Context-Aware AutoML for Accurate Wheat Disease Detection

Muhammad Uzair[1], Radwa ElShawi[1,*] and Stefania Tomasiello[1,2]

[1]Institute of Computer Science, University of Tartu, Estonia
[2]Department of Industrial Engineering, University of Salerno, Fisciano, Italy

Abstract
Timely detection and management of crop diseases are crucial for food security and agricultural productivity. Traditional methods, which rely on manual inspection, are often slow and prone to human error. With the rise of diseases like stripe rust in wheat, there is a growing need for efficient automated detection methods. This paper proposes a novel classification strategy that leverages Automated Machine Learning (AutoML) in combination with advanced feature engineering techniques. We develop a scalable framework that detects stripe rust by extracting comprehensive statistical features from images, distinguishing disease symptoms from healthy crops. To enhance feature quality, we employ Context-Aware Automated Feature Engineering, which iteratively generates meaningful features to capture subtle patterns in the data. Our method achieves 95.35% accuracy on the RustNet dataset, significantly outperforming the state-of-the-art ResNet-18 model, which achieved 85.2% accuracy. These findings highlight the potential of AutoML and automated feature engineering to revolutionize disease detection in agriculture, offering a cost-effective alternative to traditional deep learning methods that require extensive computational resources and expertise.

Keywords
AutoML, disease detection, feature engineering, large language models

1. Introduction

The Food and Agriculture Organization (FAO) forecasts a 0.9% increase in global cereal utilization for 2023/24 compared to the previous year. Wheat, as the most widely cultivated crop globally, is essential to agriculture, with rising consumption expected in regions like the European Union, China, India, the UK, and the US [1]. However, wheat faces significant threats from diseases and pests, causing substantial annual losses, roughly one-fifth of global yield [2]. Among these, wheat stripe rust, caused by Puccinia striiformis f.sp.tritici, is particularly devastating, leading to severe yield losses [3]. This disease has become increasingly prevalent worldwide, posing serious risks to food security and agricultural sustainability.

Traditional methods for monitoring wheat rust rely on manual visual inspection, which is time-consuming, labor-intensive, and costly, making it impractical for large-scale agriculture [4]. Recent advancements in imaging technologies, especially the use of Unmanned Aerial Vehicles (UAVs), offer a promising alternative for automated crop disease detection. UAVs can capture high-resolution images of large fields, enabling more efficient and accurate disease monitoring [5, 6]. This technology, combined with advanced image processing techniques, holds great potential for timely and precise identification of disease outbreaks.

Effective and timely monitoring of yellow rust is essential for both disease management and sustainable crop production. Accurate disease mapping facilitates the judicious application of fungicides and enhances breeding programs by identifying resistant wheat varieties [6]. Machine learning (ML) techniques play a crucial role in achieving high precision in disease detection, focusing on extracting relevant features from images and utilizing classifiers such as Neural Networks, Random Forest, Support Vector Machines, and K-Nearest Neighbors [7, 8]. However, the complexity and manual effort required to develop these ML models have led to a growing interest in automating the ML process. This has spurred the development of Automated Machine Learning (AutoML) techniques [9, 10], which simplify the creation of ML pipelines by automating stages such as data preprocessing, feature engineering, model selection, and optimization. By reducing the need for manual intervention, AutoML streamlines the development of effective ML models, making advanced disease detection more accessible and efficient.

This study introduces a novel approach that integrates AutoML with context-aware feature engineering for the detection of stripe rust in wheat. We extract comprehensive statistical features from UAV-captured images and refine them using Context-Aware Automated Feature Engineering (CAAFE), a feature engineering method designed for tabular datasets [11]. CAAFE leverages a large language model (LLM) to iteratively generate additional semantically meaningful features based on the dataset description, enhancing the discriminatory power of the features. These refined features are then processed using the Tree-Based Pipeline Optimization Tool (TPOT) [12], an AutoML framework that automates the selection, optimization, and construction of classification models. Our proposed framework was rigorously evaluated on the publicly available RustNet dataset [6], achieving a remarkable accuracy of 95.35%. This represents a substantial improvement over the state-of-the-art ResNet-18 model, which attained an accuracy of only 85.2% [6].

