Beyond Surface Level: A Multi-Faceted Machine Learning Strategy for Steel Plate Fault Classification

Destiny Newman
University of Galway
Galway, Ireland
D.newman7@universityofgalway.ie

***Abstract*—This study's goal is to ascertain and further evaluate the viability of a some machine learning models for the task of classification of steel plate faults from the UCI machine learning repository. As well as this the ability of these selected models to classify the faults from the dataset with respect to each other is investigated as well. Weighted KNN, non-weighted KNN, and Multi-Layer Perceptron (MLP) classifiers were chosen and evaluated. Through extensive visualization and analysis of the UCI Steel Plates Fault dataset, it was found that feature correlations and non-normal distributions made Quadratic Discriminat Analysis (QDA), Naive Bayesian and Linear Discriminant Analysis (LDA) classifiers unsuitable. Feature engineering using LDA and Principal Component Analysis (PCA) was evaluated, with LDA achieving a superior silhouette score of 0.135 compared to PCA's 0.019. The LDA implementation reduced the feature dimensionality from 27 to 6 while maintaining discriminative power. Results showed that MLP achieved the highest base performance with 74.27% sensitivity and 96.49% specificity, though its performance decreased notably after LDA reduction. In contrast, both KNN variants demonstrated more resilience to dimensionality reduction, maintaining stable accuracy around 70%. A k-means clustering analysis with k=3 yielded a silhouette score of 0.3, suggesting potential for improved classification through data partitioning, though the dramatic performance improvements observed require further validation to rule out data leakage. The study reveals the robust nature of KNN classifiers for this domain, while highlighting important considerations for feature engineering and the impact of clustering on classification performance.**

I. Introduction

This study investigates the classification of steel plate faults using various machine learning techniques. The investigation focuses on visualising the Steel Plate Fault dataset from the UCI Machine Learning Repository, and based on that, deciding which machine learning algorithms are best for classification on this specific dataset. As well as this the next objective is to then compare the performance of the chosen models; those being; A Weighted and Non-Weighted K-Nearest Neighbour (KNN) classifier as well as a Multi-Layer Perceptron (MLP) Regressor model using appropriate performance metrics such as Sensitivity, Accuaracy and Specificity. This paper is organized as follows: Section II provides background information on the dataset and classification methods, Section III details the methodology, Section IV presents the results and analysis, and Section V concludes with the findings.

II. Background

A. Dataset Description

The Steel Plates Faults dataset contains nearly 2000 instances of 27 dimensional feature vectors which describe 7 possible faults that can occur in steel plates using numeric data. The dataset used one hot encoding to convey which class/fault type each feature vector belongs to. It is a way to convey categorical data into a binary format where a '1' indicates the class is present and '0' indicates otherwise. One Hot Encoding can help improve the performance of machine learning models. It allows models to capture complex relationships within the data that might be missed if categorical variables were treated as single entities.

B. Classification Methods

1) K-Nearest Neighbour (KNN) Classifiers: A K-Nearest Neighbour (KNN) classifier is a type of non-parametric statistical model used in supervised machine learning whereby data points are fed in and are used to train the model with instance pairs (X_n, Y_n), where X_n represents the feature vector of the n^{th} data-point. Y_n is the label corresponding to n^{th} feature vector. [3] The classifier then calculates the distances between each feature vector and a new "test point", and based on the value of k, looks at what class are the majority of the k nearest feature vectors to the new point and bases its classification of the new point on that. A weighted KNN is similar except that it calculates the weights associated with the k nearest points, the weight is calculated as $1/distance$. The weights for each class are then summed up and compared to each other, whichever class' weight is greatest, the new point then gets labeled that class.

