Supplementary information

1. Unsupervised and supervised machine learning models: description of the algorithms, evaluation metrics and model interpretability methods

1.1. Unsupervised learning

Unsupervised learning is mainly used for dimensionality reduction (Principal component analysis or partial least squares), or clustering.

Principal Component Analysis (PCA) and Partial Least Squares (PLS) are multivariate statistical methods suitable to describe the relationship between variables and are commonly used to reduce the number of variables. (Godoy, Vega, & Marchetti, 2014; Höskuldsson, 1995). PCA increases interpretability while reducing information loss. The variables in the dataset are reduced to "principal components" that explain most of the data variance. To achieve this, the data need to be standardised beforehand the following equation.

$$z = \frac{x - mean}{standard\ deviation}$$

Then, the variance between the variables and the mean must be understood to understand the relationship between each other. The covariance matrix refers to the covariances related to all pairs of the variables, building a symmetrical matrix. The covariance matrix enables us to observe the correlation between variables either positive (if they increase together) or negative (if one variable increases when the other one decreases). Eigenvectors and eigenvalues of the covariance matrix are computed to calculate the "principal components". The eigenvectors of the covariance matrix determine the direction of the axis where the maximum variance is, and the eigenvalues are the coefficients of the eigenvectors, and they indicate the variance of each principal component (PC).

As mentioned above, the PCs reduce the dimensionality of the data while explaining the maximum amount of the variance. However, the interpretability of the PCs is less than the interpretability of the initial variables. The first PC (PC1) finds the largest possible variance in the data, and the second PC (PC2) will try to find the largest possible variance without being correlated to PC1. Finally, the feature vector is calculated to decide whether to keep all PCs. This is the step when dimensionality reduction happens. Then, the data is reorganised along the PCs axes using the eigenvectors of the covariance matrix (Holland, 2008).

Another unsupervised algorithm is clustering. Clusters can be defined as patterns within a cluster that are more like each other than patterns in another cluster, or as an area with a high density of data points that is separated from another area with a high density of data points (Massart, 2000).

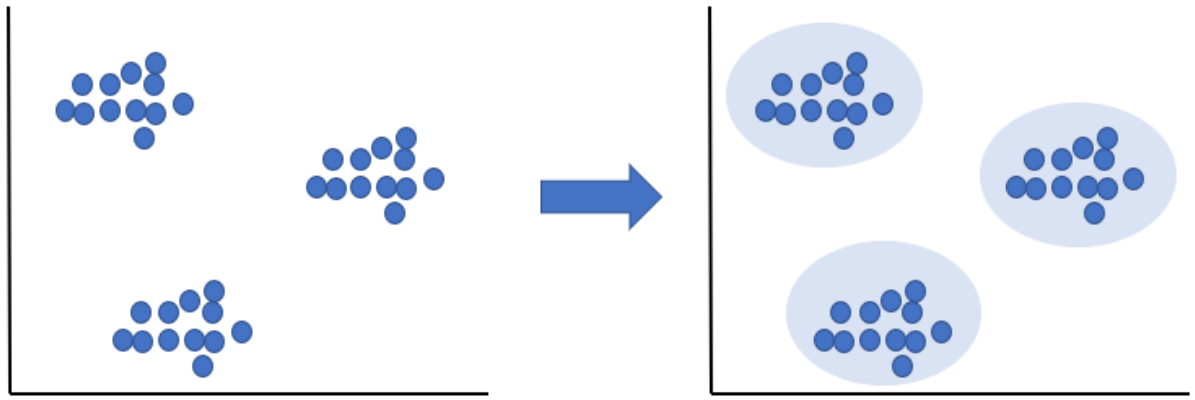

Figure 1: Clustering identifying high-density areas of data points.

Clustering aims to group unlabelled data by finding similarities or differences in the data. The main clustering algorithms are exclusive clustering, overlapping clustering, hierarchical clustering, and probabilistic clustering. Exclusive clustering is based on the idea that one data only belongs to one cluster. K-means clustering is the most typical example of exclusive clustering. It groups the data into a "k" number of clusters by defining the "k" centre of each cluster. Based on the location of the "k" centres, the results may vary. The algorithm calculates the distance between the "k" centres and assigns each data point to its nearest "k" centre. Overlapping clustering, on the other hand, is based on the idea that one data point can belong to more than one cluster to a different degree. Hierarchical clustering is based on the combination between to near clusters. Hierarchical clustering can be either agglomerative or divisive. The difference between these two methods is that agglomerative clusters the data points based on their similarities, whereas decisive clustering groups the data points based on their differences. Finally, probabilistic clustering groups the data points based on their probability to belong to a specific cluster (Grira, Crucianu, & Boujemaa, 2004).

