ZNEUS Project 1

Amal Akhmadinurov, Lukáš Šebök

November 9, 2025

1 Tools

- Torch
- TorchMetrics
- Pandas
- Other utility libraries

2 Dataset

Steel Plates Faults
https://api.openml.org/d/1504

The target variable represents the fault type, with seven classes: *Pastry*, *Z_Scratch*, *K_Scratch*, *Stains*, *Dirtiness*, *Bumps*, and *Other_Faults*. This allows both **multiclass classification** (predicting the specific fault type) and **binary classification** (e.g., distinguishing one fault type from all others or fault vs. non-fault).

2.1 Train Test Val Splits

For binary classification split was defined: 0.9, 0.05, 0.05
For multiclass classification split was defined: 0.7, 0.15, 0.15
Different splits were define to make test and validation sets more representative considering disbalance of the dataset

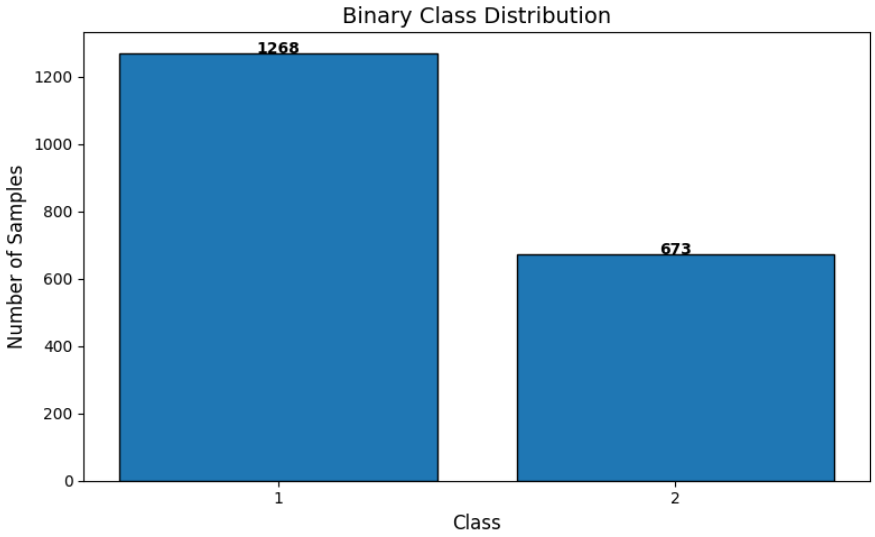

Figure 1: Class balance for binary classification

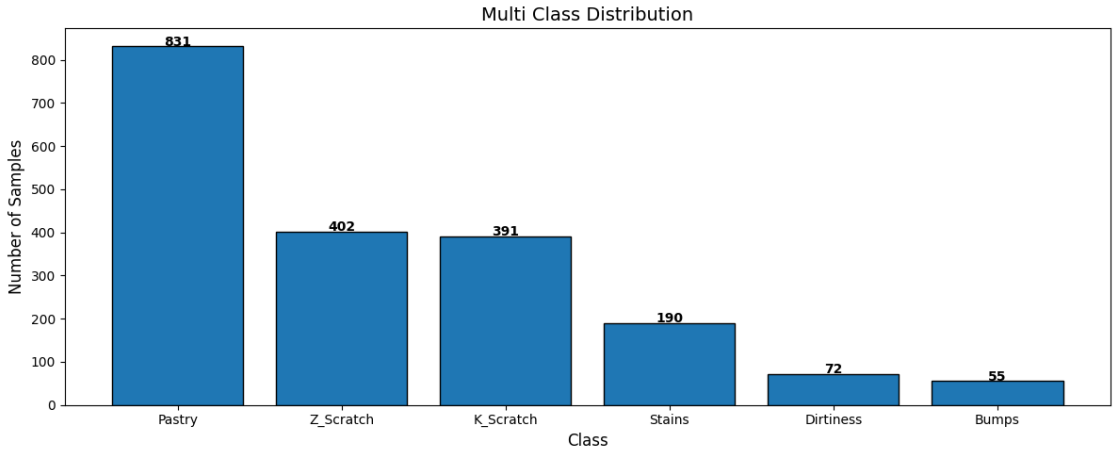

Figure 2: Class balance for binary classification

2.2 Normalization and data tranformation

2.2.1 Scaling

To normalize the data we first apply Z-Score standardization - we get the difference of each value from the mean of its column and divided by the standard deviation.

