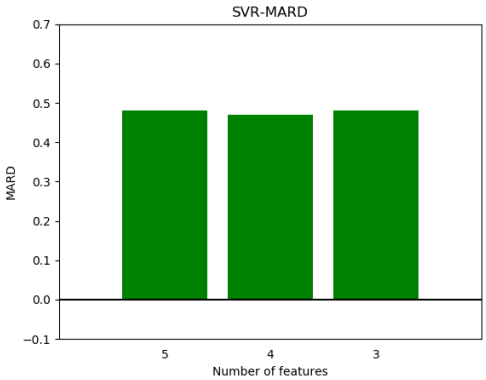
(c)

Figure 5.13 SVR-R-squared score for testing set about (a)15% (b)20% (c) 33%

In Figure 5.14 instead, are shown the MARD scores.



(a) (b)



(c)

Figure 5.14 SVR-MARD score for testing set about (a)15% (b)20% (c) 33%

5.3.6 MLP regressor

The last algorithm used is the MLP regressor that gives good results. For larger dataset this algorithm could give improved results.

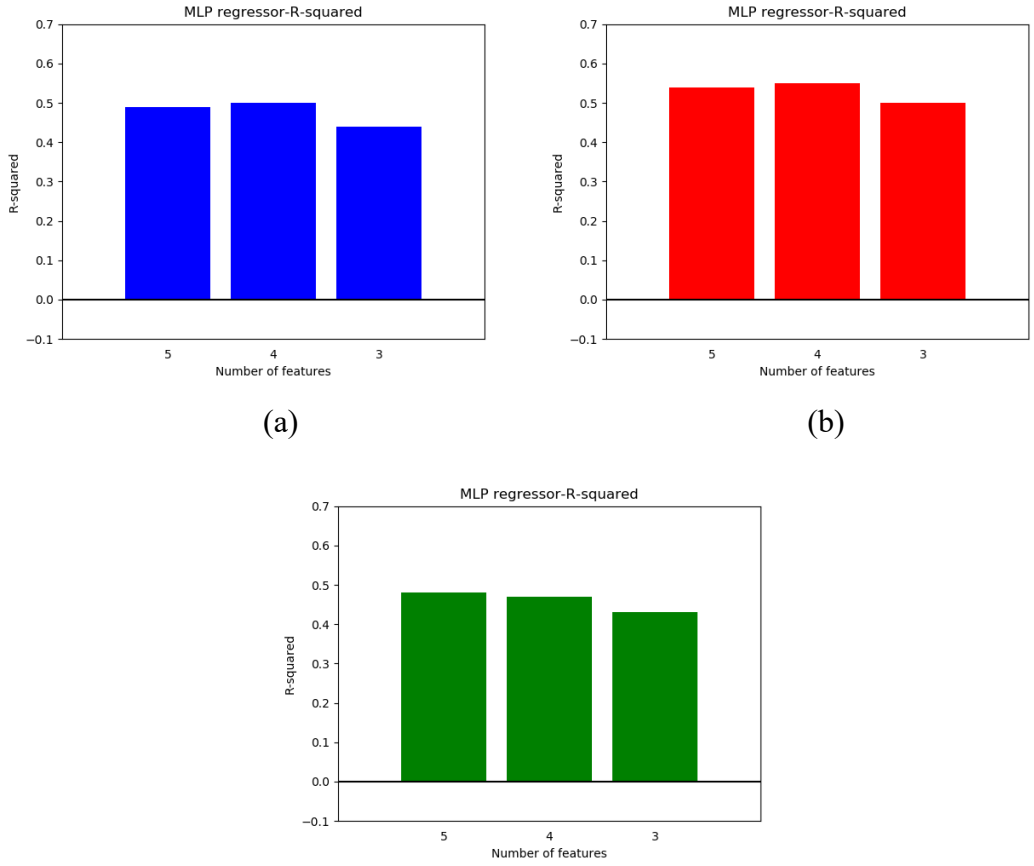
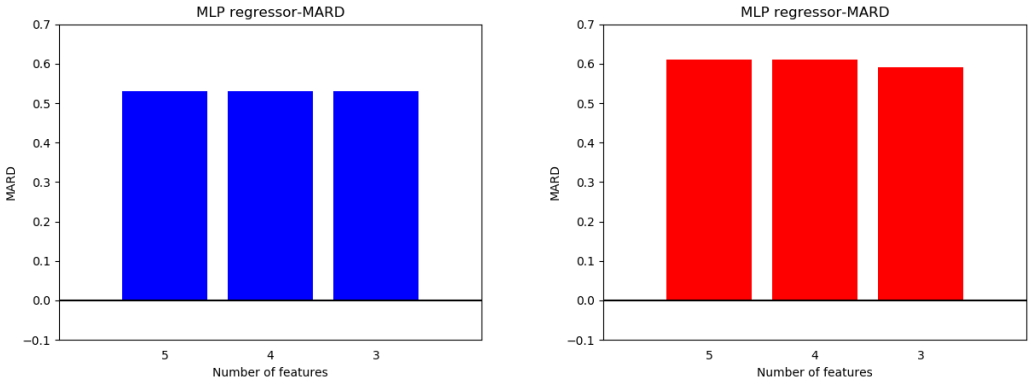


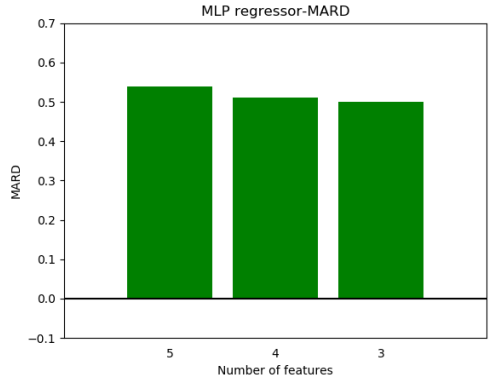
Figure 5.15 MLP regressor-R-squared score for testing set about (a)15% (b)20% (c) 33%

In Figure 5.15 it is possible to see R-squared scores, while in Figure 5.16 are reported the MARD values.



(a)

(b)



(c)

Figure 5.16 MLP regressor-MARD score for testing set about (a)15% (b)20% (c) 33%

5.3.7 Scores report

In Table 5-2 are reported all the values in numerical term for 3 different test set (15%, 20% and 33%) and for 3 different groups of features.

Table 5-2 R-squared and MARD scores

Algorithm		Number of features								
		5			4			3		
		15%	**20%**	**33%**	**15%**	**20%**	**33%**	**15%**	**20%**	**33%**
KNR	R^2	0.42	0.47	0.36	0.38	0.46	0.35	0.36	0.45	0.30
	MARD	**0.32**	0.35	0.34	0.34	0.36	0.35	0.29	0.36	0.34
Lasso	R^2	0.54	0.53	0.47	0.54	0.55	0.49	0.54	0.55	0.49
	MARD	0.53	0.60	0.50	0.53	0.60	0.50	0.53	0.60	0.50
Elastic Net	R^2	0.54	0.54	0.48	0.54	0.56	0.49	0.54	0.56	0.50
	MARD	0.53	0.60	0.50	0.53	0.60	0.50	0.53	0.60	0.50
Ridge	R^2	0.54	0.54	0.48	0.54	0.56	0.49	0.54	0.56	0.49
	MARD	0.52	0.58	0.55	0.51	0.57	0.50	0.51	0.57	0.43
Random Forest	R^2	**0.61**	0.48	0.42	0.59	0.52	0.42	0.51	0.49	0.40
	MARD	0.56	0.58	0.52	0.55	0.61	0.52	0.55	0.61	0.53
Decision Tree	R^2	0.56	0.56	0.43	0.52	0.57	0.43	0.50	0.55	0.43
	MARD	0.60	0.63	0.55	0.59	0.63	0.55	0.58	0.63	0.55
SVR	R^2	0.43	0.36	0.29	0.47	0.36	0.30	0.46	0.40	0.31
	MARD	0.53	0.56	0.48	0.50	0.55	0.47	0.51	0.56	0.48
MLP Regressor	R^2	0.49	0.54	0.48	0.50	0.55	0.47	0.44	0.50	0.43
	MARD	0.53	0.61	0.54	0.53	0.61	0.51	0.53	0.59	0.50

CHAPTER 6
PHI CHALLENGE AND IMAGE RECOGNITION

6.1 Introduction

Another relevant aspect of ML that is spread out in the last years (from 2011) is Convolutional Neural Network (CNN). This topic has assumed such importance that it has earned its own name: Deep Learning. The models and the algorithms included in this category are improved every year and a big amount of challenge and competition comes out in different fields: from entertainment (company such as Netflix, Spotify made their own competition) to Financial online security.

One of the most popular field where Deep Learning is widely used is for the Image Recognition. With this term it is indicated all the methods that allow computer CPU or, above all, GPU to classify pictures in different categories, which implies to "understand" what is represented in the pictures relying on the categories.

