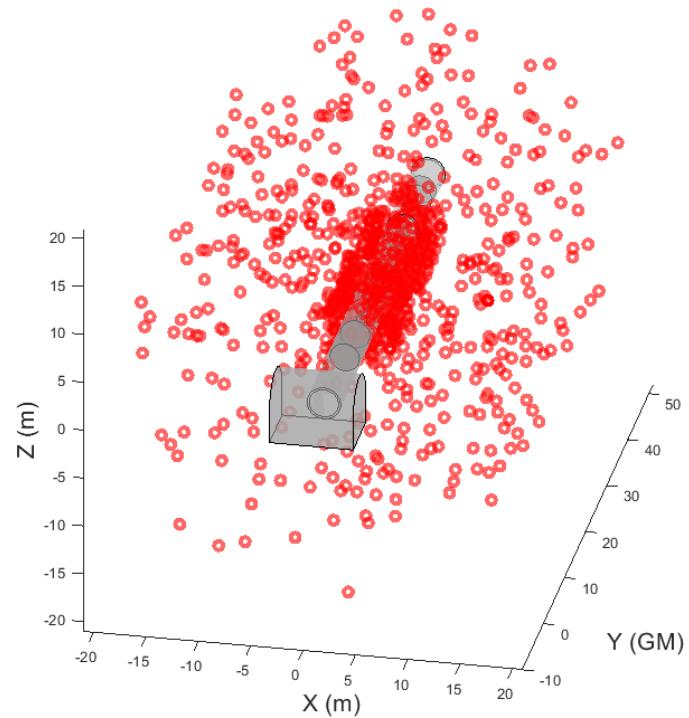


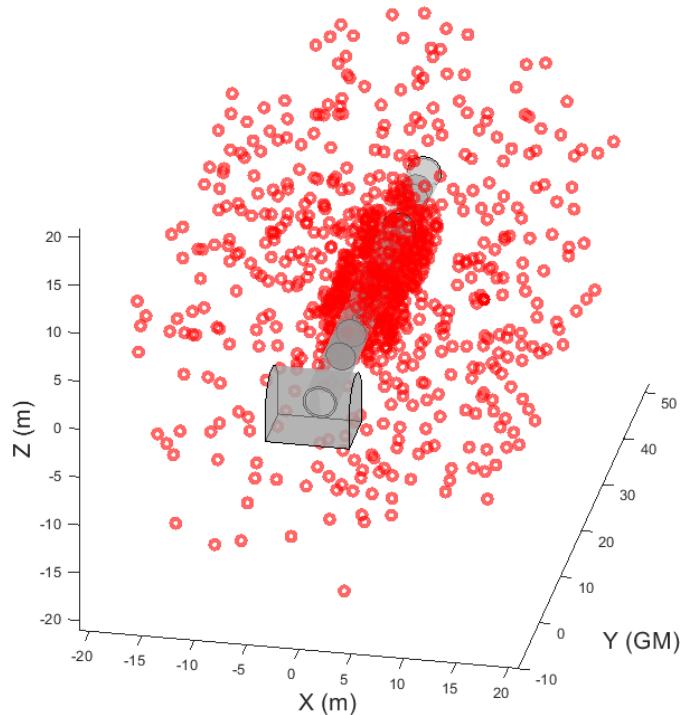
# Final Project

# Predicting Temperature of Nuclear Waste Canisters



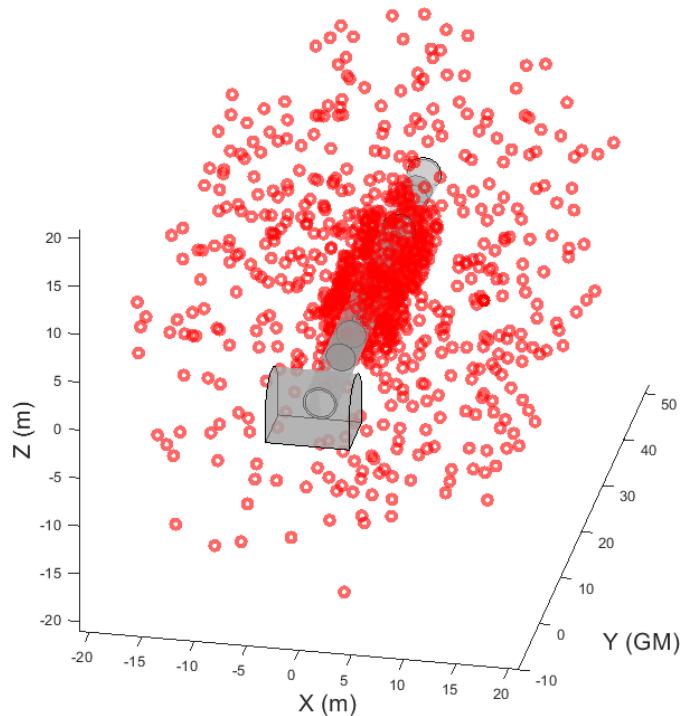
# Predicting Temperature of Nuclear Waste Canisters

- Location of sensors
- Temperature
- Pressure
- Humidity



# Predicting Temperature of Nuclear Waste Canisters

- Location of sensors
- Temperature
- Pressure
- Humidity





- Make an account ([www.kaggle.com](http://www.kaggle.com))
- Join the competition
- Download dataset
- There will be a leaderboard and a



Prize for the winner!