2) Multi-Layer Perceptron (MLP) Regressor: The Multi-layer Perceptron (MLP) is a mathematical model inspired by the fundamental mechanisms of biological neural networks. Its core operation can be expressed by the equation:

$$Y_{j,k} = F_{j,k}\left(\sum_{i=0}^{N} W_{i,j,k}x_{i,j,k} - b_{j,k}\right) \tag{1}$$

This equation encapsulates the three key components that enable an MLP to process information:

- $W_{i,j,k}$: The connection strengths (weights) from input i to neuron j in layer k
- $b_{j,k}$: The activation threshold (bias) for neuron j in layer k
- $F_{j,k}$: The activation function that determines the neuron's response
- $Y_{j,k}$: The output signal from neuron j in layer k
- $x_{i,j,k}$: The input signal i received by neuron j in layer k

At its simplest level, this mathematical framework mimics the behavior of biological neurons. Just as biological neurons integrate multiple synaptic inputs and fire when a threshold is reached, each artificial neuron in an MLP computes a weighted sum of its inputs ($\sum_{n}^{k} W_n x_n$), subtracts a bias term (b), and passes the result through an activation function (F). Traditionally, this activation function was implemented as a Heaviside step function, producing binary output (0 or 1) based on whether the input exceeded the threshold. The power of MLPs lies in their ability to learn through back-propagation. This process involves calculating the error between predicted and desired outputs across the training dataset, then systematically adjusting the network's weights and biases to minimize this error. Through this iterative process, the network learns to recognize complex patterns in the input data. [4]

III. Methodology

A. Data Preprocessing

The dataset was loaded from the UCI repository and converted to a CSV format. The data was then split into an 80/20 training/testing set and standardized using Python's `StandardScaler` function to ensure uniform scaling.

B. Classifier Viability and Data Visualisation

1) Classifier Viability: To determine the classifier models most suitable for this study a combination of theoretical and empirical measures were used. Firstly, a script was written which would test a Weighted and Non Weighted KNN, Naive Bayesian Classifier, Linear Discriminant Analysis (LDA) and Quadratic Discriminant Analysis (QDA) Classifiers and as well, an MLP classifier, the F1-Score and Accuracy were evaluated for each model on the dataset. F1-Score was chosen here as it is useful in cases where there is class imbalance in the dataset, as will be seen in the Results section. F1-Score was also chosen as false positive and false negatives have a substantial impact in the case of fault detection in steel plates. Secondly, the top 3 classifiers were chosen for this study and the empirical results were verified with theoretical knowledge to make sure they were consistent and there wasn't an error in the code.

2) Data Visualisation: The class distribution was examined using a barplot to understand the balance of fault types in the dataset. A correlation heatmap was generated using seaborn's heatmap function with a 'coolwarm' color scheme, visualizing the pairwise correlation between all of the features in the dataset. Finally, a feature distribution plot was generated for 4 random features to highlight how the various distributions existing in the individual features of the dataset vary.

C. Feature Engineering

To help the models perform better, feature engineering was conducted on the dataset. The implementation utilised two dimensionality reduction techniques: Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). PCA was applied using sklearn's PCA class with a 95% variance threshold to determine the optimal number of components, resulting in significant dimensionality reduction while preserving key data variance. LDA was implemented using the `LinearDiscriminantAnalysis`function in python, incorporating both features and class labels to maximize class separability in the reduced dimensional space, with the number of components limited by (number of classes - 1). Both methods were evaluated using KNN classification, cross-validation, and silhouette scores. The dataset produced by the method with the best scores was then used to retrain the models and was compared to their non-feature engineered counterparts.

D. Clustering Analysis

A K-means algorithm was implemented to group similar fault patterns in the steel plate dataset. The optimal number of clusters was determined using both the elbow method and silhouette scores, analyzing cluster quality across k=2 to k=14. The implementation performed clustering with the optimal k value, and generated cluster assignments and saved the new dataset with the cluster assignments into a csv file for further analysis.

E. Classification Implementation

1) Non-Weighted KNN Classifier: The non-weighted KNN implementation uses sklearn's `KNeighborsClassifier` with uniform weights, determining optimal k through cross-validation across k=1 to k=20. The model processes standardized data with an 80/20 train/test split and evaluates performance using sensitivity, specificity, and accuracy metrics. Performance visualization includes an overall metrics table and sensitivity analysis plot.

