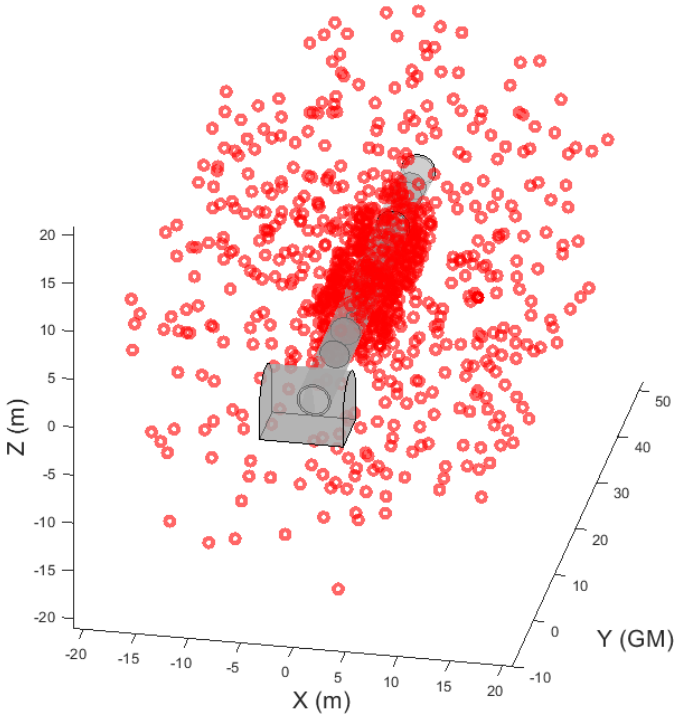
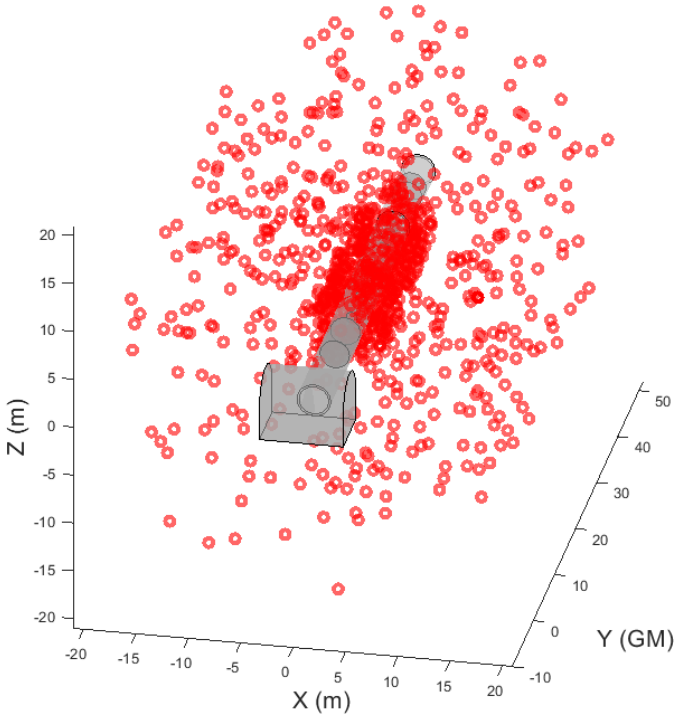
Final Project