There have been several recent image-based recognition competitions (such as the PASCAL VOC, ImageNet, and COCO challenges) based on natural objects and scenes.

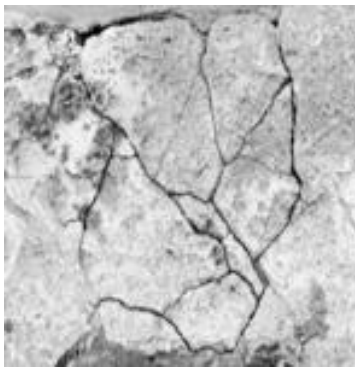
On 18th January 2018 the Pacific Earthquake Engineering Research (PEER) center announced the PEER Hub ImageNet (PHI) Challenge that provided the use of

Computer Science and Deep Learning techniques to organize the first image-based structural damage recognition competition. As it is possible to read in the main online page of the competition: “In the PHI Challenge, PEER will provide a large image dataset which is relevant to the field of structural engineering, and will design several detection tasks, which will contribute to the establishment of automated vision-based structural health monitoring. The goal of the PHI challenge is to evaluate algorithms for structural image classification using a large-scale dataset based on service conditions and past reconnaissance efforts and laboratory experiments for conditions of extreme events. The state-of-the-art algorithms to be tested in the PHI challenge are expected to enhance the accuracy and the generalization of vision-based approaches. These approaches will aim towards the construction of a big structural image dataset to solve societal-scale problems of structural health monitoring and assessment of the built environment.”
The concerned study belongs to the classification problems (different from the regression one, which we treated in the other paragraph). Even if it is still a supervised learning problem, classification problem is different from regression because now the parameter that is tried to forecast is not continuous but relies on different categories that are individuated before the training phase.

6.2 PHI Challenge tasks

In this competition, all training images were labeled by Pacific Earthquake Engineering Center, and each task has at most 4 categories. All images are assumed to be labeled well without any outlier in any category. All images have been resized to 224 x 224 x 3, which means its width, height, and channel (RGB).
The tasks have different weights for the overall score based on difficulty factors. The tasks are:
Easy:

- **Task 1**: Scene classification, 3 classes (pixel/object/structural levels);



Pixel Level

Object Level

Structural Level

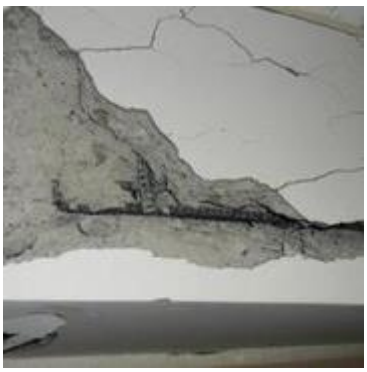
- **Task 2**: Damage check, 2 classes (yes/no);

Undamaged

Damaged

- **Task 3**: Spalling condition, 2 classes (yes/no);

No spalling



Spalling

- **Task 4**: Material type, 2 classes(steel/others);

Steel

Others

Medium:

- **Task 5**: Collapse check, 3 classes (no/partial collapse/collapse);

No collapse

Partial collapse

Collapse

- **Task 6**: Component type, 4 classes (beam/column/wall/else);

Beam

Column

Wall

Else (i.e.joint)

- **Task 7**: Damage level, 4 classes (no/minor/moderate/heavy damage);




No damage | Minor damage | Moderate damage | Heavy damage

Hard:

- **Task 8**: Damage type, 4 classes (no/flexural/shear/combined);




No damage | Flexural damage | Shear damage | Combined damage

In our training process,80% of images will be the training set, the rest 20% will be separated as the validation and test set equally (Table 6-1).

Table 6-1 Summary of Images in the training process.

	Task1	**Task2**	**Task3**	**Task4**	**Task5**	**Task6**	**Task7**	**Task8**
Training Set	13,939	4,730	3,294	3,469	412	2,104	2,105	2,105
ValidationSet	1,742	591	329	433	51	263	263	263

Test Set	1,742	591	329	433	51	263	263	263
Kaggle Test	4,356	1,479	824	1,085	129	658	658	658

After the algorithm is trained, a database of unlabeled images is provided so that predications of labels and annotation can be generated.

All the competition takes place in the Kaggle platform, which is defined in Wikipedia as "an online community of data scientists and machine learners, owned by Google, Inc. Kaggle allows users to find and publish data sets, explore and build models in a web-based data-science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges."

Through this platform it is possible to update in real time the prediction of the test and see the accuracy of the model.

6.3 Deep Learning models and Transfer Learning

The general definition of Transfer Learning (TL) is written in [15] (see the references):

"Given a source domain and its learning task, a target domain and its target task, the objective of TL is to help improve the prediction function in learning target task using the knowledge from source domain with source target."

In the concerned study, TL is practiced within the training process so that the lower level features (e.g., color, corner, and edge) can be used directly from pre-trained models (e.g., VGG16, VGG19, Inception, ResNet), which effectively reduce the training time with large-scale image set.

To obtain the baseline model, 4 state-of-the-art models have been chosen based on their performance in ImageNet Challenge, which includes VGG16, VGG19, Inception, and Xception. According to TL, lower convolutional bocks that have predictors (weights and biases) to capture common features such as edge, colour, and corner. Within the training process, those features do not need to be learned so that

the overall training time can be reduced. In order to skip the training for lower features, the number of layers that need to be frozen must be defined initialized so that those weights and biases will not be updated during the backward propagation process. The summary of frozen layers for different models in our training is shown in Table 6-2. The layer indicated in the table including convolutional layer, Rectified Linear Units (ReLU), max pooling layer, and fully connected layer.

Table 6-2 Summary of frozen layers

Model	VGG16	VGG19	Inception	ResNet50	Xception
Frozen Layers	25	20	197	160	54

6.4 Google Cloud Platform (GCP)

The dataset provided by PEER is formed by images with a dimension of 224x224 pixel. Applying this dimension for all the pictures gives an idea about the huge amount of work that is required for the algorithm to make a prediction. Therefore, treating this data needs a powerful machine that permits to run different CNN models in order to get the best accuracy in terms of predictions.

Using commercial computer, even most powerful, could give memory error working with such big database. For that reason, to run the models in very powerful virtual machines it is used Google Cloud Platform (GCP), which is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube.

In this way, it was possible use a computer with the v100 GPU having the following characteristics:

Table 6-3 Nvidia v100 GPU characteristics

Characteristic	Value

Transistors	21.1 billion
CUDA cores	5,120
Tensor cores	640
GPU clock speed	1200MHz base, 1455MHz boost
Memory capacity	12GB HBM2
Memory clock	850MHz
Memory interface	3,072-bit
Total memory bandwidth	652.8GBps
Texture units	320
Power	250W TDP via 1x 6-pin and 1x 8-pin power connectors
Ports	3x DisplayPort, 1x HDMI
Price	$3,000 on Nvidia.com

6.5 Training and results

For preprocessing, all input images were assigned as 224 x 224 x3 at the beginning. However, each model has its own default input size. For example, VGG16, VGG19, and ResNet require image size 224 x 224 x3. Both InceptionV3 and Xception require image size 299 x 299 x 3. Besides the modification of size, normalization is also implemented for all images. That is, the pixel must be divided by 255 so that its value is between 0 and 1. Based on the observation, the given images have some rotation and flip that can be confusing during the learning process. As a result, 10% of input images will be randomly rotated and the other 10% will be randomly flipped to make sure that those confusing example in the real test set can be captured.

The overall hyperparameters that utilized in our training process are shown in Table 6-4. The initial learning rate is fixed to all tasks but with different decay rate and epoch based on the size of images. The decay rate can make sure the learning process

will not go too fast to skip the global solution. The accuracy of each task is the result by voting the most predicting category among all models.

Table 6-4 Summary of the training and testing process.

	Task1	**Task2**	**Task3**	**Task4**	**Task5**	**Task6**	**Task7**	**Task8**
Mini-Batch	64	32	32	32	16	32	32	32
Initial Learning Rate	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
Learning Rate Decay	1e -03	1e -03	1e -03	1e -03	1e -03	1e -03	1e -03	1e -03
Epoch	10,000	6,000	5,000	5,000	2,000	3,000	3,000	3,000
Accuracy	0.907	0.791	0.83	0.967	0.625	0.684	0.608	0.559

CONCLUSIONS

The subject of this thesis was the development of a methodology to assess the extension of the debris caused by earthquake damage. The amount of rubble is estimated from pictures taken in the field and several modern Machine Learning algorithms are evaluated based on their prediction accuracy in terms of R-squared and Mean Absolute Relative Distance (MARD). The work begins with a collection of 198 pictures (and related data) from 14 different earthquakes.