$$z = \frac{x - \bar{x}}{\sigma}$$

- **z** - the normalized value
- **x** - value of the row
- $\bar{x}$ - mean of the column
- σ - standard deviation

2.2.2 Outlier Detection and Clearing

Next we detect and clear some outliers in the dataset using Z-Score and a modified 5% 95% method. With the Z-Score method we're erasing rows containing score above or equal to 4. This is followed by a targetted clearing of outliers that have been discovered during Exploratory Data Analysis. Specifically we target columns Pixels_Areas, X_Perimeter, Y_Perimeter, Sum_of_Luminosity, Outside_X_Index, LogOfAreas and erasing rows above 98th percentile.

2.2.3 Feature Selection

At this point the data is split into Binary and Multiclass dataset. On the binary data we apply the Random Forest feature selection targetting the column Class to discover and rank the best non-linear predictors and discard the rest.

3 Binary Classification

Report available here: Report
Warning: Some models in the report have the same name, it means that architecture is the same and only some hyperparameters are changed

3.1 Model Description

This section describes the best model created.

3.2 Model Architecture

The neural network, implemented as the `MyModel` class, consists of a sequential stack of fully connected layers with the following structure:

- **Input Layer**: Accepts input features of size `input_size`.
- **Hidden Layer 1**: Linear layer mapping from `input_size` to 512 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Hidden Layer 2**: Linear layer mapping from 512 to 256 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Hidden Layer 3**: Linear layer mapping from 256 to 128 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Output Layer**: Linear layer mapping from 128 to 1 unit, producing a single logit for binary classification.

The model applies a sigmoid activation to the output logit during training and evaluation to produce probabilities in $[0, 1]$.

3.3 Hyperparameters

The model uses the following hyperparameters, defined as constants in the implementation:

Table 1: Hyperparameters of the Model

Parameter	**Value**
Batch Size	32
Learning Rate	0.001
Number of Classes	2
Maximum Epochs	50
Positive Class Weight	2
Dropout Probability	0.35
Reduce LR Patience	10
Early Stopping Patience	10

The loss function is `BCEWithLogitsLoss` with a positive class weight of 2 to address potential class imbalance. The Adam optimizer is used with an initial learning rate of 0.001. A `ReduceLROnPlateau` scheduler reduces the learning rate by a factor of 0.5 if the validation loss does not improve for 10 epochs. Early stopping is implemented with a patience of 10 epochs and a minimum delta of 0.01.

3.4 Training Procedure

The training procedure is implemented in the `fit` method, which processes the training data over a specified number of epochs (default 50). For each epoch, the model:

1. Iterates over batches of the training data loader.
2. Computes the forward pass to obtain logits, applies a sigmoid activation to produce probabilities, and calculates the binary cross-entropy loss.
3. Updates model parameters using backpropagation and the Adam optimizer.
4. Tracks training metrics: loss, accuracy, and precision for both positive and negative classes.
5. Evaluates the model on the validation set using the `evaluate` method.
6. Saves the model weights if the validation loss improves.
7. Adjusts the learning rate using the `ReduceLROnPlateau` scheduler based on validation loss.
8. Checks for early stopping based on validation loss stagnation.

3.5 Evaluation Metrics

The model evaluates performance using the following metrics, computed for both training and validation sets:

- **Loss**: Binary cross-entropy loss with logits, weighted for the positive class.
- **Accuracy**: Fraction of correct predictions, computed using `torchmetrics.Accuracy` for binary classification.
- **Positive Precision**: Precision for the positive class (label 1), computed using `torchmetrics.BinaryPrecision`.
- **Negative Precision**: Precision for the negative class (label 0), computed by transforming outputs and labels (`1 - output`, `1 - y`).

The `evaluate` method computes these metrics on the validation set without gradient computation, ensuring efficient evaluation.

