UNIVERSITY OF PISA

Department of Information Engineering

DMML Project

A Material Classification Protocol for a Mechatronic Platform

Work Group:
Ivan Brillo
Giovanni Luca Biundo

ACADEMIC YEAR 2024/2025

Contents

1 Introduction

1.1 Problem Description

The accurate classification and characterization of materials based on their mechanical properties represents a fundamental challenge in various engineering and biomedical applications. Traditional material identification methods often rely on visual inspection or destructive testing procedures, which can be time-consuming, costly, and unsuitable for delicate or valuable samples. In particular, the ability to distinguish between materials with similar visual appearance but different mechanical properties is crucial for applications ranging from quality control in manufacturing to medical diagnostics [1].

This project addresses the need for non-destructive, automated material classification by developing a machine learning approach. The classification task involves distinguishing between six distinct material classes, each presenting unique combinations of mechanical characteristics. The challenge lies in extracting discriminative features from heterogeneous sensor data and developing robust classification algorithms that can accurately identify materials under varying experimental conditions and affected by boundary effects.

1.2 Machine Used

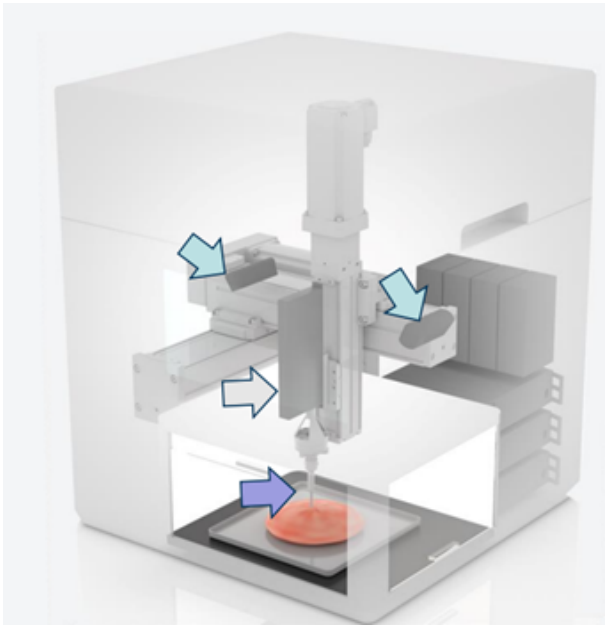


Figure 1.1: Machine platform used to gather the data

The experimental setup employs a mechatronic platform specifically designed for material characterization (see Fig.1.1). The system integrates several key components working in coordinated fashion:

Mechanical Platform: A tri-axial motorized platform provides precise positioning capabilities with millimeter-level accuracy.

Tactile Sensing System: A linear actuator coupled with a high-precision load cell enables controlled force application and measurement. The system can detect contact events and measure normal forces with high sensitivity, providing crucial information about material compliance and surface properties.

Data Acquisition and Control: The entire system is coordinated through a dedicated workstation that manages real-time data collection, platform control, and experimental sequencing.

1.3 Dataset

Material Classes: The classification targets include Dragon Skin 10 (which acts as background material), Dragon Skin 20, Dragon Skin 30, Sorta Clear, PDMS (polydimethylsiloxane), and Econ 80. These materials span a range of mechanical properties providing diverse classification challenges.
Sample Geometry: All non-background test samples maintain consistent geometric parameters with a standardized diameter of 10 mm, ensuring comparability across different materials. The samples exhibit varying curvature characteristics ranging from 0.006 mm to 0.04 mm, with corresponding radii spanning from 5 mm to 13 mm, reflecting the natural material properties and manufacturing variations (Table 1.1)
.

	1	**2**	**3**	**4**	**5**
Radius [mm]	13	7,25	5,665	5,125	5
Curvature [mm]	0,006	0,019	0,031	0,038	0,04
Diameter [mm]	10	10	10	10	10

Table 1.1: Geometric properties of materials

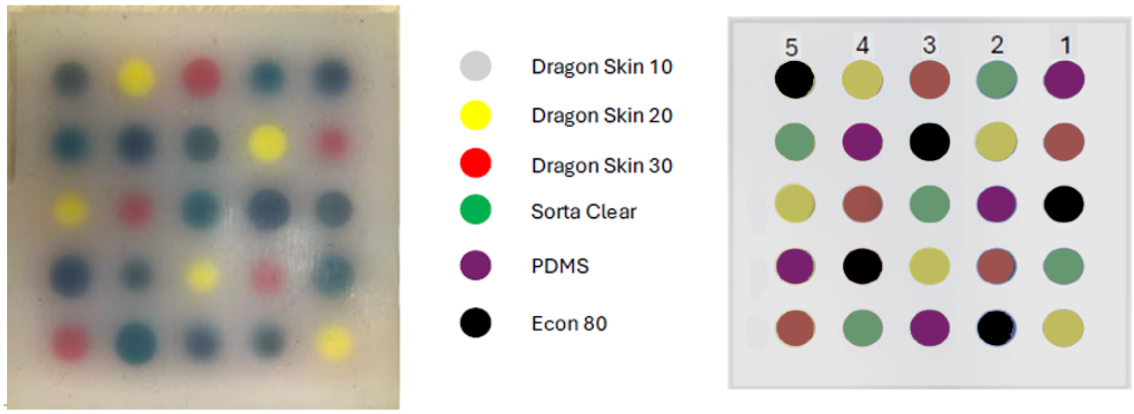
Data Structure and Format: The experimental data is organized in structured CSV files with standardized naming conventions incorporating timestamps, experiment identifiers, and sequence numbers. Tactile and positional data are recorded with 16-bit integer precision. The dataset includes metadata such as platform status indicators, contact detection flags, and temporal synchronization markers.

Variable	**Description**	**Unit**
isReady_Festo	1 if platform is operative, 0 otherwise	bit
isArrived_Festo	1 if platform is in the set position, 0 otherwise	bit
isTouching_SMAC	1 if linear actuator is touching the sample, 0 otherwise	bit
isMaxStroke_SMAC	1 if linear actuator exceeded its travel range, 0 otherwise	bit
posx	x-axis position (integer part)	mm
posy	y-axis position (integer part)	mm
posz	z-axis position (integer part)	mm
posz2	voice coil z-axis position	a.u.
isHome_SMAC	1 if platform is in position (0,0,0), 0 otherwise	bit
state_Festo	state id of the PLC state machine	number
Fz	normal force	a.u.
posx_d	x-axis position (fractional part)	mm
posy_d	y-axis position (fractional part)	mm
posz_d	z-axis position (fractional part)	mm
posz2	voice coil z-axis position	a.u.
CPXEts	timestamp of data collection	seconds

Table 1.2: Raw Data organization

Samples Analyzed: Data collection was performed on designated objects called Damasconi. Two measures spanned in time were performed on DamasconeA (used for training) and another on DamasconeB (used for testing), ensuring statistical robustness and enabling cross-validation studies (Fig. 1.2).

(a) Damascone A



(b) Damascone B

Figure 1.2: Damascone A and B with materials labels and geometric properties labeling

2 Features Extraction

This chapter describes the comprehensive feature extraction and signal processing methods employed for tactile material classification. The approach combines temporal, spatial, and statistical features extracted from force and position signals acquired during material probing experiments. The methodology encompasses signal preprocessing, temporal feature extraction, spatial analysis, and curve characterization techniques.

2.1 Signal Pre-processing

2.1.1 Sensor Data to Physical Values

Raw sensor data need to be transformed in order to get feasible physical values:

- posx + posx_d/1000 millimeters, is position X
- posy + posy_d/1000 millimeters, is position Y
- posz + posz_d/1000 + posz2/200 millimeters, is position Z
- $F_z/27648$ (if $|F_z| < 27648$) Newton, is Force along Z
- $F_z/32767$ (otherwise)

2.1.2 Gaussian Smoothing

Raw force (F_z) and position (pos_z) signals are smoothed using Gaussian filtering to reduce noise while preserving essential signal characteristics:

$$F_{z,s} = \mathcal{G}_\sigma(F_z), \quad pos_{z,s} = \mathcal{G}_\sigma(pos_z) \tag{2.1}$$

where $\mathcal{G}_\sigma$ represents a Gaussian filter with standard deviation $\sigma = 2$.