Using this dataset and 6 features, 8 different algorithms were trained and their performance in terms of predicting the extension of the debris was evaluated.

During the training process, some parameters, different for each algorithm, were altered and tested in order to achieve the best score.

Looking at the results, the main features of the present research can be summarized as follows:

- KNR algorithm gives the best MARD score (0.32) but the low value of R-squared (0.42) means that with more data the algorithm, probably, will generalize too much.
- Linear models furnish similar results;
- Random forest gives the best R-squared but a high mean absolute relative deviation (MARD) (0.56). The predictive performance can be improved if more data is collected;
- MLP Regressor, which is a type of Convolutional Neural Network, does not achieve good results because of the size of the dataset that does not allow the network to exploit his power.
- The best score, both for R-squared and MARD, are reached considering all the features and a testing set of 15%. This is due to the medium-small

dimension of the dataset. If more data are collected, maybe, better results can be reached for bigger testing data (which could give more consistence values).

In conclusion Random forest and KNR are the best choices for predicting the debris-extension and the performances suggest that there is some potential for using machine learning methodologies in this field. It is appropriate to underline that collecting more pictures and data from other Earthquakes in future could significantly improve the results.

In the last part of the concerned thesis, it is described the PHI Challenge, organized by PEER.

In this competition, state-of-the-art CNN and Transfer Learning are implemented to classify the structural image into multi-category which includes the damage status before and after the disaster as wells the materials and structural components. Due to the pre-trained models that are available in Keras/TensorFlow, the training process can be reduced to focus on the higher-level features for a specific category. GCP also provides a friendly environment for the beginner of Deep Learning who may not have enough computing power to fulfill large-scale image classification, which is also utilized in this competition.

The overall performance and the process give a good chance for civil engineers to learn the new technology and utilize it to improve current research topics.

APPENDIX

8.1 Dataset

Legend:

1. ID = Identification number of the picture
2. Earthquake = Earthquake location
3. Mat. = Material:
    - 0 = Masonry
    - 1 =Reinforced Concrete
4. Stories
5. Year
6. Magnitude
7. Distance = Distance from epicenter in kilometers
8. Log(Height) = logarithm of the buildings' heights
9. Log(EOD Norm.) = Logarithmic value of Extension Of Debris normalized with the heights of the buildings
10. d = Extent of debris calculated following the procedure in the paragraph 2.3.1

Note: A missing data is replaced with the value -1.

ID	Earthquake	Mat	Stories	Year	Magnitude	Distance [km]	Log(Height) [m]	Log(EOD Norm.)	d [m]
id.001	Mexico City	1	8	-1	8.1	60	0.826075	-0.4973	1.9091
id.002	Mexico City	0	12	-1	8.1	51	0.812913	0.093912	7.4483
id.003	Mexico City	1	7	-1	8.1	52	0.778151	-0.47716	2
id.004	Mexico City	1	6	-1	8.1	48	1.30103	-0.80272	2.8356
id.006	Mexico City	1	2	-1	8.1	54	0.90309	-0.41016	3.5
id.007	Northwestern Armenia	0	3	-1	6.9	47	0.944483	-0.66655	1.9397
id.009	Northwestern Armenia	1	5	-1	6.9	168	0.778151	-0.37069	2.5556

id.010	Northwestern Armenia	1	5	-1	6.9	167	1.255273	-0.68909	3.6833
id.011	Northwestern Armenia	0	5	-1	6.9	173	0.939519	-0.57955	2.3697
id.012	Northwestern Armenia	0	5	-1	6.9	170	0.90309	-0.13829	6.5455
id.013	Loma Prieta	0	3	-1	7.1	175	1.041393	-0.93033	1.0568
id.014	Loma Prieta	0	2	-1	7.1	-1	1	-0.8617	1.2371
id.015	Loma Prieta	0	3	-1	7.1	17	1.255273	-1.51428	0.55
id.016	Loma Prieta	0	4	-1	7.1	16	0.954243	-0.18092	5.9333
id.017	Loma Prieta	0	4	-1	7.1	17	1.255273	-0.80715	2.8056
id.019	Northern Iran	0	6	-1	7.7	14	0.954243	-0.96417	0.9778
id.020	Northridge	0	2	-1	6.8	13	1.20412	-0.95117	1.678
id.021	Northridge	0	2	-1	6.8	15	0.845098	-0.51727	1.8235
id.022	Central Italy	0	2	-1	6.1	36	0.845098	-0.55533	1.6702
id.023	Central Italy	0	2	-1	6.1	35	0.778151	-0.63097	1.4033
id.024	Emilia Romagna	0	3	1935	6.1	-1	0.982271	-0.22863	5.3165
id.025	Emilia Romagna	0	3	1935	6.1	-1	1	-0.25524	5
id.026	Emilia Romagna	0	6	1935	6.1	-1	1.255273	-0.08197	14.9032
id.028	Emilia Romagna	0	2	-1	6.1	34	0.819544	-0.49785	1.9068
id.029	Emilia Romagna	0	2	-1	6.1	33	0.799341	-0.83062	0.886
id.030	Emilia Romagna	0	3	-1	6.1	37	0.954243	-0.86012	1.2424
id.032	Emilia Romagna	0	3	1864	6.1	48	0.954243	-0.83893	1.3043
id.033	Emilia Romagna	0	3	1864	6.1	48	0.90309	-1.07935	0.75
id.035	Emilia Romagna	0	5	1744	6.1	57	1.176091	-0.92812	1.77
id.036	Emilia Romagna	0	3	-1	6.1	35	1.012837	-0.59963	2.2629
id.039	Emilia Romagna	0	5	1401	6.1	-1	1.176091	-0.7597	2.6086
id.040	Emilia Romagna	0	3	-1	6.1	69	0.944483	-0.74256	1.6279
id.043	Emilia Romagna	1	2	-1	6.1	198	0.875061	-0.80052	0.95
id.044	Emilia Romagna	1	3	-1	6.1	200	0.954243	-1	0.9
id.045	South Napa Valley	0	3	1940	6	28	0.968483	-0.54668	2.5556
id.046	South Napa Valley	0	3	1940	6	29	0.968483	-0.5373	2.6122
id.049	South Napa Valley	0	3	1905	6	30	9	0.121727	1.095541
id.050	South Napa Valley	0	2	1940	6	31	0.939519	-0.20363	5.6316
id.051	South Napa Valley	0	2	1940	6	33	0.778151	-0.65995	1.3125
id.053	South Napa Valley	0	1	1940	6	27	0.826075	-0.285	3.1125
id.054	South Napa Valley	0	1	1940	6	30	0.69897	-0.18329	3.2784
id.055	Southern Taiwan	0	2	-1	6.4	202	1.255273	-0.85356	2.5213
id.056	Southern Taiwan	0	2	-1	6.4	-1	0.973128	-0.58939	2.3163
id.057	Puebla	1	3	-1	7.1	-1	1.176091	-1.07314	1.2676
id.059	Puebla	0	2	-1	7.1	-1	1	-0.52476	2.6886
id.060	Puebla	1	2	-1	7.1	-1	0.755875	-0.1224	4.5263
id.061	Puebla	0	2	-1	7.1	-1	0.954243	-0.38143	3.7393
id.062	Puebla	0	2	-1	7.1	-1	0.954243	-0.39448	3.6289