2) Weighted KNN Classifier: The weighted KNN implementation is very similar to that of the Non-Weighted KNN and still uses sklearn's `KNeighborsClassifier` function but instead the string 'weights=distance' is passed instead of 'weights=uniform' as with Non-Weighted, optimal k was determined through cross-validation across k=1 to k=20. The model processes standardized data with an 80/20 train/test split and evaluates performance using sensitivity, specificity, and accuracy metrics. Performance visualization includes an overall metrics table and sensitivity analysis plot.

3) MLP Classifier: The MLP classifier implementation utilizes a neural network with two hidden layers (100, 50 neurons) using ReLU activation and Adam optimizer, incorporating early stopping with 10 iterations patience to ensure training stops early if the model doesn't converge in training. Performance evaluation was done by plotting its accuracy, specificity and sensitivity over all classed and was found for each class as well as micro and macro averaged values over all classes. A sensitivity analysis over each of the features was also conducted and plotted.

Note: The same respective model parameters were used for each model but was also run on the dataset generated from the k-means clustering algorithm to evaluate using a clustered dataset on the models' performance, each model was modified slightly to make it cluster sensitive, the same performance metrics were plotted except for the sensitivity analysis per feature. The same was also done on one of the two datasets generated from the feature engineering part.

IV. Results

A. Visualization Insights

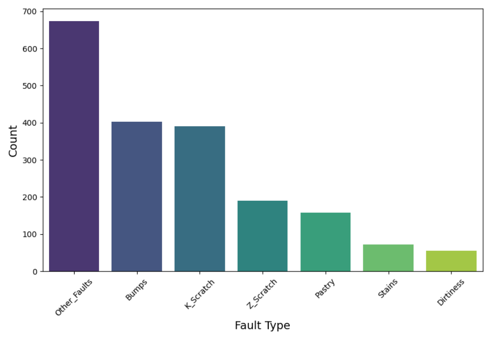

Fig. 1: Class Distributions Histogram

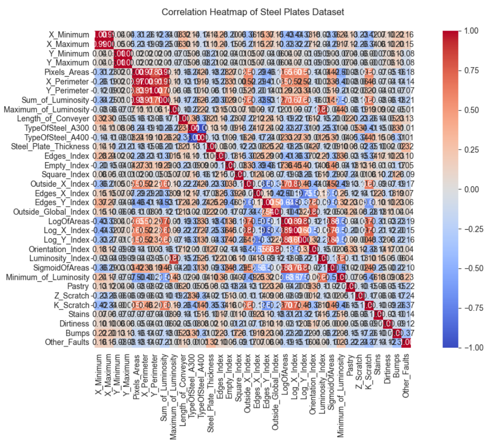

Fig. 2: Feature Correlation Heatmap

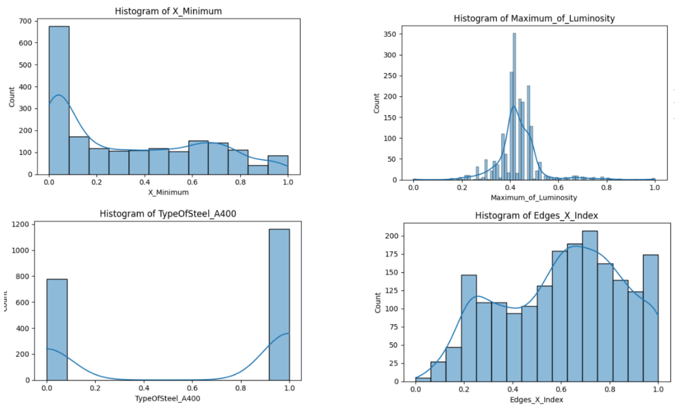

Fig. 3: Four randomly selected feature distributions from the dataset

The above graphics give good insight into why the 2 KNN and MLP classifiers were chosen for this study and the QDA, LDA and Naive bayesian classifiers weren't. The heatmap in Figure 2 shows the correlation between each feature and every othere feature in the dataset, as can be seen the *'YMinimum'* and *'Ymaximum'* features are positively perfectly correlated, while the 2 types of steel: 'A300' and 'A400' are perfectly negatively correlated, this breaks the assumption of Naive bayesian classifiers where it is assumed that all features are strongly independant of each other, which is obviously not true in this case. In Figure 3 it is seen that not all features follow a normal distribution which is a needed assumption for LDA and QDA classifiers to work, hence why they were not selected for the study.