1.2. Supervised learning

Supervised learning is one of the most frequently used tasks and it requires labelled data. In this section, a wide range of classification and regression algorithms used to build predictive models are described in detail.

Classification algorithms aim to categorise the input data. These models can either be binary classification models or multi-class classification models. Among commonly used ML methods this work is mainly focused on k-Nearest Neighbours (kNN), Support Vector Machines (SVM), Random Forest (RF), neural network (NN), Naïve Bayes (NB), Logistic Regression (LR), gradient boosting (GB), and AdaBoost (AB).

kNN is a very used classification algorithm that classifies a new observation based on the class of its neighbour. K represents the number of neighbours that the model uses to classify a new observation, and thus, a 4-Nearest Neighbour model will consider the classes of four neighbours to make the decision. The main question then is, how is "k" decided? Ideally, an odd number will be chosen if there are two classes. Another approach is to calculate the squared root of the number of data points to select "k". Another important parameter is the distance metric. The most widely used distance metrics are Minkowski distance, intended for real-valued vector spaces; Manhattan distance, calculated from the sum of the absolute difference between the Cartesian coordinates of

two data points; and Euclidean distance (default distance), calculated from a straight line between two data points in Euclidean space (Witten & Frank, 2002).

SVM creates a boundary line (hyperplane) to classify data points that belong to different classes in binary classification. The optimum hyperplane is the one that maximises the distances between the two classes (shown in Fig 2).

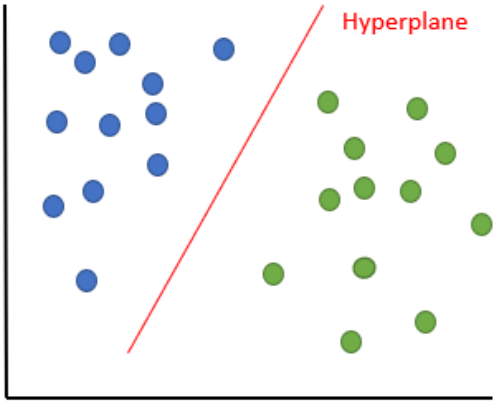

Figure 1: Optimum hyperplane to find the maximum distance between the two classes of data points.

Fig 3 shows a non-linear data set, for which finding the optimum hyperplane is more challenging. For non-linear models, SVM uses kernel functions to find the optimum hyperplane without the need to transform the data to speed up the calculation that otherwise would be needed to transform the data to a higher-dimensional space. The most popular kernel functions are linear kernel, polynomial kernel (usually less efficient), Gaussian Radial Basis (RBF) function, generally selected for non-linear data; sigmoid kernel, mostly used for neural networks, and Gaussian kernel.

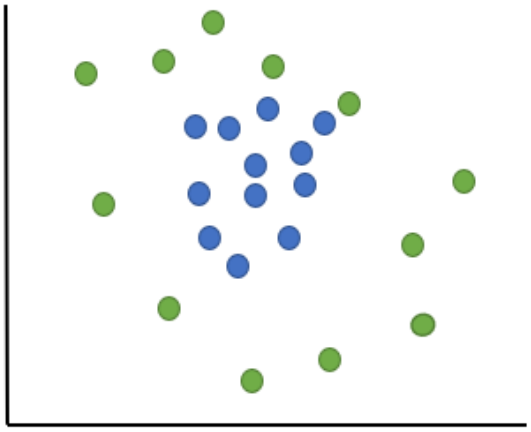

Figure 3: Non-linear dataset for which it is more difficult to find the optimum hyperparameter.

RF was first implemented by Breiman in 2001 (Breiman, 2001), and is applicable to solve both classification and regression tasks. RF is built on decision trees, so to classify a new observation, each tree classifies said observation and the most-voted output is taken as the final model output. The decision trees grow based on the training data available and the number of variables, and therefore,

fewer variables than the total number of variables randomly selected are considered to split each node (Breiman & Cutler, 2016).