id.064	Christchurch	0	2	-1	6.2	-1	1.255273	-0.51089	5.5514
id.068	Christchurch	0	3	-1	6.2	411	1.518514	-0.35018	14.7342
id.069	Christchurch	0	3	-1	6.2	403	0.845098	-0.27368	3.195
id.070	Christchurch	0	3	-1	6.2	-1	0.69897	-0.21254	3.6783
id.079	Christchurch	1	3	-1	6.2	160	0.934498	-0.57073	2.4187
id.081	Christchurch	1	2	-1	6.2	158	0.799341	-0.35566	2.6452
id.082	Christchurch	1	2	-1	6.2	162	0.778151	-0.00753	5.8966
id.084	Christchurch	0	2	-1	6.2	230	0.954243	-0.31292	4.3784
id.086	Christchurch	0	3	-1	6.2	229	1.255273	-0.70708	3.5342
id.087	Cephalonia	1	2	-1	6	232	0.778151	-1.03953	0.5479
id.088	Cephalonia	1	2	-1	6	228	0.954243	-1.04048	0.8203
id.089	Cephalonia	1	2	-1	6	233	1.255273	-1.4318	0.6667
id.090	Cephalonia	1	2	-1	6	230	1.255273	-1.39794	0.7204
id.091	Cephalonia	1	3	2007	6	-1	0.954243	-0.96019	0.9861
id.092	Cephalonia	0	1	-1	6	250	0.69897	-0.24558	2.8403
id.093	Central Italy	0	1	-1	6.2	244	1.255273	-0.8729	2.4118
id.094	Central Italy	0	3	-1	6.2	246	0.986772	-0.72955	1.6774
id.095	Central Italy	0	1	-1	6.2	247	0.863323	-0.28659	3.1011
id.096	Central Italy	0	1	1701	6.2	-1	0.973128	-0.56431	2.4545
id.097	Central Italy	1	2	-1	6.2	240	0.845098	-0.9751	0.6353
id.098	Central Italy	1	2	-1	6.2	238	1.176091	-0.88107	1.9728
id.100	Central Italy	0	4	-1	6.2	235	1.079181	-0.96738	1.2938
id.101	Central Italy	0	4	-1	6.2	233	0.778151	-0.32523	2.8377
id.102	Central Italy	0	3	-1	6.2	234	0.954243	-0.72102	1.7109
id.103	Central Italy	0	3	-1	6.2	240	1.623249	-1.45967	1.4583
id.104	Central Italy	0	3	-1	6.2	239	0.812913	-0.33106	2.7994
id.106	Central Italy	0	3	-1	6.2	235	0.954243	-0.81987	1.3622
id.107	Central Italy	0	4	-1	6.2	235	0.995635	-0.63714	2.0755
id.112	India	1	11	1995	7.7	-1	1.623249	-1.09313	3.3913
id.114	India	1	4	-1	6.9	160	1.380211	-1.22695	1.4237
id.116	India	1	2	-1	6.9	157	1.255273	-0.98548	1.8613
id.118	India	1	6	-1	6.9	222	1.518514	-1.60033	0.8296
id.120	India	1	6	-1	6.9	227	1.518514	-1.48545	1.0777
id.122	India	1	6	-1	6.9	225	1.053078	-0.95429	1
id.123	Mexico City	1	14	1962	8.1	48	1.079181	-0.07438	10.1111
id.129	Turkey	1	3	1999	6.4	20	0.832509	-1.15181	0.4227
id.133	Ecuador	1	2	-1	7.8	223	0.778151	-0.77806	1
id.134	Ecuador	1	2	-1	7.8	227	0.845098	-0.81588	0.9169
id.135	Ecuador	1	2	-1	7.8	225	0.954243	-0.77806	1.5
id.136	Ecuador	1	2	-1	7.8	224	1.041393	-1.10735	0.859
id.137	Ecuador	1	2	-1	7.8	60	0.832509	-0.82304	0.9016

id.138	Ecuador	1	2	-1	7.8	62	0.778151	-0.80024	0.9501
id.139	Ecuador	1	2	-1	7.8	58	0.954243	-1.04191	0.8175
id.140	Ecuador	1	2	-1	7.8	63	1	-0.96658	0.9717
id.141	Ecuador	1	3	-1	7.8	59	1.477121	-1.21183	1.8409
id.142	Ecuador	1	3	-1	7.8	61	0.929419	-0.69789	1.8045
id.144	Ecuador	1	3	-1	7.8	387	0.939519	-0.69702	1.8084
id.145	Ecuador	1	3	-1	7.8	407	0.792392	-0.52827	1.7778
id.146	Ecuador	1	3	-1	7.8	386	0.845098	-0.5583	1.6591
id.147	Ecuador	1	3	-1	7.8	390	0.954243	-0.70202	1.7872
id.148	Ecuador	1	3	-1	7.8	414	1.079181	-0.88339	1.5701
id.149	Ecuador	1	3	-1	7.8	392	0.954243	-0.99654	0.9074
id.150	Ecuador	1	3	-1	7.8	395	1.079181	-0.48945	3.8879
id.151	Ecuador	1	3	-1	7.8	400	1.477121	-0.87452	4.0055
id.152	Ecuador	1	5	-1	7.8	404	1.322219	-1.25806	1.1584
id.153	Ecuador	1	5	-1	7.8	3	0.954243	-0.92885	1.0601
id.154	Ecuador	1	6	-1	7.8	2	1.176091	-1.31426	0.7281
id.155	Ecuador	1	6	-1	7.8	12	1.176091	-1.23062	0.8814
id.158	Ecuador	1	3	-1	7.8	11	1	-0.81361	1.536
id.162	Ecuador	1	3	-1	7.8	9	1.079181	-0.84345	1.7203
id.163	Ecuador	1	3	-1	7.8	8	0.845098	-0.66254	1.3048
id.164	Ecuador	1	5	-1	7.8	7	1.556303	-1.41341	1.3896
id.165	Ecuador	1	5	-1	7.8	10	1.176091	-1.20482	0.9364
id.166	Ecuador	1	5	-1	7.8	10	1.380211	-1.56067	0.6606
id.167	Ecuador	1	3	-1	7.8	16	1.079181	-1.14691	0.8554
id.168	Ecuador	1	3	-1	7.8	14	0.954243	-0.74136	1.6327
id.169	Ecuador	1	3	-1	7.8	15	0.792392	-0.53417	1.7536
id.170	Ecuador	1	2	-1	7.8	17	0.778151	-0.914	0.7312
id.172	Ecuador	1	2	-1	7.8	147	1.041393	-1.11805	0.8385
id.173	Ecuador	1	2	-1	7.8	153	1.079181	-1.1831	0.7869
id.174	Ecuador	1	2	-1	7.8	98	0.653213	-0.78675	0.7354
id.175	Ecuador	1	3	-1	7.8	102	0.90309	-0.75995	1.5638
id.176	Ecuador	1	3	-1	7.8	100	1.021189	-0.73212	1.6677
id.177	Ecuador	1	3	-1	7.8	101	1	-0.73142	1.6704
id.184	Nepal	1	3	-1	7.8	65	0.929419	-0.32587	4.25
id.190	Mexico City	1	6	-1	5.7	126	1.556303	-0.91686	4.3608
id.191	Mexico City	1	6	-1	5.7	124	1.278754	-0.78755	2.9361
id.192	Mexico City	1	6	-1	5.7	123	1.176091	-0.99353	1.5219
id.193	Mexico City	1	6	-1	5.7	127	1	-0.83003	1.3311
id.194	Mexico City	1	6	-1	5.7	125	0.982271	-0.77989	1.4944
id.195	Central Italy	0	1	-1	6.2	-1	1.477121	-0.82016	4.5396
id.196	Central Italy	0	3	-1	6.2	-1	0.977724	-0.27597	4.7676

id.201	Central Italy	1	2	-1	6.2	63	0.963788	-0.47716	3
id.204	Central Italy	0	1	-1	6.2	67	0.653213	-0.1986	2.8486
id.205	Central Italy	1	3	-1	6.2	66	1.033424	-0.94962	1.0104
id.206	Central Italy	0	3	-1	6.2	65	0.939519	-0.84497	1.2861
id.209	Central Italy	0	3	-1	6.2	64	1.079181	-1.37986	0.5
id.211	Central Italy	0	3	-1	6.2	62	0.977724	-0.61618	2.1776
id.212	Central Italy	0	3	-1	6.2	70	0.838849	-0.67778	1.26
id.214	Southern Taiwan	0	8	1985	6.4	-1	1.623249	-1.18046	2.7707
id.220	Southern Taiwan	0	2	1975	6.4	44	1.255273	-0.66696	3.875
id.221	Southern Taiwan	0	2	1975	6.4	46	1.518514	-0.91757	3.9911
id.224	Southern Taiwan	1	4	1981	6.4	-1	0.819544	-0.72677	1.1255
id.226	Southern Taiwan	1	1	1982	6.4	22	1.477121	-0.88874	3.8773
id.227	Southern Taiwan	1	1	1982	6.4	21	0.954243	-0.99055	0.9194
id.228	Southern Taiwan	0	1	1971	6.4	50	0.838849	-0.64187	1.3685
id.232	Southern Taiwan	1	4	1984	6.4	15	1.079181	-1.57349	0.3207
id.234	Southern Taiwan	1	4	1984	6.4	24	1.518514	-1.57025	0.8881
id.235	Southern Taiwan	1	4	1984	6.4	24	1.380211	-1.40012	0.955
id.236	Southern Taiwan	1	4	1984	6.4	15	1.079181	-1.57349	0.3207
id.238	Southern Taiwan	1	3	1985	6.4	67	0.845098	-0.21439	3.6622
id.239	Southern Taiwan	1	4	1984	6.4	21	1.079181	-0.78068	1.9881
id.240	Southern Taiwan	1	1	1982	6.4	30	0.653213	-0.21424	3.6636
id.241	Southern Taiwan	1	1	1982	6.4	29	0.954243	-0.50626	2.8056
id.242	Southern Taiwan	1	1	1982	6.4	28	1.079181	-1.26841	0.6466
id.243	Southern Taiwan	1	1	1982	6.4	28	0.69897	-0.98047	0.6278
id.244	Southern Taiwan	1	1	1982	6.4	29	0.954243	-1.098	0.7186
id.248	Southern Taiwan	1	1	1982	6.4	31	0.778151	-0.969	0.6445
id.249	Southern Taiwan	1	1	1982	6.4	31	1.612784	-1.75203	0.7269
id.250	Southern Taiwan	1	1	1982	6.4	-1	0.832509	-0.67882	1.2571
id.251	Southern Taiwan	1	1	1982	6.4	-1	0.954243	-0.78383	1.4803
id.254	Southern Taiwan	1	1	1982	6.4	190	0.90309	-0.82681	1.3412
id.256	Southern Taiwan	1	1	1982	6.4	160	0.954243	-1.28483	0.4675
id.257	Nepal	0	3	-1	7.8	100	1.113943	-1.00789	1.1786
id.258	Nepal	0	2	-1	7.8	100	0.954243	-0.44491	3.2308
id.259	Nepal	0	2	-1	7.8	90	1.623249	-1.10458	3.3
id.260	Nepal	0	2	-1	7.8	85	0.812913	-0.26098	3.2897
id.261	Nepal	0	2	-1	7.8	110	0.778151	-0.56511	1.6331
id.262	Nepal	0	2	-1	7.8	110	0.78533	-0.40938	2.3377
id.263	Nepal	0	2	-1	7.8	95	0.929419	-0.61943	2.0417
id.264	Nepal	0	2	-1	7.8	121	0.812913	-0.14291	4.3178
id.266	Nepal	0	2	-1	7.8	118	0.778151	-0.28997	3.0771
id.267	Nepal	0	3	-1	7.8	123	0.954243	-0.42899	3.3517