2. Related Works

The application of Unmanned Aerial Vehicles (UAVs) for plant disease detection has garnered substantial interest, leading to the development of advanced methodologies that integrate image processing with Machine Learning (ML) algorithms. Gu et al. [13] introduced a method for detecting and quantifying the severity of narrow brown leaf spot, a common disease affecting rice crops. The methodology began with the extraction of color features and vegetation indices from UAV-acquired images. Pearson's correlation analysis was then employed to identify the four most significant features, which were subsequently used as inputs for support vector regression, achieving a high degree of accuracy in disease severity estimation.

Published in the Proceedings of the Workshops of the EDBT/ICDT 2025 Joint Conference (March 25-28, 2025), Barcelona, Spain
*Corresponding author: Radwa ElShawi (radwa.elshawi@ut.ee).
muhammad.uzair@ut.ee (M. Uzair); radwa.elshawi@ut.ee (R. ElShawi); stefania.tomasiello@ut.ee (S. Tomasiello)
1234-5678-9012 (R. ElShawi); 0000-0001-8208-8285 (S. Tomasiello)

In the field of wheat disease detection, Liu et al. [14] focused on identifying powdery mildew using UAV imagery. They meticulously extracted textural features such as contrast, correlation, and variance, and applied Partial Least Squares Regression (PLSR) for comprehensive analysis, yielding a nuanced understanding and quantification of the disease's impact. Additionally, a study on monitoring wheat scab using UAV remote sensing [15] emphasized the value of texture features derived from multiple spectral bands. When combined with vegetation indices, these features provided extensive data for disease monitoring, with Support Vector Regression (SVR) demonstrating effectiveness in predictive analysis. Zhang et al. [16] utilized a combination of spectral and textural features to detect Fusarium Head Blight in wheat crops, employing Logistic Regression to highlight the critical role of feature-rich datasets in accurate disease classification and monitoring. Subsequent studies [17, 18] advanced this approach by integrating spectral, textural, and color features with various classification models, including Support Vector Machines (SVM) and Neural Networks (NNs). These studies highlighted the significance of feature extraction techniques and the adaptability of ML algorithms in managing the complex datasets derived from UAV imagery. Furthermore, research on wheat yellow rust detection illustrated the interaction between traditional ML algorithms and Deep Learning (DL) techniques [19, 6, 20]. While ML methods such as SVR, NNs, and Random Forests demonstrated significant effectiveness, DL models have shown promising potential for enhancing accuracy and efficiency in disease detection tasks.

Broadening the application beyond wheat, research on UAV-based disease detection in rubber trees [21] and citrus plants [22] demonstrated the broad applicability of UAV-based disease detection techniques across different agricultural sectors. These studies emphasized the vital role of advanced image processing techniques and ML algorithms in enhancing global food security by enabling effective disease detection in a wide range of crops.

3. Methodology

Figure 1 illustrates the architecture of our approach that consists of three main stages, including data preprocessing (Section 3.2), automated feature engineering using CAFE (Section 3.3), and model training and evaluation using AutoML approach (Section 3.4). In the following subsections, we explain the different building blocks of our approach.

3.1. Dataset

In this study, we used a publicly available dataset, RustNet [6]. RustNet comprises 508 images categorized into two classes: disease and no disease. Among these, there are 281 images depicting instances of disease and 227 images without any disease. RustNet is based on data collected from two experimental wheat fields were imaged in Pullman, WA, the US, in 2021. Field 1, located at Palouse Conservation Field Station, comprised two winter wheat trials: one for testing fungicides on 'PS 279' variety and another for assessing stripe rust resistance in 23 winter wheat cultivars. Both trials had randomized designs with four replications, planted on November 1, 2020. Urediniospores of P. striiformis were inoculated twice to induce disease. Field 2, at Spillman Agronomy Farm, housed spring wheat nurseries

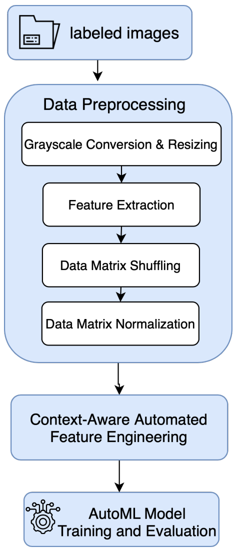
labeled images
Data Preprocessing
Grayscale Conversion & Resizing
Feature Extraction
Data Matrix Shuffling
Data Matrix Normalization
Context-Aware Automated Feature Engineering
AutoML Model Training and Evaluation

Figure 1: Flowchart of the proposed framework

with regular irrigation. Lemhi 66 cultivar in borders was highly susceptible to stripe rust, with three inoculated borders and one non-infected border. Images were collected only from the borders in Field 2.

