# Machine Learning Algorithms for Predicting Human Fall

Machine Learning (Module 1) – Final Project
ENSIIE, Université Paris Saclay – M2 Finance Quantitative

*Students:*
Pietro Gadaleta
Francesco Stampini

November 12, 2021

*The aim of our study is to apply Machine Learning models and techinques on a given dataset containing anonymous, scaled data related to people's movement on a sensored carpet. Each observation is linked to a binary label, giving information on whether the observed subject has fallen or not.*

*Our goal is therefore to classify these observations, in order to predict if the fall has taken place. For that matter, we compare models by computing their performance scores using a Stratified K-Fold technique; to do that, we first solve the issue of Imbalanced Classes and perform Feature Selection to reduce the dimensionality of the problem.*

## 1 Unbalanced Dataset

Our first matter of interest is to solve the problem of imbalanced classes. Indeed, when looking at the data, we immediately detect that the observations related to a "no fall" event are considerably more numerous than the ones related to a "fall" event (see Figure 1).

This may lead to several problems when dealing with traditional Machine Learning models and, more importantly, it makes the usual performance indicators obsolete; all of this is due to the fact that the majority class is easy to identify and, during the training stage, the model will be able to classify it correctly. On the other hand, the minority class identification will be more prone to errors due to under-representation.

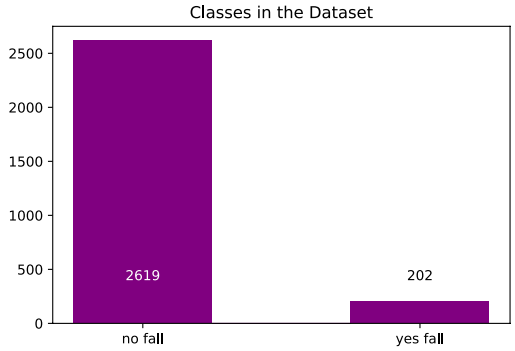


**Figure 1** – Histogram of classes in the dataset, showing the problem of unbalanced classes. The class of fall events is clearly under-represented.

Of course, the interpretation of misclassification errors differs across the two classes. For example, misclassifying a "no-fall" event to a "fall" event is clearly not desired, but undoubtedly less critical than classifying a "fall" event to the "no-fall" class.

For this matter, we resort to the **SMOTE Algorithm** for oversampling. The Synthetic Minority Oversampling Technique consists in generating synthetic observations of the minority class by linear interpolations, more precisely along the segments joining the $k$ nearest neighbors in the minority class. In our case, we use $k = 5$ and proceed with oversampling so that the 'fall' class is exactly as represented as the 'no fall' one.

Literature[1] suggests that this technique works better than *oversampling with replacement*, which apparently does not improve much minority class recognition.
We will also see later how this choice improves overall performance parameters.

## 2 Feature Selection and Dimensionality Reduction

We now focus on selecting the most relevant features among all the available 87. It is in fact crucial to reduce the dimensionality of our problem for several reasons: firstly, a lighter dataset cuts down computational speed, which is necessary – especially if we consider the purpose of this study, i.e. giving a quick prediction on whether a subject has fallen and is in need of help: the faster we know it, the better. Not only that, but we also prevent useless information (as well as noise) from feeding the model by removing it, and we avoid/reduce overfitting.

For this matter, we propose several techniques, all of which we later feed into our selected Machine Learning models for comparison. For each of these techniques, we extract the "most important" five features.
We also point out that we use the original dataset (and not the one obtained using SMOTE) for feature selection, since we know that oversampling the minority class generally violates the independence assumption that is usually made for most of the variable selection methods [2] - we are in fact performing linear interpolation, so necessarily observations will not be independent from each other.

In addition, we recall that we are using the whole dataset for the sake of the explanation. We will be re-performing feature selection in Section 3 on the **training sets only within each fold** of the K-Fold algorithm.

**Feature Selection using Random Forest.** This technique consists in considering the individual features' importances according to how they reduce impurity – on average with respect to all the decision trees that make up the Random Forest. These importances are then compared and ranked. We also try a technique using **Permutation Feature Importance** instead of the impurity measure, i.e. the decrease in a model score when the value of a single feature is randomly shuffled. This technique benefits from being model agnostic and can be computed several times with different permutations of the feature (for our purpose, we picked a number of permutations equal to 4 – increasing this number does not change the outcome). Nevertheless, the two feature selection techniques return the same results.

**Recursive Feature Elimination.** This feature selection algorithm, which we paired to a Decision Tree classifier, reduces model complexity by removing features, one at each iteration, until we reach the amount of features requested, in our case 5. The recursion is needed since, for some importance measures, the relative importance of each feature can vary when it is evaluated over a different subset of features during the stepwise elimination process, in particular when dealing with highly correlated features – as we may observe in this particular dataset.