id.268	Nepal	0	2	-1	7.8	117	0.778151	-0.68825	1.2303
id.269	Nepal	0	3	-1	7.8	122	1.041393	-0.98843	1.1298
id.270	Nepal	0	2	-1	7.8	119	0.845098	-0.41976	2.2823
id.271	Nepal	0	2	-1	7.8	120	0.778151	-0.99654	0.6048
id.272	Nepal	0	4	-1	7.8	98	0.778151	-0.59912	1.5101
id.273	Nepal	0	3	-1	7.8	102	1.060698	-0.94923	1.0118
id.274	Nepal	0	4	-1	7.8	99	0.778151	-0.55424	1.6747
id.275	Nepal	0	3	-1	7.8	101	0.60206	-0.15889	2.0809
id.276	Nepal	0	3	-1	7.8	97	0.763428	-0.39147	2.436
id.277	Nepal	0	2	-1	7.8	100	1.255273	-1.27165	0.963
id.278	Nepal	0	3	-1	7.8	103	0.799341	-0.67695	1.2627
id.279	Nepal	0	3	-1	7.8	100	0.799341	-0.40949	2.3369
id.285	Nepal	0	14	-1	7.8	90	1.39794	-1.24489	1.4219
id.286	Nepal	0	2	-1	7.8	135	0.778151	-0.31069	2.9339
id.288	Nepal	0	4	-1	7.8	115	0.812913	-0.63827	1.3797
id.289	Nepal	0	4	-1	7.8	114	0.778151	-0.25407	3.3429
id.291	Nepal	0	3	-1	7.8	105	0.94939	-0.65817	1.977
id.292	Nepal	0	2	-1	7.8	106	0.799341	-0.4318	2.22
id.294	Nepal	0	2	-1	7.8	104	1	-0.58956	2.316
id.295	Nepal	0	2	-1	7.8	107	0.863323	-0.66635	1.2935
id.296	Nepal	0	2	-1	7.8	105	0.653213	-0.44105	1.6301
id.297	Nepal	0	3	-1	7.8	102	0.954243	-0.80827	1.3996
id.298	Nepal	0	2	-1	7.8	120	0.845098	-0.60102	1.5038
id.299	Nepal	0	3	-1	7.8	115	1.079181	-0.5516	3.3694
id.300	Nepal	0	2	-1	7.8	95	0.845098	-0.2568	3.3219
id.301	Nepal	0	3	-1	7.8	90	0.778151	-0.27165	3.2102
id.302	Nepal	0	3	-1	7.8	85	1.230449	-0.96257	1.6355

8.2 Extension of debris evaluation

Note: A missing data is replaced with the value -1.

The meaning of the different parameters is well described in the paragraph 2.3.1.

The reference column shows the object or the dimension that is taken as comparison (with well-known measure) in the proportion (1.1) to evaluate the extent of the debris.

ID	Y1	Y2	y1	y2	D [m]	P	p	d [m]	EOD norm.	Reference	Coordinate Reference
id.001	301	312	299	313	1.5	11	14	1.9090	0.31818	Sidewalk	y
id.002	95	153	152	164	36	58	12	7.4482	1.24138	Height	y
id.003	280	289	285	297	1.5	9	12	2	0.33333	Sidewalk	y
id.004	12	304	169	307	6	292	138	2.8356	0.15753	Height	y
id.006	-1	-1	-1	-1	-1	-1	-1	3.5	0.38889	Sidewalk	y
id.007	162	278	163	213	4.5	116	50	1.9396	0.21552	Height	y
id.009	415	451	282	305	4	36	23	2.5555	0.42593	Wall	x
id.010	14	295	95	164	15	281	69	3.6832	0.20463	Height	x
id.011	74	312	240	287	12	238	47	2.3697	0.26331	Height	y
id.012	78	210	239	263	36	132	24	6.5454	0.72727	Height	y
id.013	333	377	326	357	1.5	44	31	1.0568	0.11742	Height	y
id.014	172	269	296	320	5	97	24	1.2371	0.13746	Height	y
id.015	380	410	401	412	1.5	30	11	0.55	0.03056	Sidewalk	y
id.016	138	198	360	449	4	60	89	5.9333	0.65926	Car	x
id.017	376	484	227	429	1.5	108	202	2.8055	0.15586	Sidewalk	x
id.019	212	302	121	143	4	90	22	0.9777	0.10864	Height	y
id.020	150	209	290	345	1.8	59	55	1.6779	0.11186	Car	x
id.021	543	594	366	428	1.5	51	62	1.8235	0.30392	Sidewalk	x
id.022	1243	1555	1870	2063	2.7	312	193	1.6701	0.27837	Door	y
id.023	1221	1550	1645	1816	2.7	329	171	1.4033	0.23389	Door	y
id.024	45	203	283	311	30	158	28	5.3164	0.59072	Height	y
id.025	143	239	41	57	30	96	16	5	0.55556	Height	y
id.026	21	176	183	260	30	155	77	14.903	0.82796	Height	y
id.028	416	534	1332	1407	3	118	75	1.9067	0.3178	Door	y
id.029	40	147	326	405	1.2	107	79	0.8859	0.14766	Window	y
id.030	354	453	993	1034	3	99	41	1.2424	0.13805	Height	y
id.032	168	260	152	182	4	92	30	1.3043	0.14493	Height	y
id.033	44	268	173	215	4	224	42	0.75	0.08333	Height	y
id.035	200	300	70	129	3	100	59	1.77	0.118	Lamp	y
id.036	50	225	146	212	6	175	66	2.2628	0.25143	Height	y
id.039	155	225	49	132	2.2	70	83	2.6085	0.1739	Signal	y
id.040	71	157	201	236	4	86	35	1.6279	0.18088	Height	y
id.043	88	148	329	348	3	60	19	0.95	0.15833	Height	y
id.044	51	191	151	193	3	140	42	0.9	0.1	Door	y
id.045	173	245	140	186	4	72	46	2.5555	0.28395	Height	y
id.046	44	191	233	329	4	147	96	2.6122	0.29025	Height	y
id.050	216	368	199	413	4	152	214	5.6315	0.62573	Car	x
id.051	304	368	292	348	1.5	64	56	1.3125	0.21875	Sidewalk	x
id.053	247	279	174	257	1.2	32	83	3.1125	0.51875	Window	y
id.054	26	179	0	418	1.2	153	418	3.2784	0.65569	Window	x