3.2. Data preprocessing

Our preprocessing phase involves several key stages: initial image acquisition, conversion to grayscale, resizing, and feature extraction. During feature extraction, we compute essential statistical measures, including mean, standard deviation, variance, correlation, energy, entropy, contrast, skewness, kurtosis, and homogeneity.

3.3. Context-Aware Automated Feature Engineering

Feature engineering is a critical component of machine learning, as it involves transforming raw input data into features that can improve predictive performance [23, 24]. In our approach, we leverage CAAFE, an automated machine learning technique specifically designed for tabular datasets. CAAFE employs an LLM to iteratively generate semantically meaningful features based on a detailed description of the dataset. This process not only generates Python code for creating new features but also provides explanations for the relevance and utility of the generated features.

CAAFE operates iteratively on both the training and validation datasets, D_{train} and D_{valid}, along with a description of the dataset's context and features. In each iteration, CAAFE constructs a prompt that includes detailed information about the dataset and the specific feature engineering task, which is then passed to the LLM. Based on this prompt, the LLM generates code to alter or create new features. The generated code is executed on the current datasets (D_{train} and D_{valid}), producing transformed datasets ($D'train$ and $D'valid$). An ML classifier is subsequently trained on $D'train$ and evaluated on $D'valid$. If the classifier's performance on $D'valid$ surpasses its performance on the original $Dvalid$, the newly generated feature is retained, and the datasets are updated accordingly. If not, the feature is discarded, and the datasets remain unchanged.

The prompt provided to the LLM includes semantic and descriptive information about the dataset, such as a user-generated dataset description, feature names, data types, the percentage of missing values, and random sample rows from the dataset. Additionally, a template for the expected format of the generated code and explanations is included, which improves the clarity and quality of the LLM's output. To further enhance performance, chain-of-thought instructions guide the LLM through a series of intermediate reasoning steps, leading to more effective code generation. By utilizing CAAFE, we integrate domain knowledge into the feature engineering process, all while maintaining interpretability and optimizing predictive performance. This approach offers a powerful and efficient method for generating high-quality features in complex datasets, marking a promising advancement in machine learning research.

3.4. AutoML approach

TPOT is an AutoML framework designed for constructing and optimizing machine learning pipelines for both classification and regression tasks. It utilizes tree-based genetic programming [25] to evolve pipelines by treating them as individuals within an evolutionary algorithm. Each pipeline is structured as a tree, with its nodes categorized as either Primitives or Terminals. Primitives represent operators that require input, such as machine learning algorithms needing data and hyperparameter values. Terminals, on the other hand, are constants that provide input to the Primitives. Notably, a Primitive can also act as input for another Primitive, allowing for complex pipeline configurations. The evolutionary process in TPOT operates by applying genetic operations such as mutation and crossover to the pipelines. Mutation involves making small modifications, such as changing a hyperparameter or introducing a new preprocessing step. Crossover, on the other hand, selects two pipelines that share common Primitives and allows them to exchange subtrees or branches. Once these operations are performed, each pipeline is evaluated and assigned a fitness score, which reflects its performance. This fitness score is used in the selection process to determine which pipelines should be retained and evolved further in the next generation, ultimately leading to the creation of highly optimized machine learning pipelines. Generally, these pipeline trees could be arbitrarily large. Nevertheless, extensive machine learning pipelines usually have downsides. Longer pipelines with numerous hyperparameters can be challenging to fine-tune, more prone to overfitting, complicate the understanding of the final model, and demand extended evaluation time, thus slowing down the optimization process. Due to these considerations, a multiobjective optimization technique, NSGA-II [26], is employed. It assists in selecting candidates based on the Pareto front, representing the balanced trade-off between pipeline length and performance.

4. Experimental Evaluation

4.1. Experimental setup

Training and test. For a fair comparison, we adopted the same train-test split methodology as outlined in the referenced study, allocating 70% of the RustNet dataset for training and 30% for testing [6]. Detailed information regarding these splits is provided in Table 1.