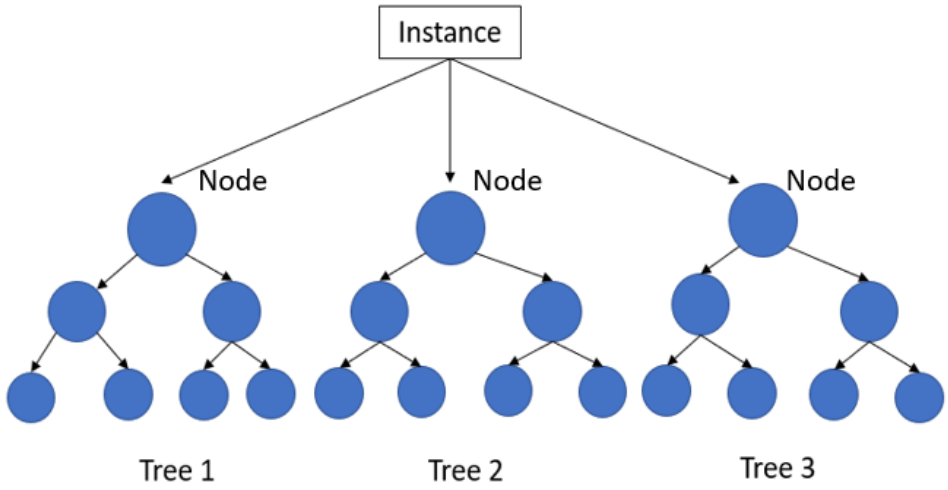

Figure 4: Random Forest decision trees. The length of the decision trees is decided by the availability of training data whereas the nodes are decided based on a randomly selected number of features that provide the best split at each node.

A multi-layer perceptron neural network (NN) is a feedforward artificial neural network, designed for binary classification tasks. It is a linear classifier that uses backpropagation to compute the training gradient, reducing the model error by adjusting the weight and bias. In a multi-layer perceptron, the information goes from the input layer through the hidden layers to the output layer, and the result of the output layer is compared to the true labels (Rosenblatt, 1958). Fig 5 shows that the multi-layer perceptron layers are fully connected. Each node, except for the input layer nodes, has an activation function, generally, a logistic function, that activates the node so the node can pass the information forward (Gardner & Dorling, 1998).

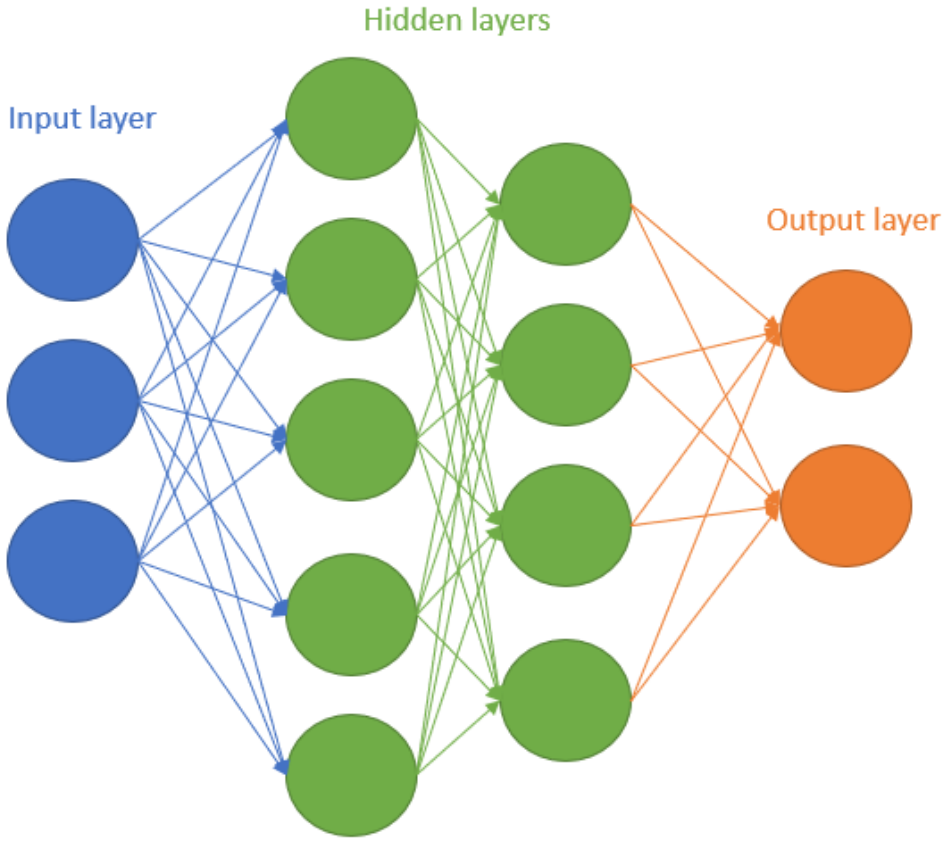

Figure 5: Diagram of the multi-layer perceptron interconnected layers.