2.1.3 Signal Normalization

Both force and position signals undergo normalization to ensure comparability across different measurements:

$$F_{z,norm} = \frac{F_{z,s} - \mu_{F,arrived}}{\mu_{F,touching}} \tag{2.2}$$

$$pos_{z,norm} = \frac{pos_{z,s} - \mu_{pos,arrived}}{\mu_{pos,touching}} \tag{2.3}$$

where $\mu_{F,arrived}$ and $\mu_{pos,arrived}$ represent the mean values during the arrival phase, and $\mu_{F,touching}$ and $\mu_{pos,touching}$ are the mean values during the final touching phase.

2.1.4 Temporal Interpolation

Signals (force and position) are interpolated to achieve uniform temporal sampling at 1 kHz using cubic spline interpolation:

$$S_i(t) = \text{CubicSpline}(t_{original}, S_{original})(t_{uniform}) \tag{2.4}$$

Resulting signals can be seen in Fig 2.1.

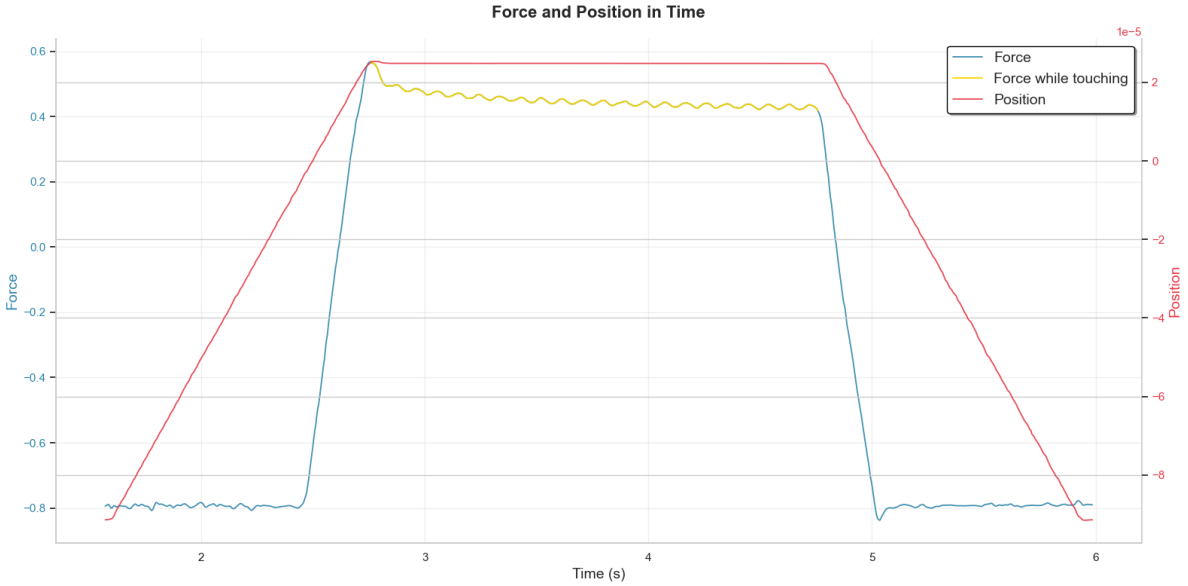


Figure 2.1: Force and position plot in time

2.2 Temporal Feature Extraction

2.2.1 Mechanical Properties

Stiffness

Material stiffness is computed as the slope of the force-displacement relationship during the loading phase:

$$k = \frac{F_{max} - F_{threshold}}{pos_{max} - pos_{threshold}} \tag{2.5}$$

where $F_{threshold} = 0.1 \cdot F_{max}$ represents the low force threshold. Note that $pos_{max} = pos|_{F=F_{max}}$. The same notation is used for the following equations.

Upstroke and Downstroke Characteristics

The upstroke stiffness characterizes the material response during loading:

$$k_{up} = \frac{F_{mid} - F_{low}}{pos_{mid} - pos_{low}} \tag{2.6}$$

where $F_{mid} = 0.5 \cdot F_{max}$ and $F_{low} = 0.1 \cdot F_{max}$.
The downstroke stiffness captures unloading behavior:

$$k_{down} = \frac{F_{high} - F_{end}}{pos_{high} - pos_{end}} \tag{2.7}$$

where $F_{high} = 0.75 \cdot F_{max}$.

Damping Coefficient

The damping coefficient is estimated through least-squares fitting of the force-velocity relationship:

$$F(t) = k \cdot pos(t) + c \cdot \dot{pos}(t) \tag{2.8}$$

The result of the fitting are (k, c) coefficients, where c represents the damping coefficient obtained from the linear regression.

2.2.2 Viscoelastic Properties

Force Decay Time Constant

The viscoelastic relaxation is characterized by fitting an exponential decay model:

$$\ln(F(t)) = \ln(F_0) - \frac{t}{\tau} \tag{2.9}$$

where τ is the time constant obtained from the slope of the linear regression in semi-logarithmic space.

Force Overshoot and Relaxation

Force overshoot quantifies the transient response:

$$\text{Overshoot} = \frac{F_{max} - F_{ss}}{F_{ss}} \tag{2.10}$$

Where F_{ss} represents the force at steady-state after the indentation transient (represent the force while the probe is touching the material $\mu_{F,touching}$). The position for which the steady-state condition is met is also saved as feature (P_{ss}).
Force relaxation measures the decay from peak to final-state:

$$\text{Relaxation} = \frac{F_{max} - \overline{F_{final}}}{F_{max}} \tag{2.11}$$

where $\overline{F_{final}}$ is the mean of the last 50 samples.

2.2.3 Contact Mechanics

Contact Area Estimation

The contact area is estimated using Hertzian contact theory:

$$A_{contact} = -\pi \left(\frac{3(1-\nu^2) F_{max} R}{4k} \right)^{2/3} \tag{2.12}$$

where $R = 5$ mm is the probe radius, $\nu = 0.5$ is the Poisson's ratio, and the negative sign ensures proper format for subsequent processing. Here we used the contact stiffness extracted in place of the elastic modulus in the Hertzian model. This substitution is justified because stiffness was obtained from the linear region of the force displacement curve. The parameters R and ν were chosen arbitrarily but consistently applied across the entire dataset to maintain uniformity in the analysis. This approach provides a practical means to incorporate geometric and material assumptions while relying on experimental data.

2.2.4 Statistical and Information-Theoretic Features

Signal Entropy

Shannon entropy quantifies the randomness in the force signal:

$$H = -\sum_{i=1}^{N} p_i \log_2(p_i) \tag{2.13}$$

where p_i represents the probability of the i-th bin in the force histogram.

Sample Entropy

Sample entropy measures signal regularity and complexity:

$$SampEn(m, r, N) = -\ln\left(\frac{A^m(r)}{B^m(r)}\right) \tag{2.14}$$

where $A^m(r)$ and $B^m(r)$ are pattern matching counters for embedded dimension $m = 2$ and tolerance $r = 0.2$.