**Feature Selection using Logistic Regression.** This method takes advantage of 'L1' penalisation to remove redundant features. Lasso regression in fact turns coefficient of redundant features to zero due to regularization. Once we get the coefficients of the trained model, we then sort them and pick the features with the five highest coefficients in absolute value.

**Principal Component Analysis**. We eventually try using PCA, always with the aim of reducing the problem's dimensionality; despite that, we lose the model's interpretability since we are transforming all features. Again, we keep up to 5 components.

Results show the following:

- the RF Feature Selection algorithm returns two raw features and three derivatives as the most important ones;
- the RFE algorithm selects one raw feature, one Fourier transform and three derivatives;

[1] Chawla, Nitesh & Bowyer, Kevin & Hall, Lawrence & Kegelmeyer, W.. (2002). SMOTE: Synthetic Minority Over-sampling Technique. J. Artif. Intell. Res. (JAIR). 16. 321-357. 10.1613/jair.953.

[2] Blagus, R., & Lusa, L. (2013). SMOTE for high-dimensional class-imbalanced data. BMC bioinformatics, 14, 106. https://doi.org/10.1186/1471-2105-14-106

- the Logistic Regression algorithm selects two raw features, one Fourier transform and two derivatives.
- by selecting 5 principal components, the percentage of the total variance explained by PCA is approximately 71%.

We therefore deduce, on a rough basis, that the best type of features according to our feature selection algorithms are the raw ones and the derivatives.

It needs to be pointed out though that each of the Feature Selection techniques returns different features to be the most important, which intuitively may seem wrong: indeed, we would want all feature selection methods to give back the same results.

We find an explanation to this by looking at the correlation of these variables: *Figure 2* shows pairwise scatterplots of all features selected using the Random Forest, Recursive Feature Elimination and Logistic Regression.