Predicting Temperature of Nuclear Waste Canisters

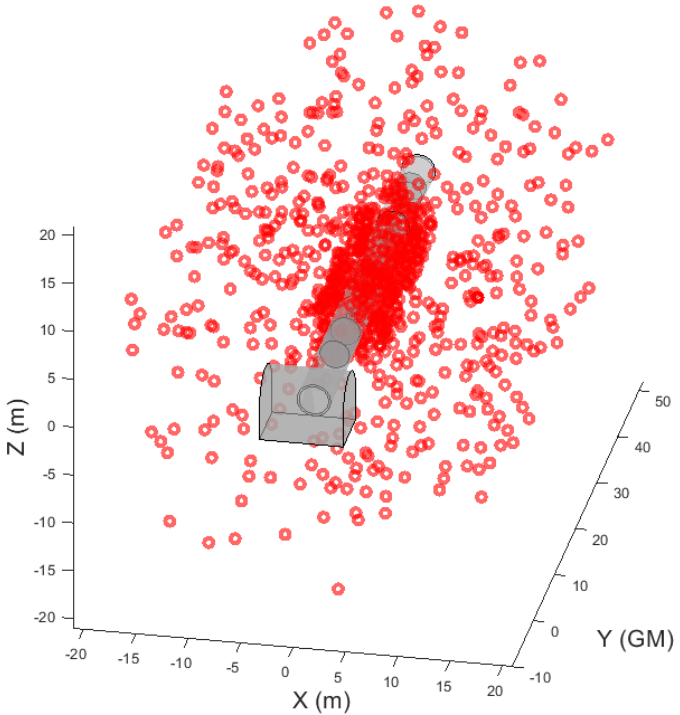


Predicting Temperature of Nuclear Waste Canisters



- Location of sensors
- Temperature
- Pressure
- Humidity

Predicting Temperature of Nuclear Waste Canisters



- Location of sensors
- Temperature
- Pressure
- Humidity

There are **outliers**!!!

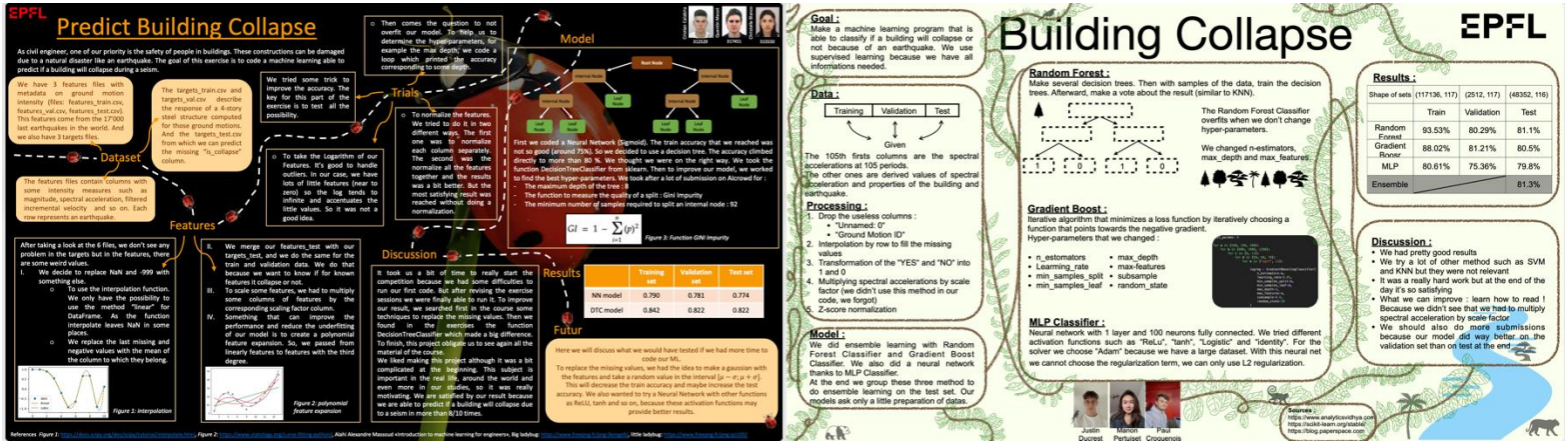
Kaggle

- Make an account (www.kaggle.com)
- Join the competition
- Download dataset
- There will be a leaderboard and a  Prize for the winner!