Hjorth Parameters

Activity and mobility parameters characterize signal variance and spectral properties:

$$\text{Activity} = \text{Var}(F) \tag{2.15}$$

$$\text{Mobility} = \sqrt{\frac{\text{Var}(\dot{F})}{\text{Var}(F)}} \tag{2.16}$$

Teager-Kaiser Energy Operator

The TKEO captures instantaneous energy variations:

$$\text{TKEO}[F(n)] = F^2(n) - F(n-1) \cdot F(n+1) \tag{2.17}$$

From which we extracted:

$$TKEO_{mean} = E[TKEO] \tag{2.18}$$

2.2.5 Signal Metrics

Root Mean Square

Force RMS quantifies the overall signal magnitude:

$$F_{RMS} = \sqrt{\frac{1}{N}\sum_{i=1}^{N} F_i^2} \tag{2.19}$$

Peak-to-Peak Amplitude

$$F_{p2p} = \max(F) - \min(F) \tag{2.20}$$

2.2.6 Peak Ratio

$$Peak_{ratio} = \frac{Max(F)}{Max(pos)} \tag{2.21}$$

Force-Position Correlation

$$\rho_{F,pos} = \frac{\mathrm{Cov}(F, pos)}{\sigma_F \sigma_{pos}} \tag{2.22}$$

2.3 Curve Shape Analysis

2.3.1 Higher-Order Moments

Loading phase skewness and kurtosis characterize curve asymmetry and tail behavior:

$$\text{Skewness} = \frac{E[(F - \mu_F)^3]}{\sigma_F^3} \tag{2.23}$$

$$\text{Kurtosis} = \frac{E[(F - \mu_F)^4]}{\sigma_F^4} \tag{2.24}$$

2.3.2 Polynomial Curve Fitting

Quartic polynomial coefficients capture nonlinear loading characteristics:

$$F(pos) = a_4 pos^4 + a_3 pos^3 + a_2 pos^2 + a_1 pos + a_0 \quad \text{(quartic)} \tag{2.25}$$

The leading coefficient a_4 is retained as feature representing the degree of nonlinearity.

3 Dataset Preprocessing

3.1 NaN Handling

First, we proceed by handling NaN values. In particular, at some sporadic points (at most 5 per acquisition experiment), the machine will fail and not sense the material. In order to overcome the issue of having a data grid with some holes, which could interfere with further smoothing, we used a *NearestNDInterpolator* to fill those gaps. This interpolator finds the closest point and uses it to cover the grid hole. The corresponding label for those newly introduced points is the median of its closest 4 neighbors.

3.2 Spatial Smoothing - Median

After the missing data imputation, we proceed with data smoothing. In particular, we perform feature-wise spatial smoothing over a $2D$ grid (x, y). It interpolates each feature's values by applying a local median filter across a 3×3 neighboring square (3.1).
We decided to keep the neighboring area as small as possible in order to allow the machinery to perform real-time prediction without needing to look at many neighboring points. We also decided to use a median filter since it is more suitable for removing outliers than a mean filter.
We handled the border points, which do not have a complete 3×3 square around them, by reflecting the internal grid points. The results for the *Upstroke* feature can be seen in Figure 3.2.

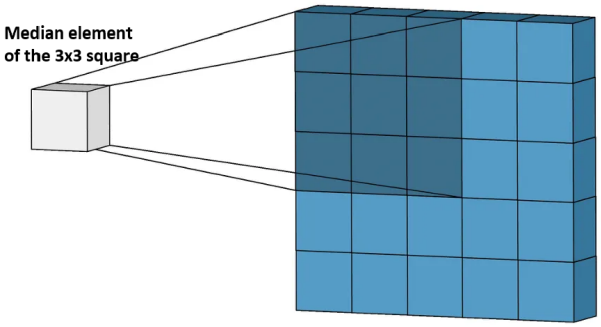


Figure 3.1: Median Spatial Smoothing Visualization

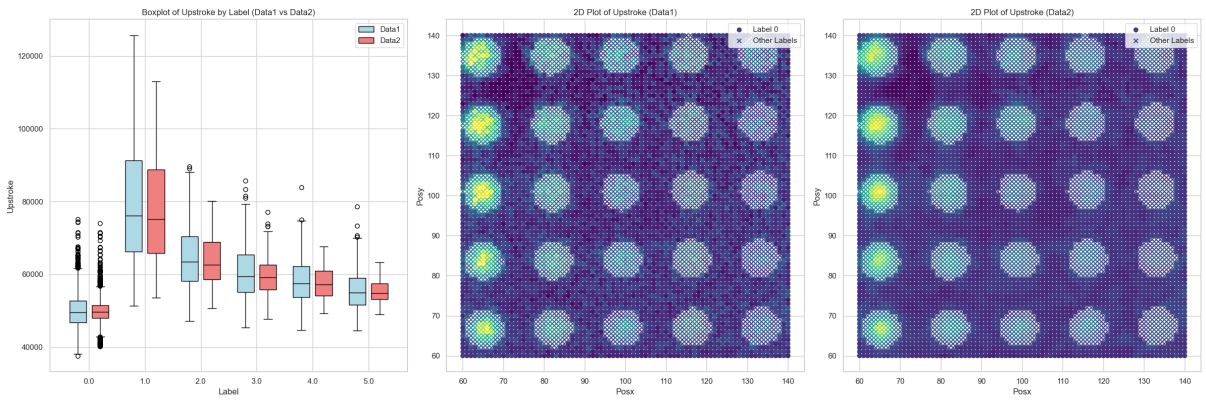


Figure 3.2: Upstroke Median Spatial Smoothing Results (Data1 and Data2 represent the non-smoothed version and the smoothed version respectively)

3.3 DB Splitting and Imbalance

The dataset is split into train, validation, and test sets using three experiments: two measurements on the same object (which will be train and validation) and one on another object. The 6 classes are highly imbalanced, since the background is common while the materials are more sporadic (3.4).

3.3.1 ADASYN for Handling Dataset Imbalance

We used the ADASYN tool to oversample the minority classes. This tool acts in a similar manner to SMOTE by looking at same-class neighboring points and generating similar ones (3.3). ADASYN in particular does not choose points randomly to perform the data synthesis, but instead favors points in low class density regions in order to generate points that are near the class decision boundaries, forcing the classifier to focus on those difficult-to-learn examples [2].

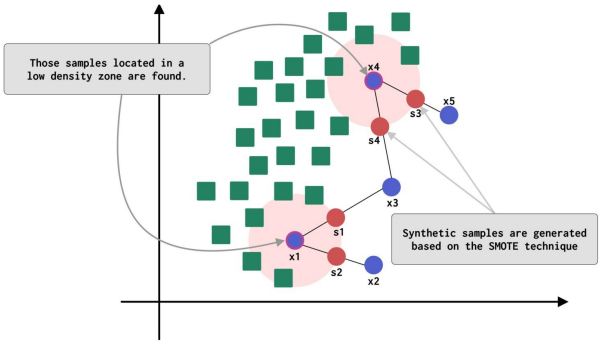


Figure 3.3: ADASYN points generation

3.4 Scaling

The benefits of scaling are numerous. For example, providing models with features having similar distributions as input could help avoid biases among them. The main reason for using independent z-scaling among each data acquisition experiment is to flatten particular drifts that could arise for different physical reasons, such as the warming of the sensor (3.5).
In particular, for the point $\mathbf{x_i}$, we scale feature j in the following way:

$$z_i^j = \frac{x_i^j - \mu^j}{\sigma^j}$$

Mean and variance are calculated independently for train, validation, and test sets.

3.5 Spatial Smoothing - Maximum

We first train a binary classifier to discern between a background point ($y = 0$) and any other class ($y \neq 0$). The second stage will train a multi-class classifier to discern the different materials.
To avoid border effects, whereby features calculated on the border of one hard object tend to dissolve towards the background, thus decreasing the overall accuracy of the model, we decided to provide a second, in-between smoothing. The concept is similar to the median filter explained before, but this time for each 3×3 square we take the maximum value in order to avoid border effects (3.6).
For the training phase, this filtering is applied looking at points with $y_i \neq 0$ (true labels), while in the inference phase, the points that are considered are those output by the binary model ($\hat{y}_i \neq 0$).