id.055	404	849	78	265	6	445	187	2.5213	0.14007	Height	y
id.056	237	882	187	436	6	645	249	2.3162	0.25736	Height	y
id.057	156	298	30	60	6	142	30	1.2676	0.08451	Height	y
id.059	84	163	251	369	1.8	79	118	2.6886	0.29873	Car	x
id.060	876	933	598	770	1.5	57	172	4.5263	0.75439	Sidewalk	x
id.061	334	502	371	720	1.8	168	349	3.7392	0.41548	Car	x
id.062	48	436	0	704	2	388	704	3.6288	0.40321	Signal	x
id.064	175	282	219	549	1.8	107	330	5.5514	0.30841	Car	x
id.068	465	544	287	481	6	79	194	14.734	0.44649	Car	x
id.069	271	351	149	362	1.2	80	213	3.195	0.5325	Rail	x
id.070	44	159	222	269	9	115	47	3.6782	0.61304	Height	y
id.071	214	508	490	589	61	294	99	20.540	2.28231	Lenght	x
id.079	219	342	455	540	3.5	123	85	2.4186	0.26874	Height	y
id.081	5	284	315	438	6	279	123	2.6451	0.44086		
id.082	280	367	239	524	1.8	87	285	5.8965	0.98276	Car	x
id.084	217	328	171	333	3	111	162	4.3783	0.48649	Height	y
id.086	187	260	174	260	3	73	86	3.5342	0.19635	Height	y
id.087	99	318	206	246	3	219	40	0.5479	0.09132	Height	y
id.088	213	341	227	262	3	128	35	0.8203	0.09115	Height	y
id.089	26	314	351	415	3	288	64	0.6666	0.03704	Height	y
id.090	22	326	309	382	3	304	73	0.7203	0.04002	Height	y
id.091	51	483	1	143	3	432	142	0.9861	0.10957	Height	y
id.092	35	392	421	590	6	357	169	2.8403	0.56807	Height	y
id.093	191	395	22	186	3	204	164	2.4117	0.13399	Height	y
id.094	352	445	651	677	6	93	26	1.6774	0.18638	Height	y
id.095	105	283	52	144	6	178	92	3.1011	0.51685	Height	y
id.096	167	277	362	407	6	110	45	2.4545	0.27273	Height	y
id.097	299	367	436	472	1.2	68	36	0.6352	0.10588	Window	y
id.098	201	385	179	300	3	184	121	1.9728	0.13152	Height	y
id.100	417	481	699	735	2.3	64	36	1.2937	0.10781	Door	y
id.101	276	353	106	201	2.3	77	95	2.8376	0.47294	Door	y
id.102	1	769	184	330	9	768	146	1.7109	0.1901	Height	y
id.103	355	643	364	434	6	288	70	1.4583	0.03472	Height	y
id.104	103	821	198	533	6	718	335	2.7994	0.46657	Height	y
id.106	218	329	73	157	1.8	111	84	1.3622	0.15135	Height	y
id.107	224	383	308	418	3	159	110	2.0754	0.23061	Height	y
id.112	366	435	240	318	3	69	78	3.3913	0.08075	Height	y
id.114	316	375	373	401	3	59	28	1.4237	0.05932	Height	y
id.116	180	317	146	231	3	137	85	1.8613	0.10341	Height	y
id.118	455	590	147	203	2	135	56	0.8296	0.02514	Signal	y
id.120	472	845	644	845	2	373	201	1.0777	0.03266	Signal	y

id.122	218	323	34	69	3	105	35	1	0.11111	Signal	y
id.123	447	474	329	420	3	27	91	10.111	0.84259	Height	y
id.129	648	868	478	509	3	220	31	0.4227	0.07045	Height	y
id.133	1712	3368	864	1416	3	1656	552	1	0.16667	Height	y
id.134	1796	3432	976	1476	3	1636	500	0.9168	0.15281	Height	y
id.135	2276	3340	3380	3912	3	1064	532	1.5	0.16667	Height	y
id.136	1948	3764	1900	2420	3	1816	520	0.8590	0.07809	Height	y
id.137	1039	2969	1859	2439	3	1930	580	0.9015	0.15026	Height	y
id.138	1024	2388	1708	2140	3	1364	432	0.9501	0.15836	Height	y
id.139	1346	2168	1102	1326	3	822	224	0.8175	0.09084	Height	y
id.140	1225	2213	3139	3459	3	988	320	0.9716	0.10796	Height	y
id.141	2068	2420	1952	2168	3	352	216	1.8409	0.06136	Height	y
id.142	481	2230	3158	4210	3	1749	1052	1.8044	0.2005	Height	y
id.144	1352	2208	36	552	3	856	516	1.8084	0.20093	Height	y
id.145	2088	2844	2780	3228	3	756	448	1.7778	0.2963	Height	y
id.146	1396	2452	1868	2452	3	1056	584	1.6590	0.27652	Height	y
id.147	703	2338	278	1252	3	1635	974	1.7871	0.19857	Height	y
id.148	1870	2426	2528	2819	3	556	291	1.5701	0.13085	Height	y
id.149	1664	2312	2038	2234	3	648	196	0.9074	0.10082	Height	y
id.150	1956	2402	730	1308	3	446	578	3.8878	0.32399	Height	y
id.151	2069	2615	388	1117	3	546	729	4.0054	0.13352	Height	y
id.152	4052	5744	724	1284	3.5	1692	560	1.1583	0.05516	Height	y
id.153	1692	3488	4572	5116	3.5	1796	544	1.0601	0.11779	Height	y
id.154	1621	2684	1279	1537	3	1063	258	0.7281	0.04854	Height	y
id.155	2552	3117	1903	2069	3	565	166	0.8814	0.05876	Height	y
id.158	1988	2488	1508	1764	3	500	256	1.536	0.1536	Height	y
id.162	3460	4604	632	1288	3	1144	656	1.7202	0.14336	Height	y
id.163	1469	4389	2179	3449	3	2920	1270	1.3047	0.21747	Height	y
id.164	4312	5780	3224	3904	3	1468	680	1.3896	0.0386	Height	y
id.165	3960	5344	2972	3404	3	1384	432	0.9364	0.06243	Height	y
id.166	4464	5572	1596	1840	3	1108	244	0.6606	0.02753	Height	y
id.167	2892	3888	8	292	3	996	284	0.8554	0.07129	Height	y
id.168	1688	2276	1760	2080	3	588	320	1.632653	0.18141	Height	y
id.169	1273	2178	2269	2798	3	905	529	1.753591	0.29227	Height	y
id.170	1498	2294	2164	2358	3	796	194	0.731156	0.12186	Height	y
id.172	1944	3808	1112	1633	3	1864	521	0.838519	0.07623	Height	y
id.173	2244	3868	460	886	3	1624	426	0.786946	0.06558	Height	y
id.174	1448	2476	1676	1928	3	1028	252	0.735409	0.16342	Height	y
id.175	599	2481	3123	3450	9	1882	327	1.563762	0.17375	Height	y
id.176	644	2603	2376	2739	9	1959	363	1.667688	0.1853	Height	y
id.177	564	2509	2670	3031	9	1945	361	1.670437	0.1856	Height	y

id.184	1039	1363	165	624	3	324	459	4.25	0.47222	Height	y
id.190	2506	2894	146	710	3	388	564	4.360825	0.12113	Height	y
id.191	2207	3099	60	933	3	892	873	2.936099	0.16312	Height	y
id.192	2768	3111	2759	2933	3	343	174	1.521866	0.10146	Sidewalk	y
id.193	1112	1716	1616	1884	3	604	268	1.331126	0.1479	Sidewalk	x
id.194	165	972	585	987	3	807	402	1.494424	0.16605	Sidewalk	x
id.195	1108	1525	1417	2048	3	417	631	4.539568	0.15132	Door	y
id.196	747	1711	837	1603	6	964	766	4.767635	0.52974	Height	y
id.201						0	0	3	0.33333		
id.204	1435	2300	525	1141	4	865	616	2.848555	0.63301	Car	x
id.205	1108	1880	76	336	3	772	260	1.010363	0.11226	Height	y
id.206	942	1274	584	706	3.5	332	122	1.286145	0.1429	Height	y
id.209						0	0	0.5	0.04167		
id.211	1117	1376	828	1016	3	259	188	2.177606	0.24196	Door	y
id.212	1278	1708	1002	1260	2.1	430	258	1.26	0.21	Door	y
id.214	363	1305	2279	3584	2	942	1305	2.770701	0.06597	Car	x
id.220	408	696	428	1048	1.8	288	620	3.875	0.21528	Car	x
id.221	452	788	968	1713	1.8	336	745	3.991071	0.12094	Car	x
id.224	759	846	1459	1765	0.32	87	306	1.125517	0.18759	License plate	x
id.226	2056	2708	280	912	4	652	632	3.877301	0.12924	Car	x
id.227	1156	2496	3448	3756	4	1340	308	0.919403	0.10216	Car	x
id.228	1308	1892	2284	2728	1.8	584	444	1.368493	0.22808	Car	x
id.232	516	1414	1315	1411	3	898	96	0.320713	0.02673	Height	x
id.234	476	1557	694	1014	3	1081	320	0.888067	0.02691	Height	y
id.235	332	1221	1226	1509	3	889	283	0.955006	0.03979	Height	y
id.236	516	1414	1315	1411	3	898	96	0.320713	0.02673	Height	y
id.238	862	1084	650	921	3	222	271	3.662162	0.61036	Height	y
id.239	3308	3442	3190	3338	1.8	134	148	1.98806	0.16567	Car	x
id.240	565	672	766	864	4	107	98	3.663551	0.61059	Car	x
id.241	221	1085	1293	2101	3	864	808	2.805556	0.31173	Height	y
id.242	480	1240	1300	1534	2.1	760	234	0.646579	0.05388	Door	y
id.243	794	1356	1200	1368	2.1	562	168	0.627758	0.10463	Door	y
id.244	492	1094	1486	1692	2.1	602	206	0.718605	0.07984	Door	y
id.248	910	1200	717	806	2.1	290	89	0.644483	0.10741	Door	y
id.249	958	1218	1347	1437	2.1	260	90	0.726923	0.01773	Door	y
id.250	648	900	174	438	1.2	252	264	1.257143	0.20952	Window	y
id.251	111	462	930	1363	1.2	351	433	1.480342	0.16448	Window	y
id.254	916	1188	1676	1980	1.2	272	304	1.341176	0.14902	Window	y
id.256	510	1040	794	912	2.1	530	118	0.467547	0.05195	Door	y
id.257	440	580	397	452	3	140	55	1.178571	0.09821	Height	y
id.258	500	669	438	620	3	169	182	3.230769	0.35897	Height	y