4 Multiclass Classification

Report available here: Report
Warning: Some models in the report have the same name, it means that architecture is the same and only some hyperparameters are changed

4.1 Model Description

This section describes the best model created.

4.2 Model Architecture

The neural network, implemented as the `MyModel` class, consists of a sequential stack of fully connected layers with the following structure:

- **Input Layer**: Accepts input features of size `input_size`.
- **Hidden Layer 1**: Linear layer mapping from `input_size` to 512 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Hidden Layer 2**: Linear layer mapping from 512 to 256 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Hidden Layer 3**: Linear layer mapping from 256 to 128 units, followed by batch normalization, ReLU activation, and dropout with probability $p = 0.35$.
- **Output Layer**: Linear layer mapping from 128 to `num_classes` (6) units, producing logits for multiclass classification.

The model applies a argmax activation to the output logits during training and evaluation to produce probabilities for each class.

4.3 Hyperparameters

The model uses the following hyperparameters, defined as constants in the implementation:

The loss function is `CrossEntropyLoss` with class-specific weights [5.0, 5.0, 2.0, 3.0, 6.0, 6.0] to address class imbalance. The Adam optimizer is used with an initial learning rate of 0.001. A `ReduceLROnPlateau` scheduler reduces the learning rate by a factor of 0.5 if the validation loss does not improve for 10 epochs. Early stopping is implemented with a patience of 20 epochs and a minimum delta of 0.01.

Table 2: Hyperparameters of the Model

Parameter	Value
Batch Size	32
Learning Rate	0.001
Number of Classes	6
Train/Validation/Test Split	[0.7, 0.15, 0.15]
Maximum Epochs	100
Loss Weights	[4.0,4.0,2.0,3.0,5.0,5.0]
Dropout Probability	0.35
Reduce LR Patience	10
Early Stopping Patience	20

4.4 Training Procedure

The training procedure is implemented in the `fit` method, which processes the training data over a specified max number of epochs (default 100). For each epoch, the model:

1. Iterates over batches of the training data loader.
2. Computes the forward pass to obtain logits, applies a sigmoid activation to produce probabilities, and calculates the weighted cross-entropy loss.
3. Updates model parameters using backpropagation and the Adam optimizer.
4. Tracks training metrics: loss, macro-averaged accuracy, and macro-averaged precision.
5. Evaluates the model on the validation set using the `evaluate` method.
6. Saves the model weights if the validation loss improves.
7. Adjusts the learning rate using the `ReduceLROnPlateau` scheduler based on validation loss.
8. Checks for early stopping based on validation loss stagnation.

4.5 Evaluation Metrics

The model evaluates performance using the following metrics, computed for both training and validation sets:

- **Loss**: Weighted cross-entropy loss, accounting for class-specific weights.
- **Accuracy**: Macro-averaged accuracy across all classes, computed using `torchmetrics.Accuracy` with `task="multiclass"` and `average="macro"`.
- **Precision**: Macro-averaged precision across all classes, computed using `torchmetrics.Precision` with `task="multiclass"` and `average="macro"`.

The `evaluate` method computes these metrics on the validation set without gradient computation, ensuring efficient evaluation.

5 Binary Classification Results

5.1 Validation set results:

Class	Count
Class Positive	46
Class Negative	35

Table 3: Class distribution in the validation set.

- Test Loss: **0.8322**
- Test Accuracy: **0.7778**
- Test Negative Precision: **0.7917**
- Test Positive Precision: **.7576**