As can be seen from Figure 1, there is a heavy class imbalance, with 'other faults' taking the vast majority of training examples in this dataset. KNN, both weighted and non-weighted were chosen as KNN classifiers are non-parametric which means that they do not need to assume any underlying dataset distribution to work effectively. A nice benefit we get from using Weighted KNN is that it is quite effective at negating the effect of outliers in the dataset due to the weight being a reciprocal of distance and for Non-Weighted KNN it's just that it's quite computationally efficient and will let us know if there is much of a discrepancy between the performance of it and a Weighted version. MLP's are quite a good classifier overall to use as they can be tuned to handle complex/non-linear datasets due to the feature of back propagation.

B. Classification Performance

Metric	MLP	Weighted KNN	Non-weighted KNN
Overall Sensitivity	0.7427	0.74	0.72
Overall Specificity	0.9649	0.95	0.94
Overall Accuracy	0.6427	0.7172	0.6999

TABLE I: Comparison of Overall Metrics Across Classifiers

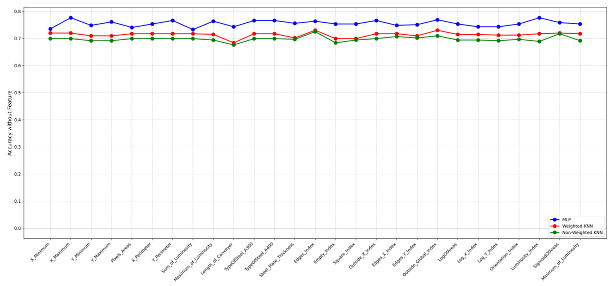

Fig. 4: Sensitivity Analysis across the 3 classifiers

From table 1 above it was found that the accuracy, sensitivity and specificity across all classes were quite good, but with much more room for improvement, also note in figure 4, the graph where the sensitivity is computed after removing an individual feature affects the models' performance, the MLP wins out here. The small fluctuations in the classifiers' sensitivity scores as features are removed shows that the models aren't overly relying on a single feature too much for classification, meaning that they are quite robust for this task and as well as this, that there is correlation between the features. Table 2 shows how each classifier performed for each individual class.

TABLE II: Performance Comparison of Different Classifiers on baseline normalised dataset (in %)

Metric	Class	MLP	W-KNN	NW-KNN
Sensitivity	Pastry	46.88	34.00	28.00
	Z_Scratch	81.82	91.00	92.00
	K_Scratch	94.37	92.00	91.00
	Stains	100.00	91.00	91.00
	Dirtiness	77.78	74.00	75.00
	Bumps	67.11	64.00	66.00
	Other_Faults	51.97	62.00	58.00
Specificity	Pastry	97.20	94.00	95.00
	Z_Scratch	97.75	95.00	96.00
	K_Scratch	98.74	97.00	97.00
	Stains	100.00	98.00	98.00
	Dirtiness	98.95	97.00	97.00
	Bumps	91.37	89.00	88.00
	Other_Faults	91.37	85.00	85.00
Accuracy	Pastry	93.06	71.00	69.00
	Z_Scratch	96.40	71.00	69.00
	K_Scratch	97.94	71.00	69.00
	Stains	96.49	71.00	69.00
	Dirtiness	94.46	71.00	69.00
	Bumps	86.63	71.00	69.00
	Other_Faults	76.35	71.00	69.00

C. *Clustering Analysis Impact*

A Silhouette Score of approximately 0.3 was computed when applying a k-means clustering algorithm to the normalised dataset, and was seperated into 3 clusters. From the table below there seems to be quite a substantial improvement over all classifiers and all metrics, but do note that these values are unnaturally high and may be due to data leakage so it cannot be said for certain just based on these metrics alone if the clustering algorithm improves performance or not without further analysis being conduct