id.259	357	447	96	261	1.8	90	165	3.3	0.07857	Car	x
id.260	2140	2604	360	1632	1.2	464	1272	3.289655	0.54828	Window	y
id.261	1992	3660	1356	2264	3	1668	908	1.633094	0.27218	Height	y
id.262	2460	2884	2184	2656	2.1	424	472	2.337736	0.38962	Door	y
id.263	1916	2924	2088	3068	2.1	1008	980	2.041667	0.2402	Door	y
id.264	1208	2064	240	1472	3	856	1232	4.317757	0.71963	Height	y
id.266	1800	2780	916	2352	2.1	980	1436	3.077143	0.51286	Door	y
id.267	1252	1484	1808	2456	1.2	232	648	3.351724	0.37241	Door	y
id.268	3128	3276	204	773	0.32	148	569	1.23027	0.20505	License plate	x
id.269	1930	2639	1345	1612	3	709	267	1.12976	0.10271	Height	y
id.270	1364	2200	3160	3796	3	836	636	2.282297	0.38038	Height	y
id.271	1555	2180	2651	2777	3	625	126	0.6048	0.1008	Height	y
id.272	587	736	363	438	3	149	75	1.510067	0.25168	Height	y
id.273	513	682	492	549	3	169	57	1.011834	0.11243	Height	y
id.274	620	699	322	385	2.1	79	63	1.674684	0.27911	Door	y
id.275	600	710	774	883	2.1	110	109	2.080909	0.69364	Door	y
id.276	658	808	1036	1210	2.1	150	174	2.436	0.406	Door	y
id.277	412	736	226	330	3	324	104	0.962963	0.0535	Height	y
id.278	562	720	845	940	2.1	158	95	1.262658	0.21044	Door	y
id.279	656	851	654	871	2.1	195	217	2.336923	0.38949	Door	y
id.285	1848	2712	1528	2113	2.1	864	585	1.421875	0.05688	Door	y
id.286	1088	2540	2200	3620	3	1452	1420	2.933884	0.48898	Height	y
id.288	2071	2485	955	1227	2.1	414	272	1.37971	0.22995	Door	y
id.289	1712	2496	680	1928	2.1	784	1248	3.342857	0.55714	Door	y
id.291	1944	2968	1624	2588	2.1	1024	964	1.976953	0.21966	Door	y
id.292	1344	2044	3748	4488	2.1	700	740	2.22	0.37	Door	y
id.294	2024	3152	1536	2780	2.1	1128	1244	2.315957	0.25733	Door	y
id.295	2459	3412	1416	2003	2.1	953	587	1.293494	0.21558	Door	y
id.296	1813	2597	1048	1474	3	784	426	1.630102	0.36224	Height	y
id.297	1849	2357	51	288	3	508	237	1.399606	0.15551	Height	y
id.298	2076	3168	1786	2568	2.1	1092	782	1.503846	0.25064	Door	y
id.299	2248	2784	5092	5952	2.1	536	860	3.369403	0.28078	Door	y
id.300	1594	2060	3255	3771	3	466	516	3.321888	0.55365	Height	y
id.301	2020	2334	3450	3786	3	314	336	3.210191	0.53503	Height	y
id.302	2740	3648	2352	2847	3	908	495	1.635463	0.10903	Height	y

8.3 Data visualization script

```
#import modules in python
```

```
import matplotlib.pyplot as plt
import numpy as np

#load dataset
matrix    =    np.loadtxt('Database_pictures_python_multi.csv',
usecols=(2,3,4,5,6,7,8),
                delimiter = ';', skiprows=1)
index_list = [1,2,3,4,5]

#define graphs properties
titles    =    ['Stories','Year','Magnitude',    'Distance    from
epicenter', 'Height']
options = [0]
for i in range(len(index_list)):
    f, axs = plt.subplots(1,1,figsize=(10,5))
    plt.title(titles[i])
    plt.ylabel('Extension of debris')
    plt.xlabel(titles[i])
    good_rows_masonry        =        np.logical_and(matrix[:,
index_list[i]]>=0,
                                       matrix[:, 0]==0)

    good_rows_concrete        =        np.logical_and(matrix[:,
index_list[i]]>=0,
                                        matrix[:, 0]==1)

    plt.scatter(matrix[good_rows_masonry,      index_list[i]],
matrix[good_rows_masonry, 6],
                marker = 'o', label='masonry')
    plt.scatter(matrix[good_rows_concrete,     index_list[i]],
matrix[good_rows_concrete, 6],
                marker = 'v', label='concrete')
    plt.legend(loc=1)
```

```
plt.show()
```

8.4 Training and test script

```
#import modules in python
from sklearn.model_selection import train_test_split
import numpy as np
from sklearn.neighbors import KNeighborsRegressor as KNR
from sklearn.preprocessing import MinMaxScaler
from sklearn import svm
from sklearn.tree import DecisionTreeRegressor
import sklearn
from sklearn.ensemble import RandomForestRegressor
from sklearn import linear_model
from sklearn.linear_model import ElasticNet
from sklearn.linear_model import Ridge
from sklearn.svm import SVR
import matplotlib.pyplot as plt
from sklearn import tree
from sklearn.neural_network import MLPRegressor
import seaborn as sns
from sklearn.manifold import TSNE
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import mean_absolute_error

#import database from folder
db      =      np.loadtxt('Database_pictures_python_multi.csv',
usecols=(2,3,4,5,6,7,8),
                delimiter = ';', skiprows=1)

index_list = [1,3,4,5]
X = db[np.sum(db[:,index_list],1) > 0,:][:,index_list]
```

```
y = db[:,6]
y = y.reshape(-1,1)

#Preprocessing
scaler = StandardScaler()
X_scaled = scaler.fit_transform(X)
#X_scaled = preprocessing.scale(X)
#min_max_scaler = preprocessing.MinMaxScaler()
#X_scaled = min_max_scaler.fit_transform(X)
#X_embedded = TSNE(n_components=2).fit_transform(X_scaled)

X_train, X_test, y_train, y_test = train_test_split(X_scaled,
y, test_size=0.33
                                                    ,
shuffle=True, random_state=0)

#run algorithms
clf = KNR(n_neighbors=5)
clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("KNR")
print("Train   set   R^2:   {:.2f}".format(clf.score(X_train,
y_train)))
print("Test   set   R^2:   {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = linear_model.Lasso(alpha=0.01, max_iter=10)
clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("Lasso")
```

```
print("Train    set    R^2:    {:.2f}".format(clf.score(X_train,
y_train)))
print("Test    set    R^2:    {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))


clf = ElasticNet(alpha=0.01)
clf.fit(X_train,y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("Elastic Net")
print("Train    set    R^2:    {:.2f}".format(clf.score(X_train,
y_train)))
print("Test    set    R^2:    {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = Ridge(alpha=0.1)
clf.fit(X_train,y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("Ridge")
print("Train    set    R^2:    {:.2f}".format(clf.score(X_train,
y_train)))
print("Test    set    R^2:    {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = SVR(C=0.1, epsilon=0.001, kernel='rbf')
clf.fit(X_train,y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("SVR")
```

```
print("Train set R^2: {:.2f}".format(clf.score(X_train,
y_train)))
print("Test set R^2: {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = RandomForestRegressor(max_depth=10, n_estimators=20,
min_samples_split=40)
clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("random forest")
print("Train set R^2: {:.2f}".format(clf.score(X_train,
y_train)))
print("Test set R^2: {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = tree.DecisionTreeRegressor(min_samples_split=25,
min_samples_leaf=10, presort=True)
clf = clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("Tree")
print("Train set R^2: {:.2f}".format(clf.score(X_train,
y_train)))
print("Test set R^2: {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))

clf = MLPRegressor(alpha=0.1, batch_size=10,
hidden_layer_sizes=1000,
                   learning_rate='constant',
learning_rate_init=0.001, solver='adam',
```

```
                        activation='relu', random_state=27)
clf = clf.fit(X_train, y_train)
y_predict = clf.predict(X_test)
MARD = mean_absolute_error(y_predict, y_test)
print("MLP Regressor")
print("Train   set   R^2:   {:.2f}".format(clf.score(X_train,
y_train)))
print("Test    set    R^2:    {:.2f}".format(clf.score(X_test,
y_test)))
print("Mean absolute error {:.2f}".format(MARD))
'''
#producing charts
X_embedded                =                TSNE(n_components=2,
random_state=0).fit_transform(X_scaled)
g = sns.scatterplot(X_embedded[:,0], X_embedded[:,1])
g.set(xticks=[])
g.set(yticks=[])
g.set_title('t-SNE data visualization')
plt.show()
#sns.scatterplot(X_embedded[:,0], X_embedded[:,1], hue=y[:,0])
```