Class	**Train**	**Test**	**Total**
disease	208	73	281
no_disease	172	55	227
Total	**380**	**128**	**508**

Table 1
Number of images in train and test split for the RustNet dataset

Baselines. Given the randomized nature of the experiments reported in [6], we conducted new experiments using the same computational setup as described in their study. Specifically, we employed ResNet-18, following the original architecture and hyperparameters outlined in [6], and initialized the model with pre-trained weights.

CAAFE setting. We leverage the advanced capabilities of OpenAI's language models, including GPT-3.5, as LLM within the CAAFE framework [27, 28]. The integration of these powerful language models enables CAAFE to generate semantically meaningful features iteratively, enhancing the effectiveness of feature engineering. To ensure robust performance and accuracy, we conduct ten feature engineering iterations using the CAAFE framework. Additionally, in the iterative evaluation of code blocks, we employ TabPFN (Tabular Predictive Functional Network), as proposed by Hollmann et al. [29], to assess the effectiveness of generated features and their impact on model performance.

TPOT setting. To ensure a fair comparison, an equal time budget was allocated for both TPOT and ResNet methodology. Experiments were constrained to a 20-minute time limit. This consistent time allocation ensures parity in computational resources between the methods, enabling a thorough and unbiased evaluation of their respective performances. The input to TPOT is a data matrix after performing the feature engineering step from CAAFE. The hyperparameters for TPOT were configured with a set number of generations, specifically 10, and a population size of 100. The resulting pipeline generated by TPOT, constrained by the specified time budget, is a multi-layer perceptron classifier with a learning rate of 0.01 and regularization parameter of 0.0001. The latter is a penalty term, constraining the size of the weights [30]. The aim of such a strategy is to reduce overfitting and enhance the generalization ability of the NN.

Hardware Resources. We conducted our experiments on a CPU environment. The CPU environment runs on Windows 11 Pro 64-bit (10.0, Build 22621) with 16 core Intel(R) Core(TM) i9-10885H Processor @ 2.40GHz,32 GB DIMM memory, and 1000 GB SSD data storage. All the approaches have been implemented in Python.

Performance metrics. Since the classification problem is being tackled in this study, the performance metrics used are Accuracy, Precision, Recall, and F1-score.

4.2. Results

4.2.1. Preprocessing

We followed the preprocessing steps described in Section 3.2. Regarding the conversion of the class associated with the image to a numerical equivalent, we adopted for RustNet dataset $disease = 1$ and $no\_disease = 0$.