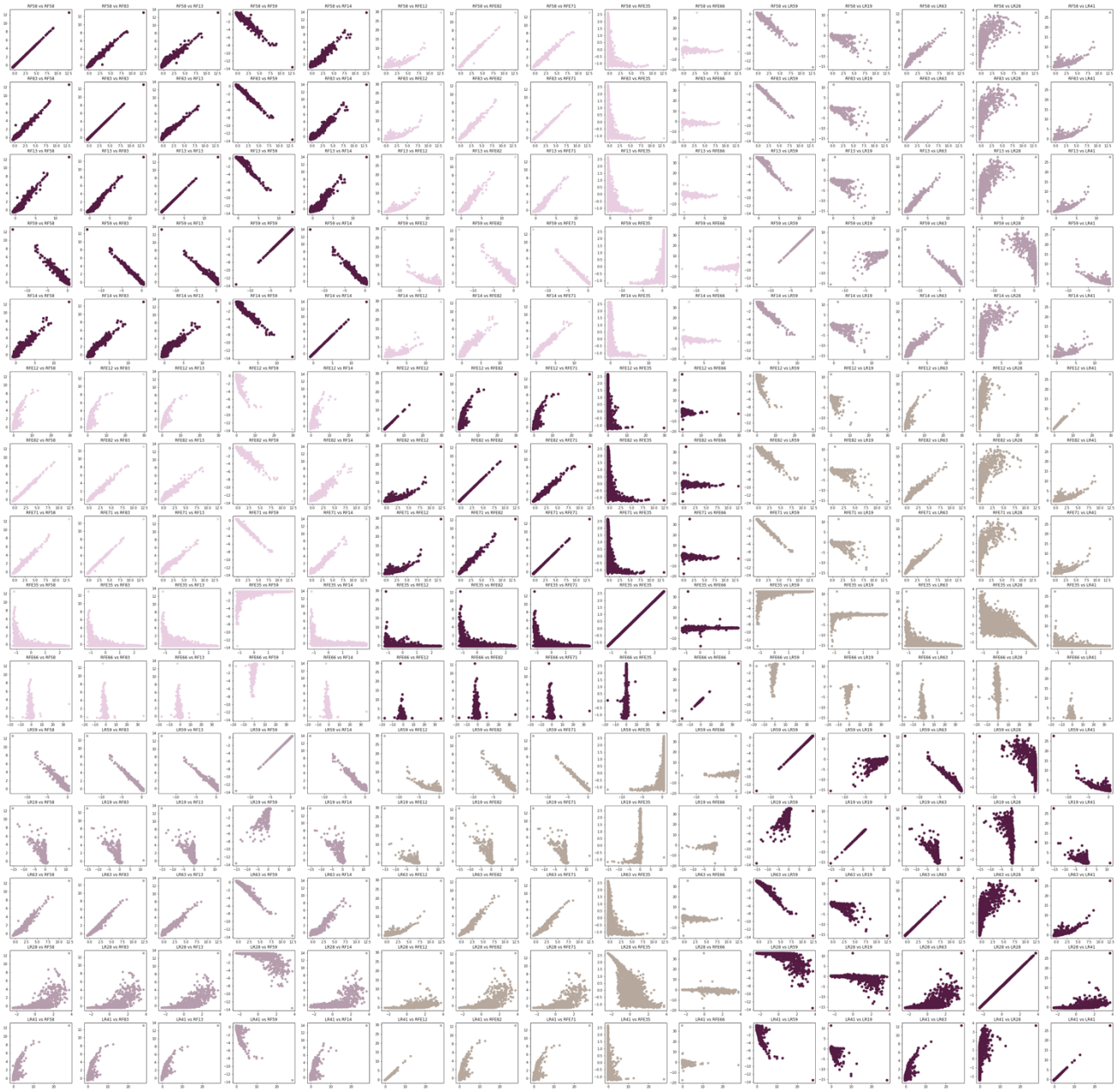

**Figure 2** – Pairwise scatterplot of features selected with Random Forest (1), Recursive Feature Elimination (2), Logistic Regression (3) algorithms. In dark purple, scatterplots of features selected by the same algorithm. In light pink, scatterplots of features from (1) and (2); in lilac, scatterplots of features respectively from (1) and (3); in beige, features selected from (2) and (3) are represented.

It is clear that there is high correlation between features selected using different methods (see plots in light pink, lilac and beige), proving overall collinearity among most of the features, therefore it seems reasonable that different algorithms select different variables as the most relevant: indeed, high collinearity between two features implies that they explain reality pretty similarly.

Moreover, we point out that all features selected using the Random Forest algorithm present high correlation: it will be therefore irrelevant to consider all of them when training the models in the next step.

# 3 Model Comparison

In a second moment, we proceed by choosing five Machine Learning classification models to train using our dataset. We opted for both simple models (Gaussian Naive Bayes, Logistic Regression, Decision Trees) and more complex ones (Random Forest, Extra Trees), in order to compare the quality of their predictive abilities, but also taking into account the computational time taken for these predictions to be made.

To these five models, we fed from 2 to 5 features, according to the importances computed using feature selection algorithms presented in Section 1 **on the training set only** – in the case of PCA, we trained models using from 2 to 5 principal components. The reason we do this is to avoid bias and overfitting by adding information in the test set to help feature selection. To evaluate the model's performance, we use **Stratified K-Fold** cross-validation to consider all data for both training and test sets. We chose the stratified version of the technique (and not the regular K-Fold) to ensure that each fold of the dataset has the same proportion of the observations with a given label, which is ideal when working on a classification problem with imbalanced class distribution, like ours.

Moreover, we implement the SMOTE Technique for oversampling introduced in *Section 1* directly in each fold of the K-Fold, by oversampling the minority class in the training set only; in this way, we do not touch the test set, and obtain results for only real observations, independent on the ones from the training set.

**Feature Selection vs. Choice of Machine Learning Model.** When selecting our features in *Section 2*, we used some techniques based on classifiers; these techniques should actually be used when paired with the same classifier, since they measure feature importance according to the specific model. Despite that, what we will do is to try all models from the pool above, in order to verify whether the model matching to the feature selection algorithm is really the most performing one.

We report results on the test set obtained down below: Table 1 shows the most performing models when considering single performance measures (accuracy, precision, recall, F1, AUC, F2). The F2 score can be defined as:

$$F2 := \frac{5 \cdot \text{Precision} \cdot \text{Recall}}{4 \cdot \text{Precision} + \text{Recall}}$$

and puts more focus on minimizing false negatives than false positives; in other words, it gives more weight to recall. Indeed, what we are the most concerned of is that the highest proportion of falls is classified as such – therefore, F2 is the performance measure we consider the most relevant to make conclusions on models.

Results are generally pretty good for all models. In particular for the features selected using the Random Forest, we observe that performance measures do not vary when adding new features to the same model, this due to the high collinearity of most of the selected features as explained in the previous section.

Accuracy is generally high, but this is a common effect of datasets with imbalanced classes (we recall that the test set has not been oversampled). All other scores – especially recall – actually benefit from using the SMOTE oversampling technique: results lie in a higher range than if we didn't use it. Precision tends to be generally lower than other scores, but this is because it calculates the ratio of true positives among all observations classified as positives: when we deal with a number of 'negative' observations which is significantly bigger than the amount of 'positive' ones, misclassification of the former will have a bigger impact on the precision ratio. Nevertheless, precision values are not too low.