NB classification algorithm is built on Bayesian classification methods relying on Bayes' theorem, which is used to calculate the probability of one event after another event had already occurred. It is based on the premise that the variables are independent of one another, and that the data is Gaussian distributed. Bayes' theorem is used in classification tasks to calculate the probability of one observation (*E*) belonging to a class (*E*), calculated using the following equation (Zhang, 2004).

$$P(C|E) = \frac{P(E|C)P(C_c)}{\Sigma_{k=1}^{N} P(E|C_k)P(C_k)}$$

Logistic regression, despite its name, is only used for classification tasks, particularly for binary classification. It is a simple and efficient non-linear algorithm that uses a sigmoid function to classify observations.

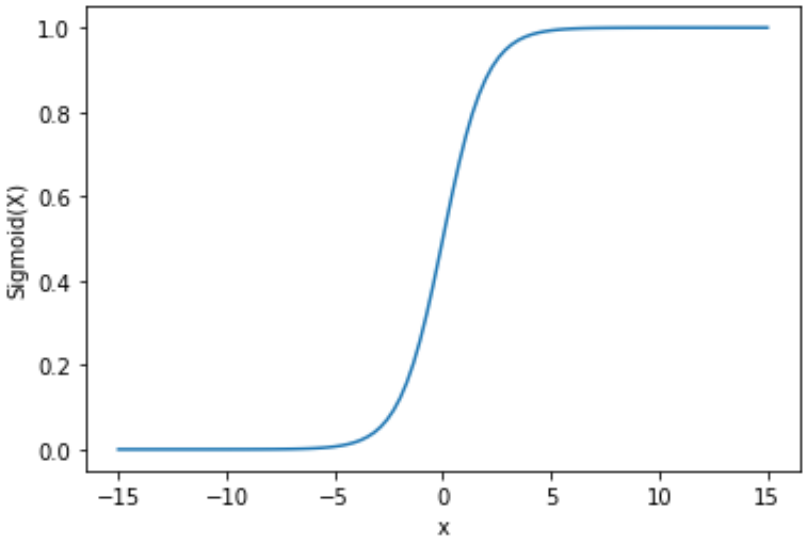

Figure 6: Sigmoid function used in logistic regression to classify observations in class 0 or class 1.

Logistic regression is used in day-to-day examples such as identifying a poisonous mushroom or detecting if an email is a spam or not. To achieve the classification, the algorithm has a threshold to categorise the instances, and it could also be applied to multi-class classification following a "one vs all" approach (Dreiseitl & Ohno-Machado, 2002).

Boosting algorithms assign weights to the training data to identify stronger learning and combine them to build a stronger algorithm. The difference between a strong and a weak learner is that the weak learner can perform better than randomly guessing, but its performance is worse than the performance of a strong learner. The first boosting algorithm implemented was AB (Freund & Schapire, 1996). AB is built on decision stumps (incomplete decision trees than only have one node and two leaves). In the first decision stump, all the weights are the same, but as other stumps are built for all the variables, the weights assign to the data vary: decision stumps that achieve higher accuracy get higher weights (Freund & Schapire, 1996; Kuhn & Johnson, 2013).

Gradient boosting is also an ensemble algorithm that builds a stronger model by combining weak decision trees (decision trees that perform better than randomly guessing but worse than a stronger decision tree). The main difference between GB and RF is that in RF all the decision trees are created at once, whereas in GB, the decision trees are created subsequently. In GB, each new tree is built to better the previous one while minimising the gradient of the loss function.

Scientists have been using regression models before the rise of machine learning (Huang, Ko, Shu, & Hsu, 2020). Though, ML regression models are currently being widely used in different fields such as medical diagnosis or financial forecasting achieving good accuracy. Regression is a method for exploring the relationship between independent variables and a dependent variable. Hence, it can be used as a modelling technique in ML for a continuous response. For this work, random forest (RF) and boosted decision trees algorithms (gradient boosting and AdaBoost) have been used to build regression models. These algorithms have been described in detail in the previous section.