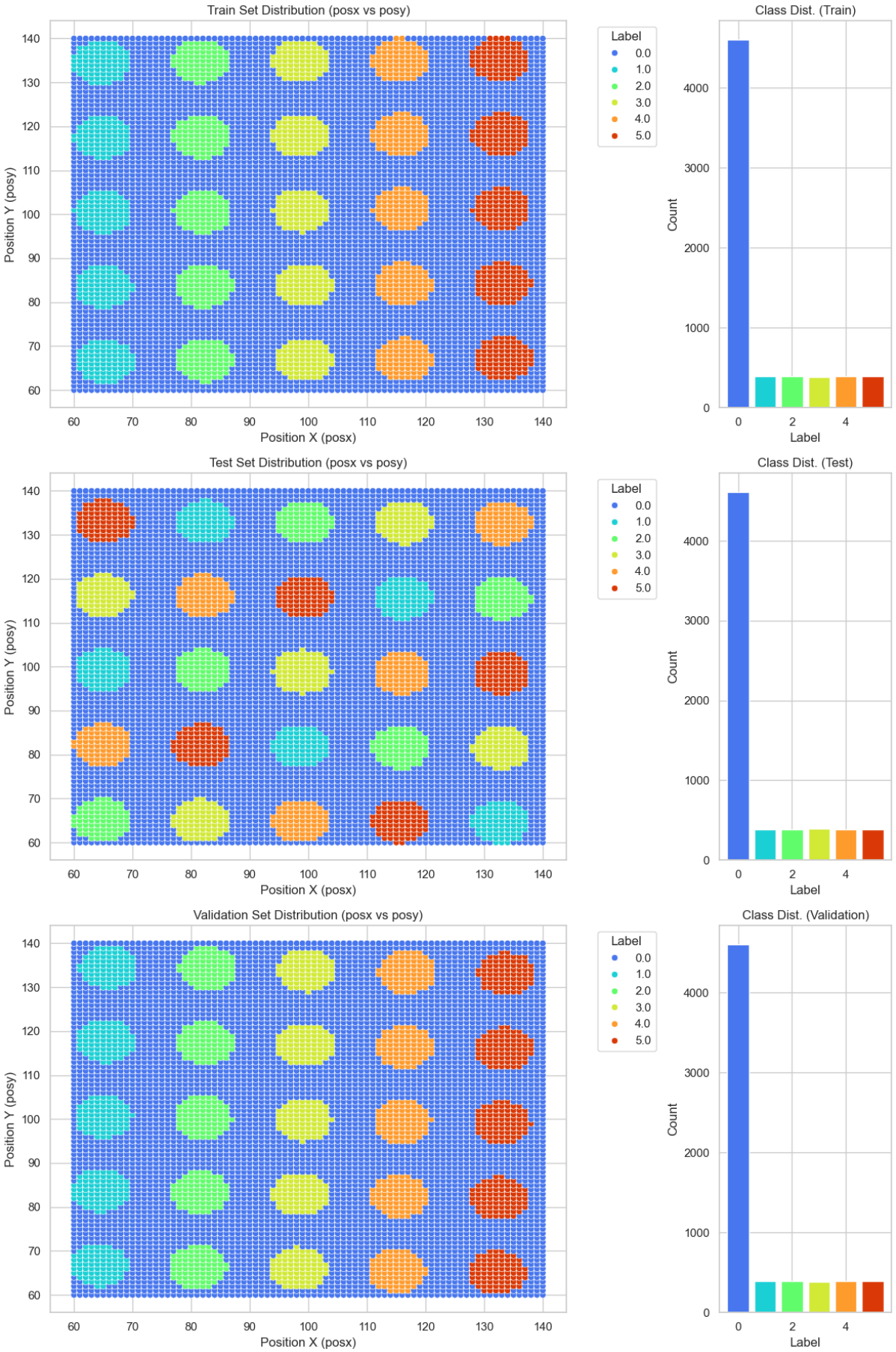
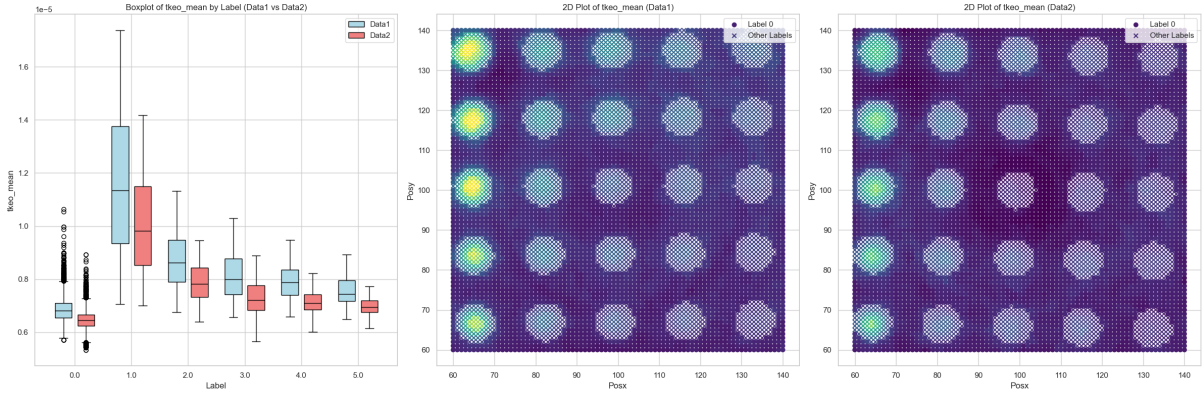
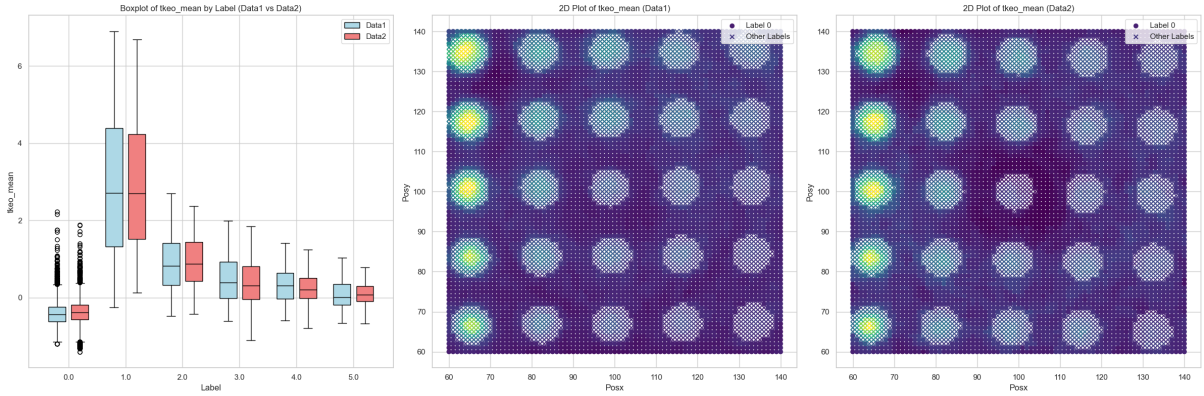


Figure 3.4: Class distributions among different objects



(a) Train and validation dataset before scaling



(b) Train and validation dataset after scaling

Figure 3.5: Effect of scaling on the *tkeo_mean* feature for train and validation datasets