- Team of 2 (should be registered on moodle by Tuesday)
- Deadline for the leaderboard is 18<sup>th</sup> of May.
- Deadline for submission on moodle (poster + code) is 23<sup>rd</sup> of May.
- Best teams will present at the last week of class.
- Grading:
  - 50% poster
  - 50% performance (you need to pass a baseline)

# Poster example

**EPFL Predict Building Collapse**

As civil engineer, one of our priority is the safety of people in buildings. These constructions can be damaged and even collapse during an earthquake. The goal of this exercise is to code a machine learning able to predict if a building will collapse during a seismic.

**Dataset**

We have 3 features files with metadata on ground motion: `train.csv`, `val.csv`, `features_test.csv`. This features come from the 17'000 test set. We have 3 targets files. And we also have 3 targets files.

**Features**

The features files contain columns with some intensity measures such as `max_acceleration`, `max_deceleration`, `incremental velocity` and so on. Each now represents an earthquake.

**Dataset**

After taking a look at the 6 files, we don't see any problem with the data but in the features, there are some weird values.

- We decide to replace NaN and -999 with some values.
  - To use the interpolation function. We only have the possibility to use the `interpolate` function for Dataframe. As the function `interpolate` leaves NaN in some places.
  - We replace the last missing values with the mean of the columns to which they belong.
- We merge our `features_test` with our targets files. We do that for the training and validation data. We do that because we want to know if for known features, we can predict the right target.
- To scale some features, we had to multiply some columns of features by the column `max_acceleration`.
- Something that can improve the performance and reduce the underfitting of our model is to use polynomial feature expansion. So, we passed from linearly features to features with the third degree.

**Figure 1: Interpolation**

**Figure 2: polynomial feature expansion**

**Model**

Then comes the question to not use a neural network. We tried some trick to make it work. We tried to make a key for this part of the exercise to test all the possibility.

**Trials**

We tried to do it in two different ways. The first one is to normalize each column separately. Then we merge all the features together and the results was a bit better. But the most satisfying result was reached without doing a normalization.

**Goal:**

Make a machine learning program that is able to classify if a building will collapse or not during an earthquake. We use supervised learning because we have all informations needed.

**Figure 3: Function Gini Impurity**

**Results**

	Training set	Validation set	Test set
NN model	0.790	0.781	0.774
DTC model	0.842	0.832	0.832

**Discussion**

It took us a bit of time to really start the project. We had to learn how to use Python to run our first code. But after revising the exercise sessions we were finally able to run it. To improve our results we used some machine learning techniques to replace the missing values. Then we should use the `train` and `validation` functions. Doing `train` and `validation` is a big difference. To finish, this project obligate us to see again all the code.

We liked making this project although it was a bit complicated at the beginning. This subject is very interesting and we can apply it in many cases and even more in our studies, so it was really motivating. We are satisfied by our results because we have a good accuracy of a building to collapse due to a seismic in more than 8/10 times.

**Figure 1: Interpolation**

**Figure 2: polynomial feature expansion**

**Figure 3: Function Gini Impurity**

**Figure 4: Decision Tree**

**Figure 5: Neural Network**

**Figure 6: Random Forest**

**Figure 7: Gradient Boost**

**Figure 8: MLP Classifier**

**Figure 9: Ensemble**

## Building Collapse

**Random Forest:**

Make several decision trees. Then with samples of the data, train the decision trees. Afterward, make a vote about the result (similar to KNN).

**Data:**

Given

The 105th firsts columns are the spectral accelerations at 105 periods. The other ones are derived values of spectral accelerations and properties of the building and earthquake.

**Processing:**

- Drop useless columns :
  - Unnamed: 0
  - "Ground Motion ID"
- Interpolate by row to fill the missing values.
- Transformation of the "YES" and "NO" into 1 and 0
- Normalise spectral accelerations by scale factor (we didn't use this method in our code, we forgot)
- Score normalization

**Gradient Boost:**

Iterative algorithm that minimizes a loss function by iteratively choosing a function that points towards the negative gradient. Hyper-parameters that we changed

**MLP Classifier :**

We did ensemble learning with Random Forest Classifier and Gradient Boost Classifier. We also did a neural network thanks to MLP Classifier.

At the end we group these methods to ensemble learning on the test set. Our models ask only a little preparation of data.