1.3. Classification metrics

To assess the performance of the trained model, it needs to be tested on data. To achieve this, the data is first sampled into training and testing sets. There are several methods to sample the data, being the most common methods random splitting or k-fold cross-validation. For k-fold cross-validation, the dataset is split into "k" number of groups and k-1 groups are used for training and the other fold is used for testing. This process is iterated until the model has been trained and tested on all folds. Once the model is trained, the actual values of the test set, and the predicted values are reported in a confusion matrix to evaluate the model performance. The predicted values are on the columns and the actual values, on the rows. Based on the layout of these values, four parameters that are used to calculate the evaluation metrics are described: True Positives (TP), the instances that were predicted positive and they were actually positive; False Positives (FP), which represents the instances that are negative but were predicted positive; False Negatives (FN), which refers to the instances that were predicted negative but, they are positive; and finally, True Negatives (TN), which are the negative instances that were predicted indeed negative.

Table 1: Confusion matrix.

Confusion matrix

	Actually positive	**Actually negative**
Predicted positive	True positive (TP)	False positive (FP)
Predicted negative	False negative (FN)	True negative (TN)

From the confusion matrix, the evaluation metrics can be calculated. The accuracy is calculated by the ratio of the corrected predicted observations to the total of the observations, as shown in Eq x. Classification accuracy CA) is one of the most used metrics to inform the performance of the model however, it can be misleading when there are data imbalance issues (one class is either underrepresented or overrepresented). This can be explained by the "1:100 class imbalance paradox". In a dataset than contains 100 samples, only one sample belongs to class 0 whereas the rest of them belong to class 1. The CA of this model would be 99%; however, the model would not be accurate at predicting class 0 samples since this class is underrepresented in the training dataset. (Galar, Fernandez, Barrenechea, Bustince, & Herrera, 2011).

$$Classification\ accuracy\ (CA) = \frac{TP+TN}{Total\ of\ observations}$$

To avoid the class imbalance paradox, there are other evaluation metrics that can provide more information about the performance of the model. Precision (Eq x) is calculated by dividing the TP by the FP and the TP, emphasising the importance of the samples that are correctly predicted as true (Sokolova & Lapalme, 2009).

$$Precision = \frac{TP}{FP+TP}$$

Moreover, recall, also known as sensitivity, is calculated by the ratio of the TP to the TP and the FN, highlighting the number of positive samples that were correctly predicted among all the positive samples (Powers, 2020; Sokolova & Lapalme, 2009).

$$Recall\ (or\ sensitivity\ or\ true\ positive\ rate) = \frac{TP}{TP+FN}$$

F-measure, also known as F1 Score, is the weighted average between precision and recall. It takes values from 0 to 1, and the greater the F1 Score value, the better is the performance of our model. Even though this metric might not be the most straight forward parameter, it is generally more useful than classification accuracy, particularly if there is a class imbalance (Powers, 2020).

$$F1 = 2\,x\frac{TP}{TP+\frac{1}{2}(FP+FN)}$$

The receiver operating characteristic (ROC) curve is a well-studied method to assess the ability of the algorithm to distinguish between two classes. This method has been widely applied in other fields such as in drug discovery for virtual screening to select or reject a molecule (Triballeau, Acher, Brabet, Pin, & Bertrand, 2005). AUC – ROC (Area Under Curve Receiver Operating Characteristics) is a performance measurement for classification at different thresholds. ROC is the probability curve and AUC represents the ability to separate between classes. Hence, the greater the AUC value, the better the model is at discriminating between classes. An excellent model would have an AUC – ROC value of 1, whereas a poor model would have a value closer to 0, which means its ability to separate between classes is very low. AUC – ROC is calculated by plotting the True Positive Rate (TPR) or Recall against the False Positive Rate (FPR). TPR or Sensitivity has been described before. FPR is 1 – specificity, which is the calculated dividing the TN by the FP and TN (Powers, 2020).