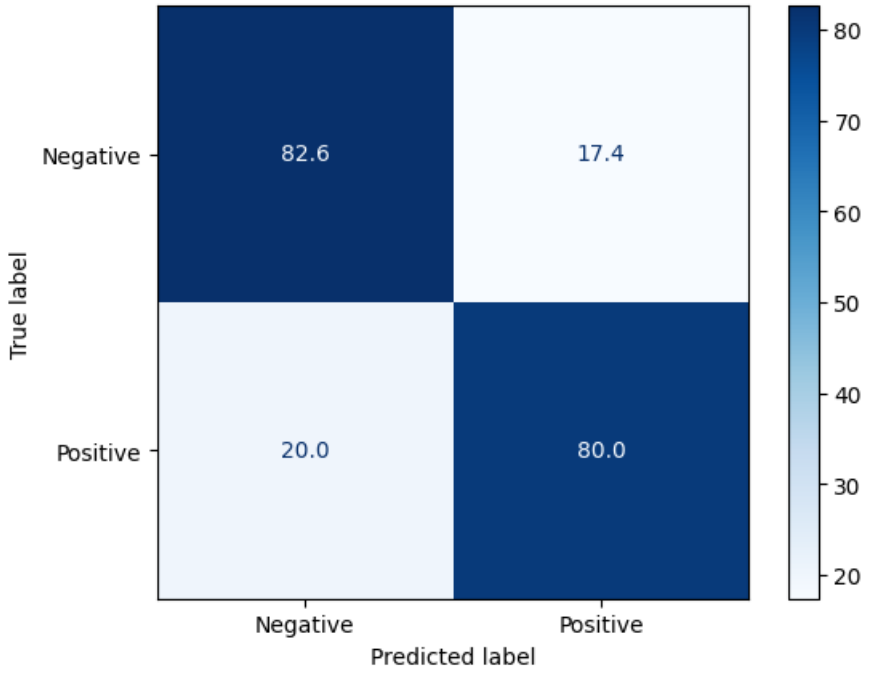

Figure 3: Confusion Matrix Validation set

5.2 Test set results:

Class	Count
Class Positive	52
Class Negative	28

Table 4: Class distribution in the test set.

- Test Loss: **0.8152**
- Test Accuracy: **0.8000**
- Test Negative Precision: **0.8103**
- Test Positive Precision: **0.7727**

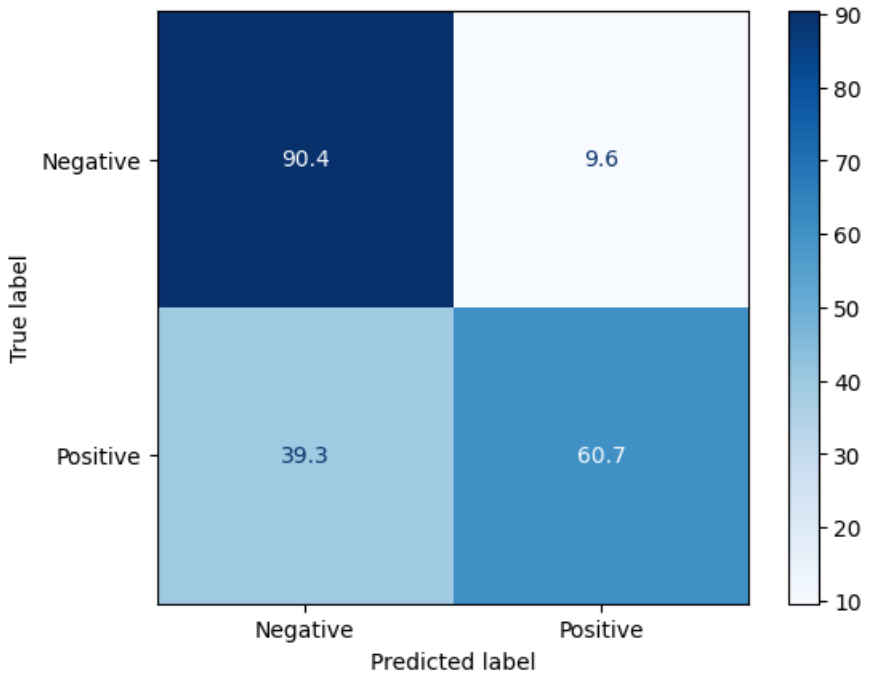

Figure 4: Confusion Matrix Test set

6 Multiclass Classification results

6.1 Validation set results

Class	Count
Pastry	110
Z_Scratch	28
K_Scratch	26
Stains	11
Dirtiness	8
Bumps	59

Table 5: Class distribution in the validation set.