**Results:**

Shape of sets (117136, 117)	(2912, 117)	(48352, 116)	
Train	Validation	Test	
Random Forest	93.53%	80.29%	81.1%
Gradient Boost	88.02%	81.21%	80.5%
Random Forest + Gradient Boost	80.61%	75.36%	79.8%
Ensemble			81.3%

**Discussion:**

We had pretty good results

- We try a lot of other method such as SVM and KNN but they were not relevant
- It was a bit difficult to code but at the end of the day it's so satisfying
- What we can improve : learn how to read !
- It's a good idea to see that we had to multiply spectral acceleration by scale factor
- We should also do more submissions because our model did very well in the validation set than on test at the end

**Sources:**

- https://www.kaggle.com/c/vita
- https://www.kaggle.com/c/vita

# Poster example

## BUILDING COLLAPSE DETECTION

Classify whether a building collapses based on some earthquake data

Lara - 311555 David - 324309 Vincent - 313961

> Team name on AGCrowd VDS

> AGCrowd VDS

> AGCrowd VDS

**Goal:** provided with **2 features files** containing data for about 17000 past earthquakes as well as the corresponding response of a 4-story steel structure, **2 targets files**, we shall develop a **machine learning model** able to **predict the potential collapse** of a new set of data.

### Data:

- Training and validation dataset:
  - First 105 columns represent function of **spectral acceleration** with respect to a period
  - Other columns are **earthquakes characteristics** (magnitude, velocity,...)

- Test dataset:
  - Whether the 4-story steel structure **collapses or not**

Train set: 70%

Val set: 20%

Test set: 28%

Data of more than 17000 past earthquakes

### 1 Pre processing

#### 1.a Interpolation

- **Interpolate** in the train data set along the **horizontal axis**
  - Reason: the first 105 columns represent the function of spectral acceleration depending on period  $[\text{Sa}(T)]$  and assuming it is linear, interpolation is the best solution

#### 1.b Imputation

- Imputing rows with **-999** missing values by replacing **-999** by 0
  - Reason: preserve test datasets shape

#### 1.c Scaling factor

- Multiply every feature related to the acceleration
  - $\text{sa\_avg}$ ,  $\text{fiv3}$  and every  $\text{Sa}(T)$

#### 1.d Data cut and normalisation

- Only applied to the regressions data
  - Removed **outliers** present in the  $\text{fiv3}$  and  $\text{sa\_avg}$  columns
  - Features displaying a **heavy-tail distribution** are transformed using a **logarithm** or **square-root scaling**
  - **Z-normalisation** on all the data

### 2 Machine Learning Models

#### 2.a Logistic regression

**Features:**  
We considered all features except the  $\text{Sa}(T)$  columns

**Features expansion:**  
We expand the features to obtain a stronger and more complex fitting power.

**Setting the parameters:**

- Regularization strength:  $C = 1e-2$

**Results:**  
Had a **0.794** test accuracy which is acceptable

**Discussion:**  
The logistic regression is a **simple but powerful tool** which is meaningful for data structures that aren't too complex. **Features expansion** is a useful tool to make logistic regression stronger, up to a certain extent.

#### 2.b Neural network

3 layer neural network with 2 **relu** activations and 1 **sigmoid** activation

**Features:**  
We tried considering only several features instead of taking them all. However, the results on the accuracy were in this case lower. We therefore kept **all features** to train our model.

**Setting the parameters:**

- Batch size = 100
- Number of epochs = 25
- Loss function = **Binary Cross-entropy**
- Optimizer = Adam

**Results:**  
Had a **0.765** test accuracy which isn't very satisfactory.

**Discussion:**  
Initially, we started overfitting, we therefore tried different solutions:
 

- Adding weight regularization
- Adding dropout

We hoped to have a much better test accuracy with a NN but it wasn't the case even after tuning the hyper-parameters.

#### 3 Overall results

This table makes us clearly realize that the **Random Forest Classifier** gave us the **best test accuracy**. However, we are also able to realize that the model is **overfitting**. The train accuracy is too big and is thus unable to generalize. By **tuning our hyper-parameters**, we would be able to deal with the overfitting problem and thus increasing our test accuracy.

**Final thoughts and future:** We spent a lot of time on developing our neural network as well as dealing with our data. We tried at the very end the **Random Forest Classifier** which clearly performed in a much better way on our data, which could also be improved. If we had more time, we would then implement an algorithm able to find the **best hyper-parameters** and try our model with those. We think that by implementing this, we could easily increase our test accuracy.

#### 4 References

- **Neural Networks** for binary Classification
  - <https://michael-massini.github.io/2021-02-18-ml-artificial-neural-networks-for-binary-classification/>
- **Machine learning course: Introduction to machine learning** by Professor Alexandre Alahi
- **Random Forest Hyperparameter tuning**
  - <https://towardsdatascience.com/random-forest-hyperparameter-tuning-117a559382e5>