$$True\ negative\ rate\ (or\ specificity) = \frac{TN}{FP+TN}$$

$$False\ positve\ rate\ (FPR) = 1 - specificity$$

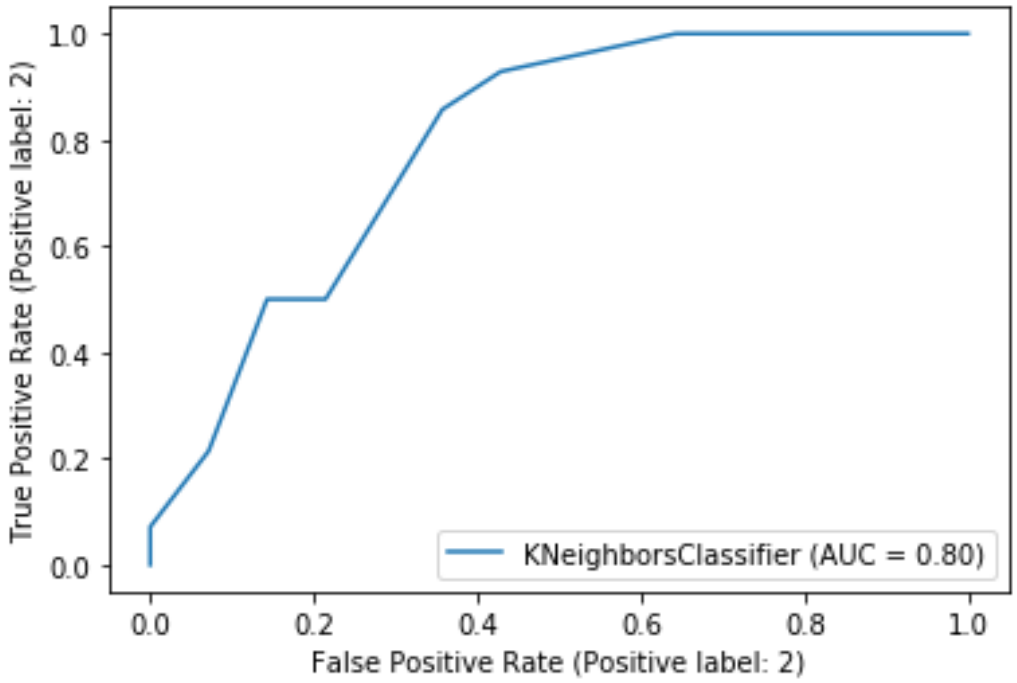

Figure 7: ROC-AUC curve, calculated from the true negative rate plotted against the false positive rate to study the performance of an algorithm.

Figure 7 shows the ROC-AUC of a machine learning model. The AUC value is 0.8, which means that there is an 80% probability for the model to discriminate between classes. This metric is generally used for binary classification, but it can be used for multi-class problems. However, in a multiclass model, the ROC-AUC will indicate the probability of on class to be classified against the other classes.

G-measure is calculated from the geometric mean of precision and recall (Powers, 2020). The geometric mean calculated from the sensitivity and the specificity is explained by Kubat *et al* (1997) in the European conference on machine learning (Heidelberg, Germany) (Kubat, Holte, & Matwin, 1997). They proposed a new metric method to assess the performance of a learning model built on imbalanced training data.

$$Geometric\ measure\ (G-measure) = \sqrt{sensitivity\ x\ specificity}$$

Cohen's Kappa and Matthews Correlation Coefficient (MCC) are generally used for multi-class classification models, and usually they are correlated. However, when there is an imbalance issue, Cohen's Kappa report less reliable results than MCC, i.e., a worse model gets better results in Cohen's Kappa than MCC. Kappa was originally designed to measure the chance agreement between two judges, but this concept has evolved into the present and Kappa is now used in several fields such as neuroscience, machine learning, psychology. When this metric is used in classification to measure the agreement between the actual values and the predicted values, Cohen's kappa is described in the following equation:

$$K = \frac{CA - P_e}{1 - P_e}$$

where P_e is the probability of chance of agreement between predicted and actual values, that is subtracted from the classification accuracy (Delgado & Tibau, 2019).

The main advantage of using Cohen's Kappa as the evaluation method is that it removes the possibility that the model is predicting by guessing randomly. It takes values from -1 to 1, however, the values are not similarly reachable: when the class distribution is balanced, it is easier to get higher values. However, Cohen's Kappa will not give reliable information about the accuracy of the prediction of a single value.