TABLE III: Performance Comparison of Different Classifiers on Clustered Data (in %)

Metric	Weighted KNN	KNN	MLP
Overall Accuracy	99.49	99.49	92.54
Macro-averaged Sensitivity	97.72	97.72	92.99
Macro-averaged Specificity	99.91	99.91	99.21

D. *Feature Engineering Impact*

LDA and PCA were both tested to see which of them performed better when feature engineering the normalised dataset, it was found that LDA came out with a silhouette score of 0.135 and PCA a score of 0.019 and each algorithm reduced the feature vector size to 6 and 10 features respectively from the original 27. It was decided that the models would be evaluated on the dataset modified by the LDA algorithm. See tabulated results in table 4.

TABLE IV: Performance Comparison of Different Classifiers After LDA Reduction (in %)

Metrics	Class	MLP	Weighted KNN	KNN
Overall	Accuracy	71.72	73.01	71.72
	Macro-avg Sensitivity	74.18	71.79	71.34
	Macro-avg Specificity	94.83	94.94	94.69
Sensitivity	Pastry	55.17	59.38	48.28
	Z_Scratch	90.24	86.84	87.80
	K_Scratch	91.57	97.44	92.77
	Stains	92.31	92.86	92.31
	Dirtiness	62.50	36.36	50.00
	Bumps	69.44	66.67	66.67
	Other_Faults	58.04	62.96	61.54
Specificity	Pastry	95.56	96.08	96.67
	Z_Scratch	96.55	96.30	96.84
	K_Scratch	99.35	98.71	99.35
	Stains	99.73	100.00	100.00
	Dirtiness	98.95	98.68	98.69
	Bumps	85.49	90.58	86.75
	Other_Faults	88.21	84.25	84.55

V. Conclusion

It was found that using a Non-Weighted and Weighted KNN as well an MLP served the best for working with the steel plates faults dataset. When running the script to decide which models to use for classification, the LDA model came out with an accuracy of 55% and an F1-Score of 0.66 while the QDA and Naive Bayesian Classifier came out with accuracies of less than 40% each and F1-Scores similar to that of the LDA model, While both the KNN models and MLP scored an accuracy in the 70 percents with an F1-Score of about the same caliber. It was found that there exists a heavy class imbalance in the dataset with 'Other Faults' trumping the other fault types, with it almost doubling the number of appearances compared to that of its runner up 'Bumps'. The features are not normally distributed, which had an impact on the viable classifiers for this dataset. Both the KNNs and MLP models were found to be viable as well as quite stable and didn't rely too heavily on a single feature when performing a sensitivity analysis.

A k-means clustering algorithm was implemented and it was found that it made the overall accuracy, specificity and sensitivity blow up to perhaps artificial amounts, more analysis is needed to determine its effects for certain. LDA was chosen for feature engineering as it had a greater silhouette score compared to that of PCA for the dataset, yet it appears that the MLP classifier has a notable decrease in performance compared to that of the MLP trained on the baseline dataset with the accuracy dropping nearly 20%. In contrast both KNNs were more resilient as they displayed stable accuracy scores around 70%. Detection of 'Stains' saw a decrease in sensitivity by about 8% in the MLP after LDA reduction. Specificity metrics remained consistent across all models in both cases. Overall KNN models showed more overall robustness when LDA reduction was applied. Note that further analysis would be needed to be carried out to verify if these results are due to statistical anomaly or not.

References

[1] UCI Machine Learning Repository: Steel Plates Faults Data Set, https://archive.ics.uci.edu/dataset/198/steel+plates+faults

[2] M. Fakhr and A. M. Elsayad, Steel Plates Faults Diagnosis with Data Mining Models, https://thescipub.com/pdf/jcssp.2012.506.514.pdf, Journal of Computer Science, Vol. 8, No. 4, pp. 506-514, 2012.

[3] My first assignment submission for this module.

[4] My third assignment submission for this module.