- Test Loss: **1.2534**
- Test Accuracy: **0.7838**
- Test Precision: **0.8431**

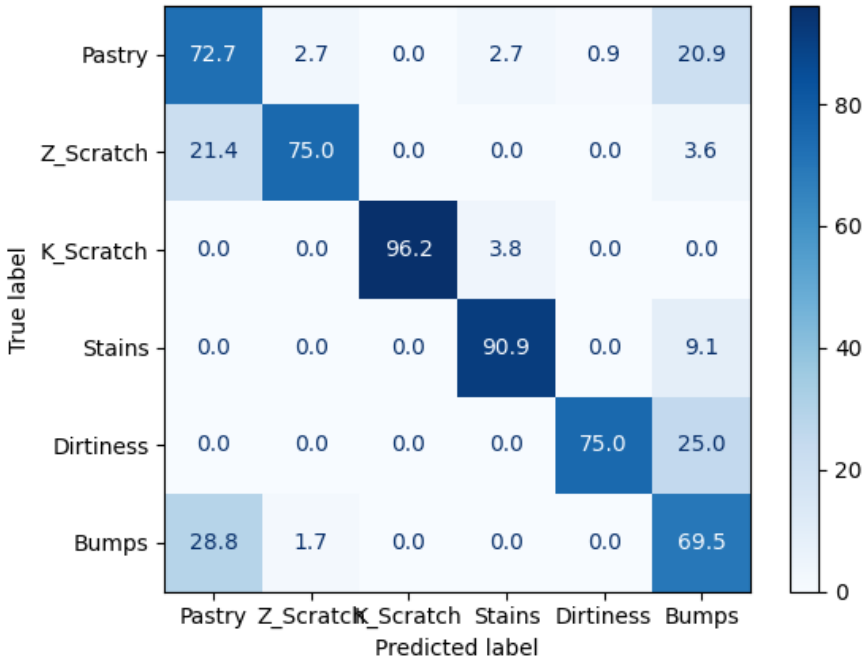

Figure 5: Confusion Matrix Validation set

6.2 Test set results

Class	Count
Pastry	110
Z_Scratch	28
K_Scratch	26
Stains	11
Dirtiness	8
Bumps	59

Table 6: Class distribution in the test set.

- Test Loss: **1.2503**
- Test Accuracy: **0.7988**
- Test Precision: **0.7985**

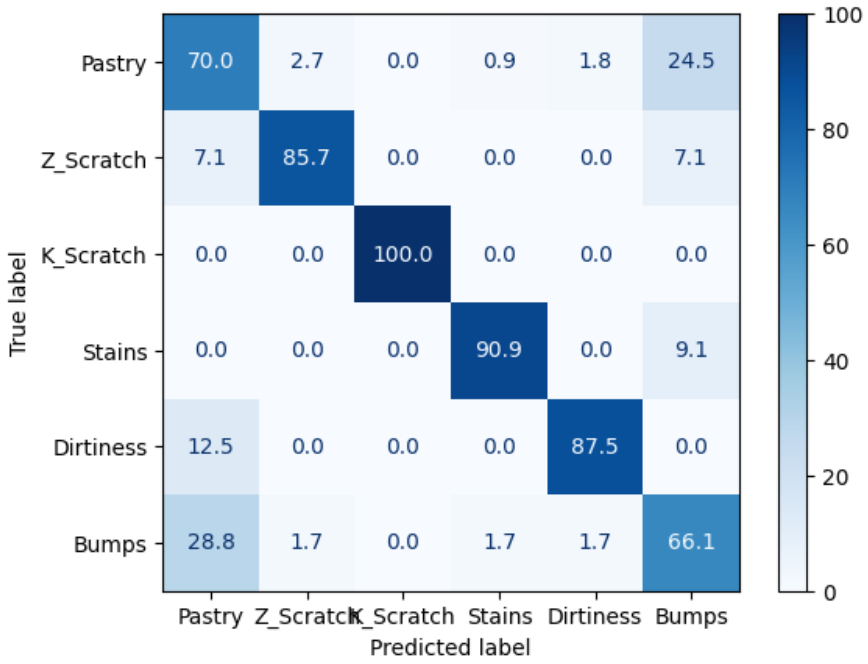

Figure 6: Confusion Matrix Test set