Matthews Correlation Coefficient (MCC) was introduced by Matthews to evaluate the performance of binary classification models (Matthews, 1975) as an evaluation metric that accounts for class imbalance. It was then re-proposed and extended for multi-class classification in 2000 by Baldi *et al* (Baldi, Brunak, Chauvin, Andersen, & Nielsen, 2000). MCC takes values from -1 to 1. Since MCC measures the agreement between actual and predicted values, when MCC is 1, it shows that the predicted and actual values are in perfect agreement, whereas when MCC is -1, the predicted and actual values are in disagreement.

$$MCC = \frac{TP\ x\ TN - FP\ x\ FN}{\sqrt{(TP+FP)(TP+FN)(TN+FP)(TN+FN)}} \quad 1$$

Nonetheless, these evaluation methods only apply for classification models, and therefore different metrics are needed for regression. The most widely used metrics to assess the performance of regression models are: Mean Squared Error (MSE), Root Mean Squared Error (RMSE), Mean Absolute Error (MAE), and Coefficient of determination (R^2). The training is sampled following the same sampling methods as described for classification, and hence, the evaluation metrics stem from the comparison (or the error) between the actual values and the predicted values (Botchkarev, 2019).

The MSE measures is calculated by the squared difference between the predicted and the actual values. the difference between the predicted values and the actual values is squared to ensure a positive number and to punish large errors.

$$MSE = \frac{1}{number\ of\ observations} \sum_{i=1}^{n} (actual - predicted)^2$$

where *n* is the number of observations.

RMSE is an extension of the MSE calculated from the square root of MSE. The main advantage of the RMSE over the MSE is that the RMSE is reported in the same units as the target variable. RMSE removes the squared values from MSE, however, it still punishes large errors.

$$RMSE = \sqrt{MSE} = \sqrt{\frac{\sum_{i=1}^{n} (actual - predicted)^2}{number\ of\ observations}}$$

where *n* is the number of observations.

MAE measures the absolute error between the predictions and the actual values, and it is also reported in the same units as the target variable. the main difference between MAE and RMSE is that the changes in MAE are linear and therefore larger errors have the same importance than smaller errors.

$$MAE = \frac{\sum_{i=1}^{n} |actual - predicted|}{number\ of\ observations}$$

where *n* is the number of observations.

Finally, R^2 (coefficient of determination) studies the proportion of variance in the dependent variable that is predicted from the independent variable. However, it has been advised that R^2 might not be the appropriate measure to evaluate the accuracy of predictive models because it could be misleading (Li, 2017).

Data-driven models have proven their potential applicability in several fields. However, there is a lack of trust that leads to not using data-drive models as much as they could be used. This lack of trust stems from the lack of interpretability ("black box" models). Understanding how a machine learning model makes the prediction and what it can be learnt from it plays a crucial part in the implementation of these models. Therefore, several methods have been proposed to understand complex models, (Ribeiro, Singh, & Guestrin, 2016; Shrikumar, Greenside, Shcherbina, & Kundaje, 2016) and SHAP (Shapley Additive exPlanations) has been used in this project.

SHAP assigns an importance score, also known as SHAP value, to each feature of the training dataset, and by raking the SHAP values, it can be understood how the model is making the predictions. Furthermore, this analysis can be also performed in single observations, improving local interpretability (S. M. Lundberg & Lee, 2017).

The benefits of using SHAP values over other interpretability methods are: global interpretability, SHAP values are used to analyse the feature importance of all variables plus the direction of each variable on the model outcome; local interpretability, SHAP values can be calculated for each individual prediction to understand the variables that contribute towards the prediction and how much these variable contribute; lastly, SHAP values are compatible with a wide range of algorithms (decision-trees-based algorithms, kernel-based algorithms, neural network), for classification and regression, whereas other methods are specific to limited algorithms (S. Lundberg, 2018).

2. List of training materials

- 4-aminobenzoicacid
- Ac-Di-SolSD
- Acetazolamide
- AffinisolHPMC
- Aspirin
- Aspirin(30%)+MC-102
- AvicelPH-102
- BenecelK100M
- Benzoic Acid
- Benzydamine hydrochloride
- Bromhexine
- Caffeine
- Calcium Carbonate
- Calcium Carbonate(20%)-binary
- Calcium Carbonate(20%)-multicomponent

- Calcium Carbonate(40%)-binary
- Calcium Carbonate(40%)-multicomponent
- Calcium Carbonate(5%)-binary
- Calcium Carbonate(5%)-multicomponent
- Calcium Phosphate Dibasic
- Cellulose
- Croscarmellose Na
- D-glucose
- D-mannitol
- Dropropizine
- D-sorbitol
- FastFlo316
- FLOWLAC90
- Granulac140
- Granulac230
- HPMC_2
- Ibuprofen(36%)+Soluplus
- Ibuprofen10%+Povidone
- Ibuprofen30%+Povidone
- Ibuprofen40%+Povidone
- Ibuprofen50%+Povidone
- Ibuprofen50(20%)-binary
- Ibuprofen50(40%)-binary
- Ibuprofen50(40%)-multicomponent
- Ibuprofen50(5%)-binary
- Ibuprofen50(5%)-multicomponent
- Ibuprofen70
- Ibuprofen Sodium Salt
- Lactose
- Lidocaine
- LUBRITOSEAN
- Lubritose Mannitol
- LUBRITOSE MCC
- LUBRITOSE PB
- Lubritose SD
- Magnesium Stearate
- Material1
- Material13
- Material18
- Material19
- Material2
- Material25
- Material26
- Material27
- Material28
- Material29