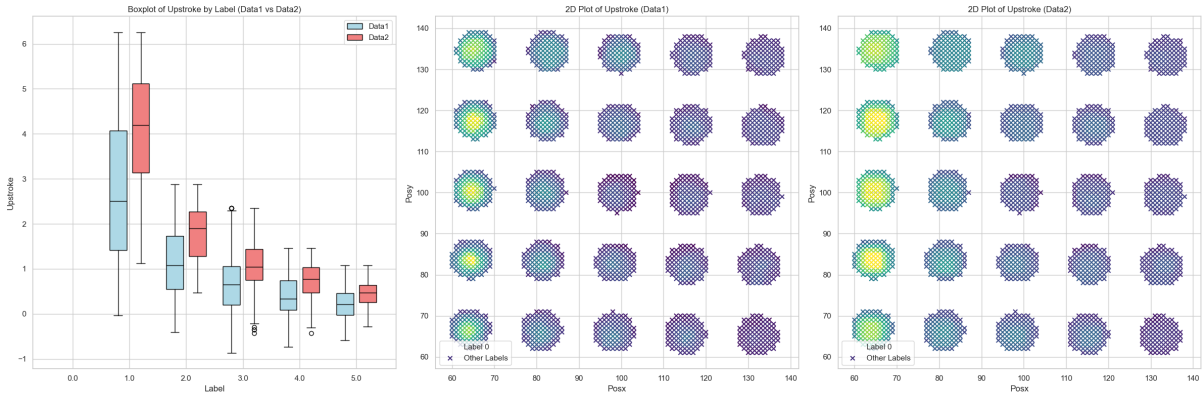


Figure 3.6: Effect of hardspots maximum smoothing for the *Upstroke* feature in validation

4 Feature Selection

4.1 MI Score

First, we compute the mutual information for each feature against the ground truth for training and validation. In particular, for the multiclass case, the formula is:

$$I(X;Y) = \sum_{x \in \mathcal{X}} \sum_{y \in \mathcal{Y}} p(x,y) \log \left(\frac{p(x,y)}{p(x)\, p(y)} \right)$$

where $\mathcal{X}$ is the set of points projected to a single feature for training or validation, while $\mathcal{Y} = \{0, 1, 2, 3, 4, 5\}$ is the label set. The results can be visualized in the image below (4.1).

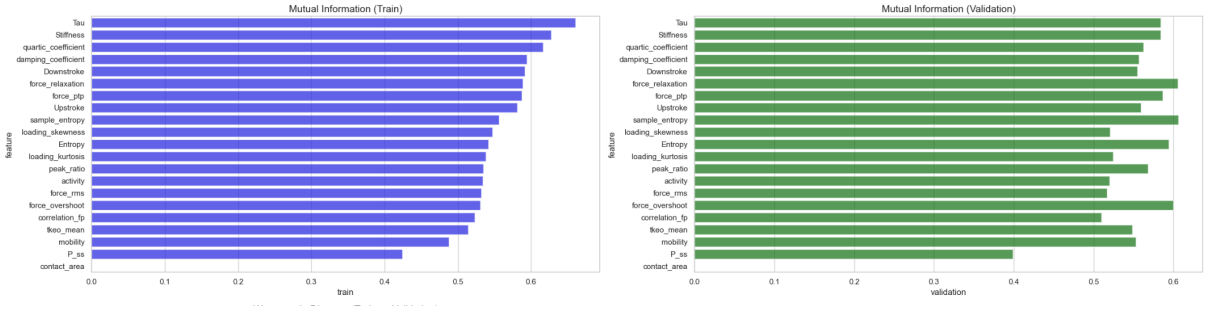


Figure 4.1: Mutual Information (MI) of training and validation dataset with labels

4.2 Global Wasserstein Distance

Different experiments are subjected to some degree of distortion of the collected features due to various factors, such as the temperature of the sensor. Our first concern was to find features invariant among experiments. To do so, we used the Wasserstein distance among probability distributions. In particular, given two PDFs—one for training points and the other for validation points projected onto a single feature—their distance is defined as the minimum cost to transform one into the other. The cost is the summation of the different probability masses times the distance they need to be moved in order to make one distribution equal to the other. The computation results of this metric for each feature are shown below (4.2).

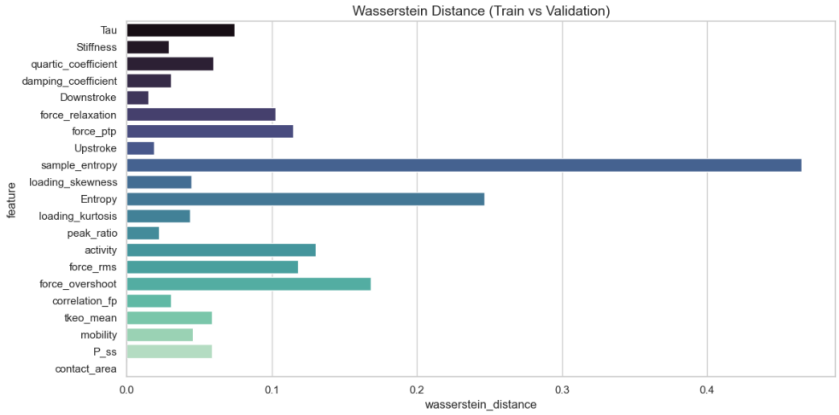


Figure 4.2: Global Wasserstein distance for each feature

4.3 Per-Class Wasserstein Distance

After computing the global Wasserstein distance, we also proceeded to compute this metric for each (feature, class) pair. The computation results for the *Upstroke* feature are shown below (4.3).

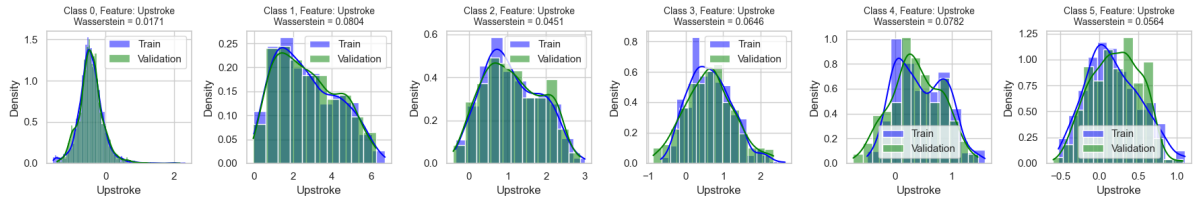


Figure 4.3: Per-class Wasserstein distance of *Upstroke* feature

4.4 Silhouette Score

For some groups of features, as we will explain in the next section, we compute the silhouette score, considering as clusters the points projected to the features set belonging to the same label. This could help us understand which group of features is better clustered, hoping they will provide better classification results, since the classes would be better spatially separated. We used the fact that the different features seem to follow a Gaussian distribution, thereby forming convex clusters, in order to interpret the silhouette score meaningfully.
The Silhouette score is defined as follows:

$$s(i) = \frac{b(i) - a(i)}{\max\{a(i), b(i)\}}$$

where:

- $a(i)$ is the average distance between point i and all other points in the same cluster (same label)
- $b(i)$ is the minimum average distance from point i to all points in any other cluster, i.e., the nearest cluster to which i does not belong
- $s(i) \in [-1, +1]$, with higher values representing more clustered datasets

4.5 Final Feature Selection Pipeline

Even though we calculated the MI scores for each feature, we did not use them in this phase, since they are almost all informative. We focused on the Wasserstein distance values. In particular, we fixed two moving thresholds for the two metrics. For the global Wasserstein distance, we chose values $\in [0.025, 0.15]$ with $step = 0.025$, while for the per-class distance values $\in [0.025, 0.2]$ with the same step.
For each of those threshold pairs $(w\_dist_i, w\_pc\_dist_j)$, we computed the features that have lower distances. In order to reduce the number of possible feature sets, we computed the silhouette score for sets with the same number of features and kept only the one with the highest metric. Furthermore, we discarded every feature set that had a negative silhouette coefficient.
We also exploited the fact that with preliminary models, the classes $\in \{0, 1\}$ had really high f1-scores, so the two distances were calculated only on a subset of points—those with labels $\in \{2, 3, 4, 5\}$. By doing so, we forced the process to pay more attention to those difficult classes with similar values.
The feature sets that we found are:

features	**silhouette**
['Stiffness', 'Upstroke', 'Downstroke']	0.21262237
['Stiffness', 'Upstroke', 'Downstroke', 'tkeo_mean']	0.19775134
['Stiffness', 'Upstroke', 'Downstroke', 'damping_coefficient', 'mobility', 'tkeo_mean']	0.24100173

Table 4.1: Feature sets and their corresponding silhouette scores

5 Model Selection and Hyperparameter Tuning

5.1 Ensemble Classifier

In order to achieve the best results, we divided the classification task into two parts. The first is to distinguish background from all the other classes ($y \neq 0$), and then to discern the different classes. To do so, we implemented an ensemble model built with a binary classifier and a multiclass classifier. Between the two, there is the max spatial smoothing as described in the previous chapter. A visual representation of the stages of the ensemble is presented below (5.1).