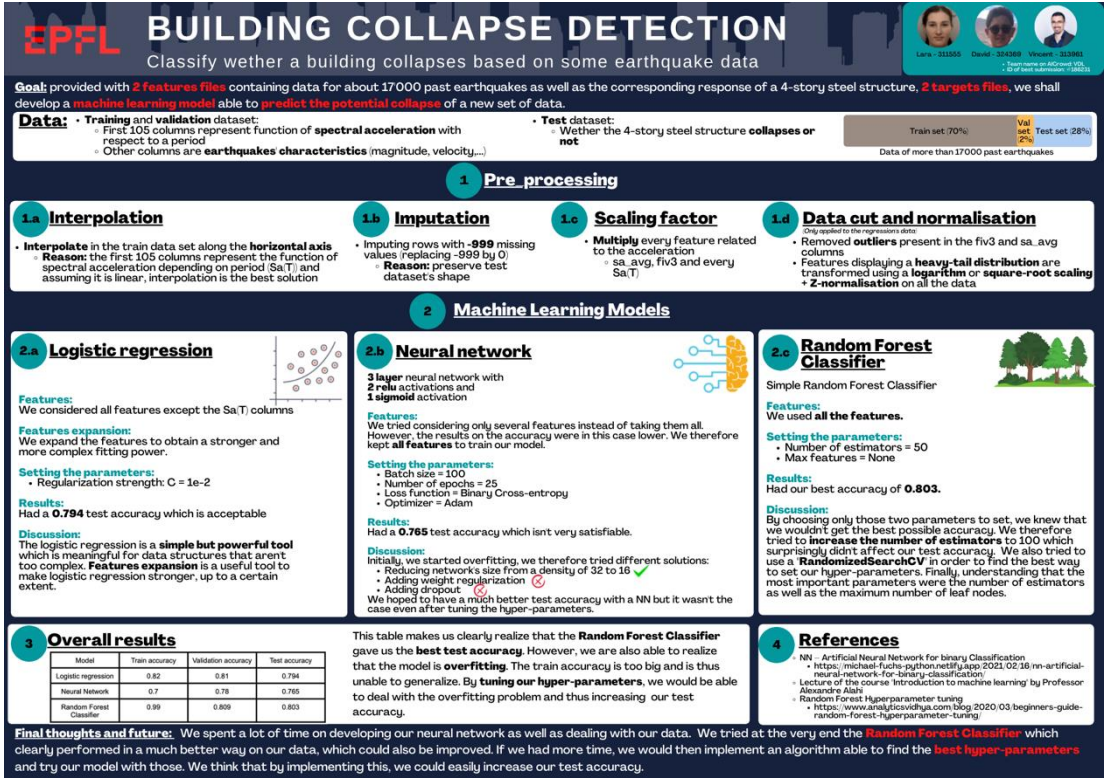
Logistics

- Team of 2 (should be registered on moodle by Tuesday)
- Deadline for the leaderboard is 18th of May.
- Deadline for submission on moodle (poster + code) is 23rd 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



Poster example



EPFL

BUILDING COLLAPSE DETECTION

Classify wether a building collapses based on some earthquake data

Lara - 311555 David - 324369 Vincent - 313961

- Team name on AICrowd: VDL
- ID of best submission: 196216

Goal: provided with **2 features files** containing data for about 17'000 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:
  - Wether the 4-story steel structure **collapses or not**

Train set (70%) | Val set 2% | 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 (Sa(T)) and assuming it is linear, interpolation is the best solution

1.b Imputation
- Imputing rows with **-999** missing values (replacing -999 by 0)
  - **Reason:** preserve test dataset's shape

1.c Scaling factor
- **Multiply** every feature related to the acceleration
  - sa_avg, fiv3 and every Sa(T)

1.d Data cut and normalisation
Only applied to the regression's data
- Removed **outliers** present in the fiv3 and 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 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 satisfiable.

Discussion:
Initially, we started overfitting, we therefore tried different solutions:
- Reducing network's size from a density of 32 to 16 ✓
- 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.

2.c Random Forest Classifier

Simple Random Forest Classifier

Features:
We used **all the features.**

Setting the parameters:
- Number of estimators = 50
- Max features = None

Results:
Had our best accuracy of **0.803.**

Discussion:
By choosing only those two parameters to set, we knew that we wouldn't get the best possible accuracy. We therefore tried to **increase the number of estimators** to 100 which surprisingly didn't affect our test accuracy. We also tried to use a **'RandomizedSearchCV'** in order to find the best way to set our hyper-parameters. Finally, understanding that the most important parameters were the number of estimators as well as the maximum number of leaf nodes.

3 Overall results

Model	Train accuracy	Validation accuracy	Test accuracy
Logistic regression	0.82	0.81	0.794
Neural Network	0.7	0.78	0.765
Random Forest Classifier	0.99	0.809	0.803

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.

4 References

- NN – Artificial Neural Network for binary Classification
  - https://michael-fuchs-python.netlify.app/2021/02/16/nn-artificial-neural-network-for-binary-classification/
- Lecture of the course 'Introduction to machine learning' by Professor Alexandre Alahi
- Random Forest Hyperparameter tuning
  - https://www.analyticsvidhya.com/blog/2020/03/beginners-guide-random-forest-hyperparameter-tuning/

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.