- Material3
- Material7
- Material8
- Material9
- MC-102+Pearlitol(50%)
- Mefenamic Acid
- Mefenamic Acid(20%)-binary
- Mefenamic Acid(20%)-MC
- Mefenamic Acid(35%)-binary
- Mefenamic Acid(35%)-MC
- Mefenamic Acid(5%)-binary
- Mefenamic Acid(5%)-MC
- MethocelDC2
- MicrocelMC-102
- MicrocelMC-200
- Nimesulide
- Paracetamol Granular Special
- Paracetamol Granular Special(20%)-binary
- Paracetamol Granular Special(20%)-MC
- Paracetamol Granular Special(40%)-binary
- Paracetamol Granular Special(40%)-MC
- Paracetamol Granular Special(5%)-binary
- Paracetamol Granular Special(5%)-MC
- Paracetamol Powder
- Paracetamol Powder(20%)-binary
- Paracetamol Powder(20%)-MC
- Paracetamol Powder(40%)-binary
- Paracetamol Powder(40%)-MC
- Paracetamol Powder(5%)-binary
- Paracetamol Powder(5%)-MC
- Parteck LUBSTA
- Pearlitol200SD-Mannitol
- Pearlitol300DC
- Phenylephrine Hydrochloride
- PluronicF-127
- Potasium chloride
- PVP
- Roquette MgSt
- Roxithromycin
- S-Carboxymethyl-L-cysteine
- Soluplus
- SOLUPLUS20%+METHOCELMC2
- Stearic acid

3. List of materials included in the surface area and surface energy models

- Cellulose

- Avicel PH-101
- PVP
- Pearlitol 200 SD
- Microcel MC-102
- Microcel MC-200
- Pearlitol 100 SD - Mannitol
- Avicel PH-102
- LUBRITOSE MCC
- Ibuprofen 50 (20%) - multicomponent
- Lubritose mannitol
- Ibuprofen 50 (40%) - multicomponent
- Ibuprofen 50 (5%) - multicomponent
- Paracetamol Powder(40%)-MC
- Paracetamol Powder(20%)-MC
- Paracetamol Powder(5%)-MC
- Caffeine
- Ibuprofen 50 (20%) - binary
- Ibuprofen 50
- Ibuprofen 50 (40%) - binary
- LUBRITOSE PB
- Paracetamol Powder
- Pearlitol 300 DC
- Ibuprofen 50 (5%) - binary
- Lubritose SD
- Ibuprofen 70
- Paracetamol Powder(5%)-binary
- Paracetamol Powder(40%)-binary
- Paracetamol Powder(20%)-binary
- aspirin
- Span60
- D-glucose

4. Model performance

		Predicted			
		Cohesive	Easy-flowing	Free-flowing	**Σ**
Actual	Cohesive	13	8	8	**29**
	Easy-flowing	5	14	13	**32**
	Free-flowing	3	8	40	**51**
	Σ	**21**	**30**	**61**	**112**

Figure 1: MLP neural network model confusion matrix for the single-step model evaluated by 10-fold cross-validation (class 1 is cohesive, class 2 is easy-flowing, and class 3 is free-flowing.

		Predicted		
		Non-free-flowing	Free-flowing	Σ
Actual	Non-free-flowing	45	16	**61**
	Free-flowing	17	34	**51**
	Σ	**62**	**50**	**112**

		Predicted		
		Non-free-flowing	Free-flowing	Σ
Actual	Non-free-flowing	3	2	**5**
	Free-flowing	1	2	**3**
	Σ	**3**	**5**	**8**

Figure 2: a) The confusion matrix for the neural network model for Step 1 as evaluated by 10-fold cross-validation; b) External validation for Step 1 of the neural network model. Only 62.5% of the materials were correctly classified.

Code: https://github.com/lpd19/lpd19