8.5 Results visualization script

```
import numpy as np
import matplotlib.pyplot as plt
import matplotlib as mpl

mpl.style.use('default')
x = np.arange(3)
plt.bar(x, height= [0.29,0.27,0.26], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
```

```
plt.ylabel('R^2')
plt.title('KNR-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()

x = np.arange(3)
plt.bar(x, height= [0.33,0.31,0.31], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('Lasso-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
```

```
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()

x = np.arange(3)
plt.bar(x, height= [0.33,0.31,0.31], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('Elastic Net-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off
```

```
plt.show()

x = np.arange(3)
plt.bar(x, height= [0.33,0.31,0.31], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('Ridge-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()

x = np.arange(3)
plt.bar(x, height= [0.27,0.26,0.26], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('SVR-MARD')
```

```
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()

x = np.arange(3)
plt.bar(x, height= [0.23,0.22,0.22], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('Random forest-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
```

```
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()

x = np.arange(3)
plt.bar(x, height= [0.23,0.28,0.27], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('Decision tree-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()
```

```
x = np.arange(3)
plt.bar(x, height= [0.24,0.24,0.24], color='b')
plt.xticks(x, ['5','4','3'])

plt.xlabel('Number of features')
plt.ylabel('R^2')
plt.title('MLP regressor-MARD')
plt.axis([-1, 3, -0.1, 0.6])
plt.grid(False)
#plt.axes().spines['right'].set_visible(False)
#plt.axes().spines['top'].set_visible(False)
plt.axes().spines['bottom'].set_position(('data', -0.1))
plt.axes().tick_params(axis='x', pad=5)
plt.axhline(y=0, color='k')
plt.axes().axhline(linewidth=0, color="k")
plt.tick_params(
    axis='x',          # changes apply to the x-axis
    which='both',          # both major and minor ticks are
affected
    bottom=False,      # ticks along the bottom edge are off
    top=False,         # ticks along the top edge are off
    labelbottom=True) # labels along the bottom edge are off

plt.show()
```

8.6 PHI Challenge labeling algorithm

```
from __future__ import absolute_import
from __future__ import division
from __future__ import print_function

import argparse
import sys
```

```
import tensorflow as tf

parser = argparse.ArgumentParser()
parser.add_argument(
    '--image', required=True, type=str, help='Absolute path to
image file.')
parser.add_argument(
    '--num_top_predictions',
    type=int,
    default=5,
    help='Display this many predictions.')
parser.add_argument(
    '--graph',
    required=True,
    type=str,
    help='Absolute path to graph file (.pb)')
parser.add_argument(
    '--labels',
    required=True,
    type=str,
    help='Absolute path to labels file (.txt)')
parser.add_argument(
    '--output_layer',
    type=str,
    default='final_result:0',
    help='Name of the result operation')
parser.add_argument(
    '--input_layer',
    type=str,
    default='DecodeJpeg/contents:0',
    help='Name of the input operation')


def load_image(filename):
```

```
  """Read in the image_data to be classified."""
  return tf.gfile.FastGFile(filename, 'rb').read()


def load_labels(filename):
  """Read in labels, one label per line."""
  return [line.rstrip() for line in tf.gfile.GFile(filename)]


def load_graph(filename):
  """Unpersists graph from file as default graph."""
  with tf.gfile.FastGFile(filename, 'rb') as f:
    graph_def = tf.GraphDef()
    graph_def.ParseFromString(f.read())
    tf.import_graph_def(graph_def, name='')


def run_graph(image_data, labels, input_layer_name,
output_layer_name,
              num_top_predictions):
  with tf.Session() as sess:
    # Feed the image_data as input to the graph.
    #   predictions will contain a two-dimensional array,
where one
    #  dimension represents the input image count, and the
other has
    #  predictions per class
    softmax_tensor                                          =
sess.graph.get_tensor_by_name(output_layer_name)
    predictions, = sess.run(softmax_tensor, {input_layer_name:
image_data})

    # Sort to show labels in order of confidence
    top_k = predictions.argsort()[-num_top_predictions:][::-1]
```

```
    for node_id in top_k:
      human_string = labels[node_id]
      score = predictions[node_id]
      print('%s (score = %.5f)' % (human_string, score))

    return human_string, score


def main(argv):
  """Runs inference on an image."""
  if argv[1:]:
    raise ValueError('Unused Command Line Args: %s' %
argv[1:])

  if not tf.gfile.Exists(FLAGS.image):
    tf.logging.fatal('image file does not exist %s',
FLAGS.image)

  if not tf.gfile.Exists(FLAGS.labels):
    tf.logging.fatal('labels file does not exist %s',
FLAGS.labels)

  if not tf.gfile.Exists(FLAGS.graph):
    tf.logging.fatal('graph file does not exist %s',
FLAGS.graph)

  # load image
  image_data = load_image(FLAGS.image)

  # load labels
  labels = load_labels(FLAGS.labels)

  # load graph, which is stored in the default session
  load_graph(FLAGS.graph)
```

```
  run_graph(image_data,       labels,       FLAGS.input_layer,
FLAGS.output_layer,
            FLAGS.num_top_predictions)


if __name__ == '__main__':
  FLAGS, unparsed = parser.parse_known_args()
  tf.app.run(main=main, argv=sys.argv[:1]+unparsed)
```

REFERENCES

[1] Andreas C. Muller, Sarah Guido "*Introduction to Machine Learning with Python".* O'Reilly Media, Inc., Sebastopol, 22 September 2016.

[2] Ian Goodfellow, Yoshua Bengio, Aaron Courville. "*Deep Learning".* The MIT Press, 2016.

[3] James Bialas. "*Object-based classification of earthquake damage from high-resolution optical imagery using machine learning".* Michigan University, 2015.

[4] Chul Min Yeum, Shirley J. Dyke, Julio Ramirez. *"Visual data classification in post-event building reconnaisance".* Elsevier, 22 October 2017.

[5] T. Kim, O. Kwon, J. Song "*Assestment of seismic responses of nonlinear structural system using deep-learning*", 11NCEE Conference, Los Angeles, 2018.

[6] M. Kovacevic, Z. Stojadinovic, D. Marinkovic, B. Stojadinovic. *"Sampling and machine learning methods for a rapid earthquake loss assessment system".* 11NCEE Conference, Los Angeles, 2018.

[7] C. M. Yeum, S. J. Dyke, B. Benes, T. Hacker et al. *"Automating visual data processing to support post-earthquake reconnaisance".* 11NCEE Conference, Los Angeles, 2018.

[8] M. Matsuoka, Y. Ishii, N. Maki, K. Horie, S. Tanaka et al. *"Damaged building recognition using deep learning with photos taken after the Kobe earthquake".* 11NCEE Conference, Los Angeles, 2018.

[9] Cimellaro G.P. (2016) "*Urban Resilience for Emergency Response and Recovery*", Springer International Publishing, Switzerland, 2016.

[10] Bishop C. "*Pattern Recognition and Machine Learning*", Springer, 2006.

[11] Mitchell T. M., "*Machine Learning*", McGraw Hill, 1997.

[12] Bai JW, Hueste MBD, Gardoni P. "*Probabilistic Assessment of Structural Damage due to Earthquakes for Buildings in Mid-America*". Journal of Structural Engineering, 2009.

[13] "*Scikit-learn user guide*", Scikit-learn developers, 2018.

[14] Y.Gao, K.Mosalam "*Deep Transfer Learning for Image-Based Structural Damage Recognition*". Computer-Aided Civil and Infrastructure Engineering, April 2018.

[15] S.J.Pan, Q.Yang "*A Survey on Transfer Learning*" IEEE transactions on knowledge and data engineering, 2010.

[16] Chollet, F. (2017). "*Xception: Deep learning with depthwise separable convolutions.*" arXiv preprint, 1610-02357.