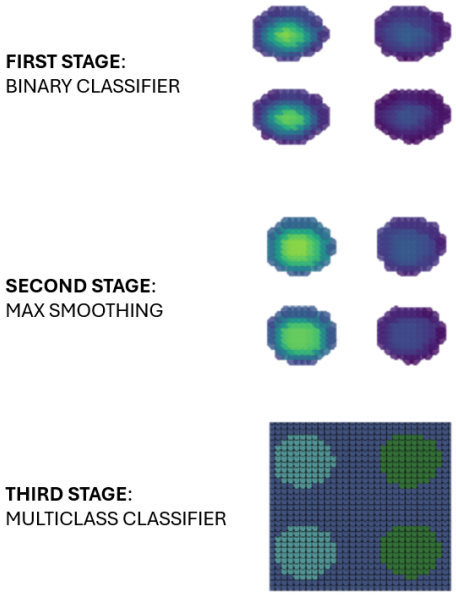


Figure 5.1: Ensemble Classifier Stages

5.2 Custom Cross Validation

Since we decided to perform the in-between spatial max-smoothing to avoid border effects, we cannot rely on a stratified k-fold cross validation. Thus, we decided to perform a 2-fold cross validation, using the whole train or validation set as folds. In this way, we maintain the spatial coherence necessary to apply the max filter in 3×3 grids.

5.3 RandomizedSearchCV

In order to select features, models (first and second stage), and their hyperparameters, we used a *RandomizedSearchCV* across all our parameters.
The models for binary and multiclass prediction are presented below, along with their hyperparameters:

Model	**Hyperparameters**
Support Vector Classifier	kernel: `rbf` C: [0.01, 0.1, 1, 10] γ: [`scale`, `auto`, 0.001, 0.01, 0.1]
Random Forest Classifier	n_estimators: 300; max_features: 0.3; criterion: `entropy`; bootstrap: `True` max_depth: [6, 8, 10] min_samples_split: [5, 10, 15] min_samples_leaf: [4, 8, 12]
Logistic Regression	max_iter: 300; solver: `saga` C: [0.001, 0.01, 0.1, 1.0] penalty: `[l1, l2]`
XGBoost Classifier	eval_metric: `mlogloss` n_estimators: [200, 300] learning_rate: [0.01, 0.05, 0.1] max_depth: [3, 5, 6] γ: [0, 0.1, 0.5, 1] subsample: [0.6, 0.8] colsample_bytree: [0.6, 0.8] min_child_weight: [3, 5, 7]

Table 5.1: Models and their hyperparameters

We ran 55000 iterations across the combinations of models, hyperparameters, and feature sets, training the ensemble classifier first on training and calculating the score (f1-score macro) on validation, and vice versa. The process results, meaning the best combinations of classifiers, their hyperparameters, and the most informative feature sets, are presented in the next section.

6 Training Results

6.1 Best Combinations

Table 6.1 presents the optimal hyperparameter configuration identified through random search optimization. The configuration consists of an XGBoost classifier for the binary classification stage and a Support Vector Classifier with RBF kernel for the multiclass classification stage.
Best achieved **f1-macro score** : 0.787.

Table 6.1: Best hyperparameter configuration found by random search

Component	**Parameter**	**Value**
Binary Classifier	Algorithm	XGBClassifier
	learning_rate	0.05
	n_estimators	200
	max_depth	3
	min_child_weight	3
	subsample	0.6
	colsample_bytree	0.6
	gamma	0.1
Multiclass Classifier	Algorithm	SVC
	kernel	RBF
	C	10
	gamma	scale
	probability	True
Feature Set	Selected Features	Stiffness, Upstroke, Downstroke, damping_coefficient, mobility, tkeo_mean

The optimal configuration employs a conservative XGBoost approach with a low learning rate (0.05) and moderate tree depth (3) to prevent overfitting, while using subsampling techniques (subsample=0.6, colsample_bytree=0.6) to improve generalization. The multiclass classifier utilizes an SVM with RBF kernel and automatic gamma scaling, configured to output class probabilities for downstream analysis.

6.2 Predict Probability Method

Since we are using a two stage classifier we had to readapt the classical `predict_proba` available for the base classifiers to compute class probabilities. The method follows the following steps:

1. Use the stage 1 classifier `predict_proba` method to compute:

$$P(y = 0) \quad \text{and} \quad P(y \neq 0)$$

   for each sample $\mathbf{x}$.

2. For samples where the stage 1 classifier predicts class 0, distribute $P(y \neq 0)$ uniformly over the remaining classes:

$$P(y = k) = \frac{1}{K-1} \cdot P(y \neq 0) \quad \text{for } k = 1, \ldots, K-1$$

   where K is the number of classes.

3. For samples predicted as non-zero, apply the stage 2 classifier `predict_proba` method after smoothing to get:

$$P(y = k \mid y \neq 0) \quad \text{for } k = 1, \ldots, K-1$$

4. Rescale the probabilities using conditional probability definition:

$$P(y = k \mid y \neq 0) = \frac{P(y = k \cap y \neq 0)}{P(y \neq 0)} \quad \text{for } k = 1, \ldots, K-1$$

   but since:

$$y = k \subseteq y \neq 0$$

   then:

$$P(y = k) = P(y \neq 0)P(y = k \mid y \neq 0) \quad \text{for } k = 1, \ldots, K-1$$

5. The final probability vector for each sample is:

$$P(y = 0), P(y = 1), \ldots, P(y = K-1)$$

6.3 Performance Metrics

The following subsections illustrate the results obtained after training the classifier on the whole training and validation set.

6.3.1 Confusion Matrices

Resulting confusion matrices can be seen in Fig. 6.1

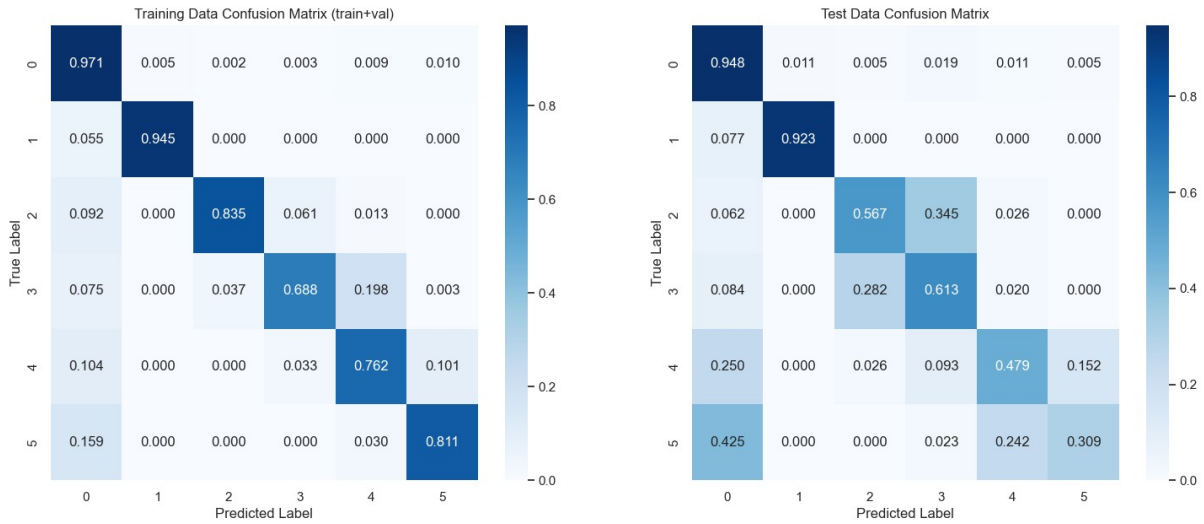


Figure 6.1: Confusion matrices of training (DamasconeA first measure and DamasconeA second measure) and test (DamasconeB)

Accuracy: 0.8383
F1 Score (macro): 0.6438

6.3.2 Single Value Metrics

Table 6.2 presents the detailed classification performance metrics for each class and overall macro-averaged scores on the test dataset. The model demonstrates strong discriminative ability for classes 0 and 1, achieving F1-scores above 0.89, while showing moderate performance for the remaining classes.