*For all the results and plots, please see the Python Notebook.*

# 4 Conclusion

It now remains to select a model for our classification purposes. We choose to select models with the RFE Feature Selection dataset, since they provide the

| *Using dataset:* | Best Accuracy | Best Precision | Best F1-Score | Best Recall | Best AUC Score | Best F2-Score |
|---|---|---|---|---|---|---|
| **Random Forest Feature Selection** | ExtraTrees (5 features) **0.97944** | ExtraTrees (5 features) **0.81939** | ExtraTrees (5 features) **0.86347** | Gaussian NB (2 features) **0.95049** | Log. Regression (5 features) **0.9604** | Log. Regression (5 features) **0.89742** |
| **Recursive Feature Elimination** | ExtraTrees (5 features) **0.98759** | ExtraTrees (5 features) **0.90739** | ExtraTrees (5 features) **0.91394** | Gaussian NB (2 features) **0.96037** | Log. Regression (3 features) **0.96322** | ExtraTrees (5 features) **0.92026** |
| **Logistic Regression Feature Selection** | ExtraTrees (5 features) **0.98475** | ExtraTrees (5 features) **0.88403** | ExtraTrees (5 features) **0.8912** | Gaussian Naïve (2 features) **0.94537** | Log. Regression (5 features) **0.9593** | Log.Regression (5 features) **0.89958** |
| **PCA** | ExtraTrees (3 features) **0.97802** | Random Forest (5 features) **0.82237** | ExtraTrees (3 features) **0.85683** | Gaussian NB (2 features) **0.99500** | ExtraTrees (2 features) **0.95573** | ExtraTrees (3 features) **0.89003** |

**Table 1** – Best Models when considering single performance measures on the test set after K-Fold, for each reduced dataset; below in bold, the mean value of the measure is reported. Note that results may slightly vary due to randomness.

highest performance scores. It has high computation times, but that is due to the feature selection only; in addition, the method filters correlated features – something other methods do not consider, and maintains the feature interpretability that PCA loses.

Particularly, we focus on recall and the F2-score presented in Section 3, which puts more weight in the recall score but also takes precision into account.

As for the choice of a classifier, all models show excellent recall, yet simple models strongly lack on precision (see Figure 3) – especially Gaussian Naïve Bayes, which generally "sacrifices" precision scores to have a good recall. On the other hand, the ExtraTrees Classifier, as well as the Random Forest, provide good results for the F1, F2, Recall and Precision scores - especially for the latter, which tends to improve as the number of features selected increases.

Therefore, we believe that **an ExtraTrees classifier with 5 features selected using Recursive Feature Elimination** is ideal for making a highly performing classification for our problem.

Finally, we would like to highlight the benefits of using SMOTE for our purposes: without using it, we would have slightly higher accuracies, but poorer performance when it comes to all of the other scores.

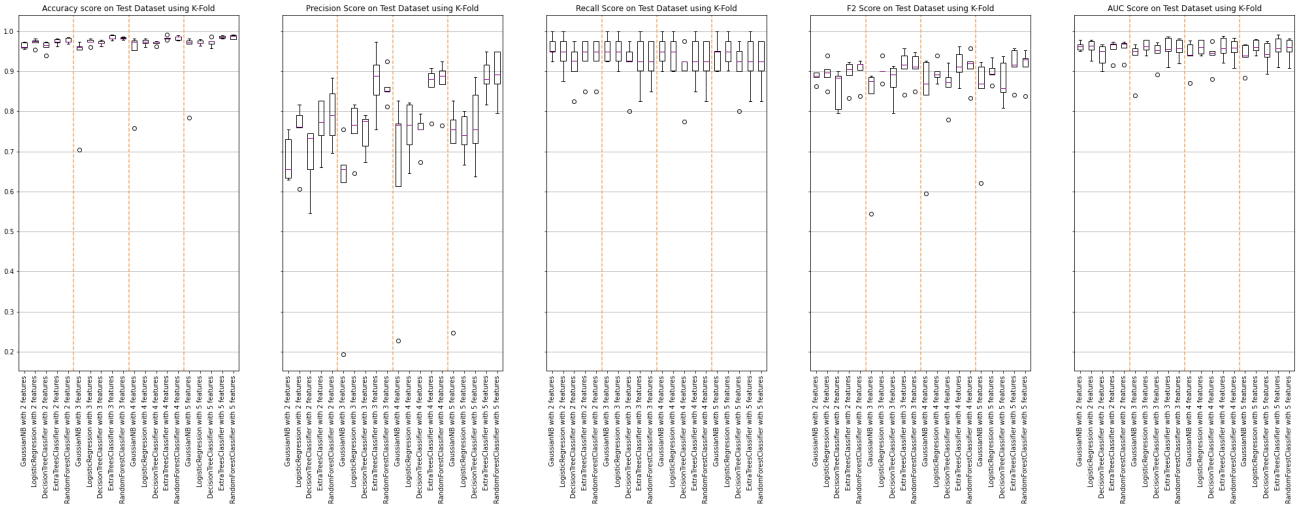


**Figure 3** – Performance scores' boxplots after using K-Fold with features selected using Recursive Feature Elimination for Feature Selection.