Table 6.2: Classification performance metrics by class and macro-averaged scores

Class	Precision	Recall	F1-Score	AUC
0	0.926	0.948	0.937	0.956
1	0.871	0.923	0.896	0.994
2	0.603	0.567	0.584	0.955
3	0.473	0.613	0.534	0.921
4	0.534	0.479	0.505	0.918
5	0.591	0.309	0.406	0.876
Macro Average	**0.666**	**0.640**	**0.644**	**0.937**

Our two-stage classifier achieves high Area Under the ROC Curve (AUC) values for multiple classes, while simultaneously exhibiting mediocre F1-scores. The inflated AUC values are likely due to the extreme class imbalance in our test dataset, as highlighted in recent studies (e.g., [3]). Therefore, we do not consider AUC a reliable metric for evaluating our model.

6.3.3 ROC Curves

The ROC curves (Fig. 6.2) suggest the model generally distinguishes between most classes better than random guessing. Classes 0, 1, and 2 show relatively strong performance, with curves that rise quickly at the start and maintain higher true positive rates across a range of false positive rates. However, the performance varies quite a bit between classes.

Class 5 (brown curve) stands out as the weakest, with a much flatter curve that's closer to the diagonal line, indicating it struggles more to separate positive from negative cases. Classes 3 and 4 fall somewhere in the middle — they do well initially but their curves don't maintain a consistent strong lift.

The macro-average curve (thick blue line) reflects this mix, staying closer to the better classes but not fully capturing the challenges seen in the weaker ones. All curves do eventually reach the top right corner, but the gap from the diagonal line shows the model does have some useful predictive power.

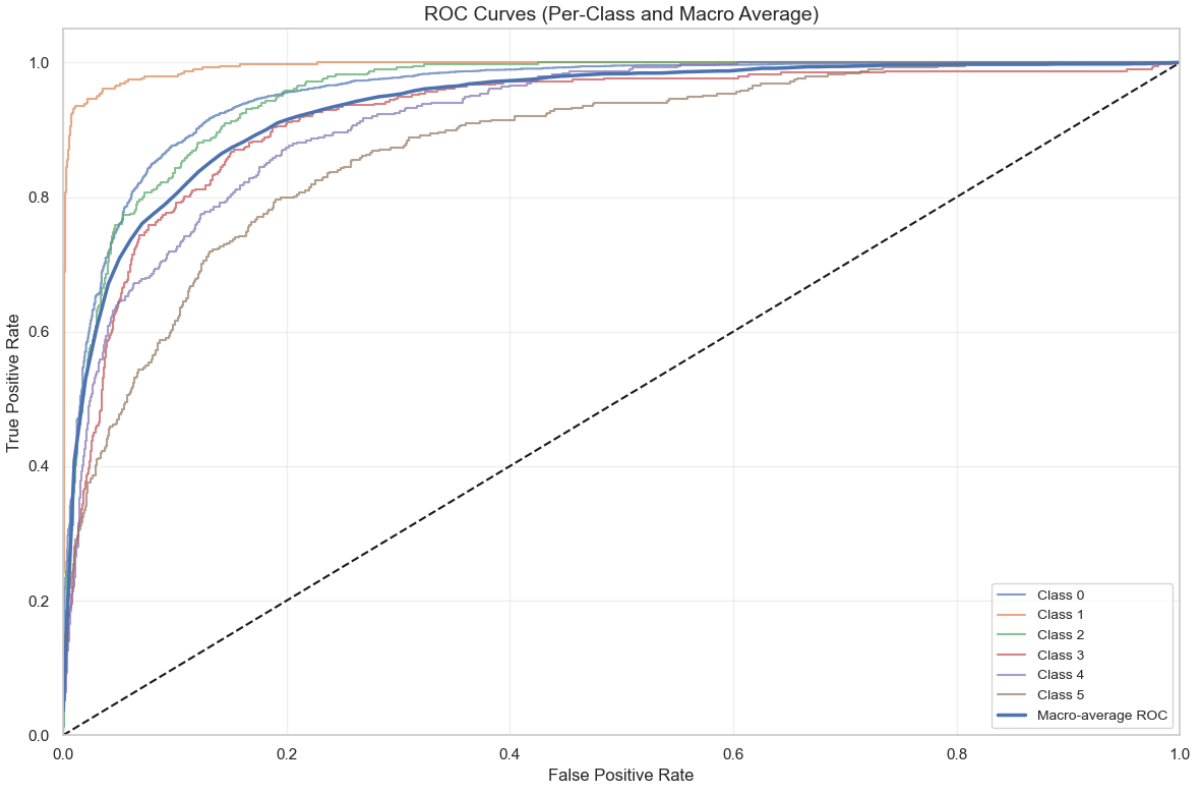


Figure 6.2: ROC curves per class and the average ROC curve for TEST set

6.4 Interpretation of the results

The results indicate that certain materials are more challenging for the model to distinguish, likely due to similarities in their physical characteristics. Dragon Skin 10, 20, and 30 (classes 0–2) are generally well classified, with Dragon Skin 10 (class 0) standing out as the most reliably identified. This is likely due to its greater presence in the dataset and more distinctive features.
In contrast, the model tends to confuse Dragon Skin 30 (class 2) with Sorta Clear (class 3), suggesting these materials may share similar traits that complicate classification. A similar pattern is seen between PDMS (class 4) and Econ 80 (class 5). Interestingly, Econ 80 is frequently misclassified as Dragon Skin 10, likely due to their shared characteristics, particularly stiffness.
The apparent similarities may be influenced by differences in **curvature** across samples. Since **geometric factors** could strongly affect the way material properties are measured. Future studies could explore methods to incorporate this additional degree of complexity (such as curvature effects) into the classifier or the data preprocessing pipeline
Despite these challenges, the spatial prediction map (Fig. 6.3) reveals that each material's insertion zone includes at least some correctly classified points. This suggests that the model is able to detect the presence of each material, even if not with complete accuracy. From an application standpoint, this is valuable: even partial detection can help identify potentially important or hazardous zones, offering meaningful support for downstream decision-making.

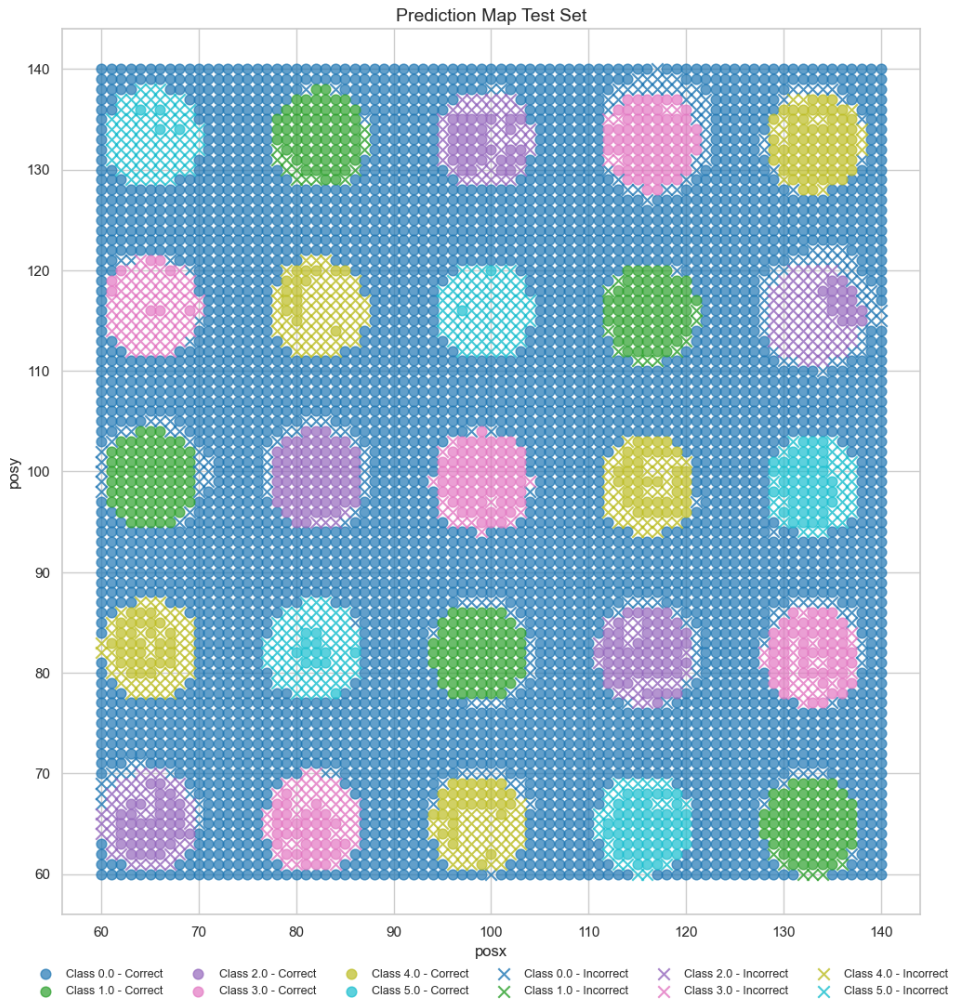


Figure 6.3: Spatial prediction map

7 SHAP for Model Explanation

7.1 SHAP Values Plot

To understand how each feature impacts the model's prediction, we computed the SHAP values on 10% of the test set (see Figure 7.1). We also computed per-class SHAP values; the results for class 3 are presented below (Figure 7.2).
The mathematical definition of SHAP values is the following:

$$\vec{f}(\mathbf{x_i}) = \vec{\phi}_0 + \sum_{j=1}^{M} \vec{\phi}_{ij} \tag{7.1}$$

Since SHAP operates on the output of `predict_proba` (i.e., the probability vector $\vec{f}(\mathbf{x}_i)$ for the i-th sample), the SHAP values are on the probability scale. In particular, for each dimension (class), the sum of SHAP values must be bounded within $[0, 1]$. Specifically, if for a data point i, feature j, and class k, the SHAP value $\phi_{ijk} = 0$, it indicates that feature j has no additional impact beyond the baseline expectation $[\vec{\phi}_0]_j$ (computed from a random 1% sample of the training dataset) for class k. Conversely, if $\phi_{ijk} \neq 0$, the SHAP value reflects a direct contribution (positive or negative) to the likelihood of class k for sample i.
Given that our features are normalized, we can directly compare the magnitudes of SHAP values across different features to assess their relative importance. We computed the SHAP values for a random 10% sample of our test set, and then calculated the mean. The features are listed in order of importance for the model's prediction. The results can be interpreted as follows:

- *damping_coefficient* is the most important feature across all classes except class 1, where stiffness is more informative.
- Higher values of *damping_coefficient* tend to decrease the probability of predicting class 3.
- Features that have SHAP values clustered near zero for a particular class are not informative or meaningful for that class (e.g. *mobility* is not informative for most of the points for class 3).
- It is uncommon to observe negative mean SHAP values, as this would imply that, for all data points, the feature decreases the probability of predicting a particular class, rendering it uninformative regardless of its value.

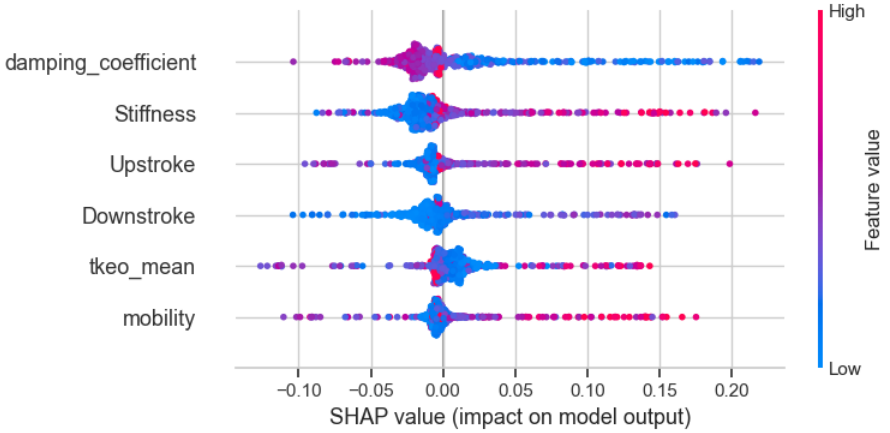


Figure 7.2: Class 3 SHAP plot

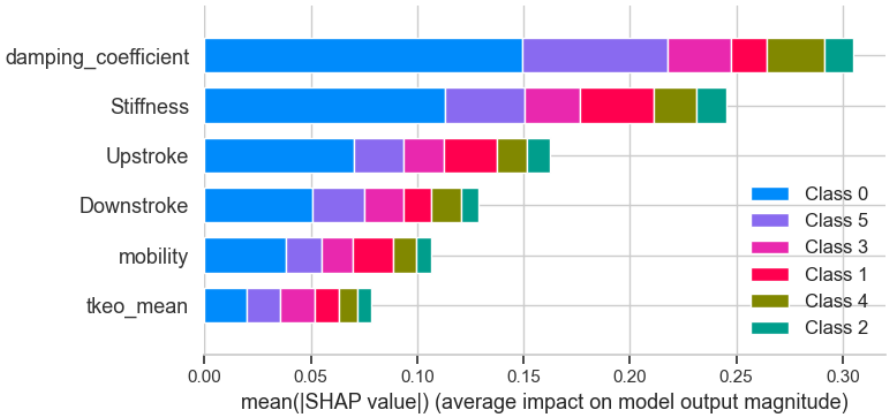


Figure 7.1: SHAP values on 10% of test set (stratified sample)

Bibliography

[1] Massari, L.; Bulletti, A.; Prasanna, S.; Mazzoni, M.; Frosini, F.; Vicari, E.; Pantano, M.; Staderini, F.; Ciuti, G.; Cianchi, F.; et al. A Mechatronic Platform for Computer Aided Detection of Nodules in Anatomopathological Analyses via Stiffness and Ultrasound Measurements. Sensors 2019, 19, 2512, doi: https://doi.org/10.3390/s19112512

[2] Haibo He, Yang Bai, E. A. Garcia and Shutao Li, "ADASYN: Adaptive synthetic sampling approach for imbalanced learning," 2008 IEEE International Joint Conference on Neural Networks (IEEE World Congress on Computational Intelligence), Hong Kong, 2008, pp. 1322-1328, doi:10.1109/IJCNN.2008.4633969.

[3] Movahedi F, Padman R, Antaki JF. Limitations of receiver operating characteristic curve on imbalanced data: Assist device mortality risk scores. J Thorac Cardiovasc Surg. 2023 Apr;165(4):1433-1442.e2. doi: 10.1016/j.jtcvs.2021.07.041. Epub 2021 Jul 30. PMID: 34446286; PMCID: PMC8800945.