After the statistical features are extracted from images, the resulting feature set is normalized using *min-max* normalization, where each feature has a value between 0 and 1. The general formula for *min-max* normalization is:

```
Dataset Description:
This dataset contains handcrafted statistical features from images. The images are
wheat images and there are two types of images, i.e with disease and without
disease. This dataset predicts whether an image is infected with disease or not.

Attribute Information:
- mean: mean of the image (numerical)
- std: standard deviation of the image (numerical)
- var: variance of the image (numerical)
- skewness: skewness of the image (numerical)
- entropy: entropy of the image (numerical)
- kurtosis: kurtosis of the image (numerical)
- contrast: contrast of the image (numerical)
- correlation: correlation of the image (numerical)
- energy: energy of the image (numerical)
- homogeneity: homogeneity of the image (numerical)
- class: class variable 1: disease
                        0: no disease

Some samples from the dataset:
<Sample 1, class=1>:
mean: 0.769010384, std: 0.761649904,var: 0.580110568, skewness: 0.466414969,
entropy: 0.963101,kurtosis: 0.369918743, contrast: 0.489227072,
correlation: 0.756602121, energy: 0.688121882,homogeneity: 0.954844229
.
.
.
<Sample 5, class=1>:
mean:0.636383104,std: 0.87729013,var: 0.769637974,skewness: 0.586431784,
entropy: 0.973940969,kurtosis: 0.37709383,contrast: 0.488186784,
correlation: 0.526767611,energy: 0.799576354,homogeneity: 0.947847399
<Sample 6, class=0>:
mean: 0.677572277,std: 0.854699258,var: 0.730510817,skewness: 0.549493916,
entropy: 0.980216291,kurtosis: 0.395917515,contrast: 0.474908547,
correlation: 0.69608474,energy: 0.762089244,homogeneity: 0.983787588
.
.
.
<Sample 10, class=0>:
mean: 0.568418388,std: 0.879868278,var: 0.774168186,skewness: 0.7706753,
entropy: 0.964696511,kurtosis: 0.47484991,contrast: 0.483253361,
correlation: 0.717102329,energy: 0.859660247,homogeneity: 0.926171716
```

```
# Feature name and description: "mean_variance_ratio", ratio of mean to variance.
# Usefulness: This feature could highlight the relationship between the mean and
    variance of the image, which could be useful in identifying patterns related
    to the presence of a disease.
# Input samples: 'mean': [0.4967, 0.7235, 0.5506],
                 'var': [0.879374249, 0.774168186, 0.691100425]


df['mean_variance_ratio'] = df['mean'] / df['var']
```

```
Iteration 1
Performance before adding features ROC 0.912, ACC 0.946.
Performance after adding features ROC 0.911, ACC 0.949.
Improvement ROC -0.001, ACC 0.003. The code was executed and changes to ´df´
were kept.
```

```
# Feature name and description: "contrast_energy_ratio", ratio of contrast to
    energy.
# Usefulness: This feature could highlight the relationship between the contrast
    and energy of the image, which could be useful in identifying patterns
    related to the presence of a disease.
# Input samples: 'contrast': [0.5, 0.51, 0.48], 'energy': [1.0, 0.78, 0.92]


df['contrast_energy_ratio'] = df['contrast'] / df['energy']
```

```
Iteration 2
Performance before adding features ROC 0.911, ACC 0.949.
Performance after adding features ROC 0.910, ACC 0.950.
Improvement ROC -0.001, ACC 0.001. The code was executed and changes to ´df´
were kept.
```

```
# Explanation why the column ‘correlation‘ is dropped
# The correlation of an image might not provide much useful information in terms
    of disease detection as it measures the degree to which the two variables in
    the image move in relation to each other, which might not be directly related
    to the presence of a disease.


df.drop(columns=['correlation'], inplace=True)
```

```
Iteration 5
Performance before adding features ROC 0.911, ACC 0.949.
Performance after adding features ROC 0.911, ACC 0.950.
Improvement ROC 0.000, ACC 0.001. The code was executed and changes to ´df´
were kept.
```

```
# Explanation why the column ‘contrast‘ is dropped
# The contrast of an image might not provide much useful information in terms
    of disease detection as it measures the degree of brightness and darkness
    of an image and it might not be very useful in disease detection.


df.drop(columns=['contrast'], inplace=True)
```

```
Iteration 7
Performance before adding features ROC 0.911, ACC 0.950.
Performance after adding features ROC 0.913, ACC 0.952.
Improvement ROC 0.002, ACC 0.002. The code was executed and changes to ´df´
were kept.
```

```
# Explanation why the column ‘energy‘ is dropped
# The contrast of an image might not provide much useful information in terms
    of disease detection as it measures local change in an image and it
    might not be very useful in disease detection.


df.drop(columns=['energy'], inplace=True)
```

```
Iteration 8
Performance before adding features ROC 0.913, ACC 0.952.
Performance after adding features ROC 0.913, ACC 0.953.
Improvement ROC 0.000, ACC 0.001. The code was executed and changes to ´df´
were kept.
```

Figure 2: Exemplary run of CAAFE on the RustNet image dataset. User-generated input is shown in blue, ML-classifier generated data in red, and LLM-generated code with syntax highlighting. The generated code contains a comment per generated/deleted feature following a template (Feature name, description of usefulness, features used in the generated code, and sample values). CAAFE improved the ACC on the validation dataset from 0.946 to 0.953 over 10 iterations, but only those improving ACC are shown.

$$x_i' = \frac{x_i - min(X)}{max(X) - min(X)}$$

where x_i' is the normalized value, $x_i \in X, i = 1, 2, \ldots, n$ is the original value.

4.3. Feature Engineering

A demonstration of CAAFE using the RustNet dataset is illustrated in Figure 2. User inputs are highlighted in blue, ML-classifier-generated data in red, and LLM-generated code is presented with syntax highlighting. The code includes comments for each generated feature, adhering to a predefined template in CAAFE's prompt. This template comprises the feature name, its utility description, the features utilized in the generated code, and sample values for these features. The retained generated features from CAAFE after 10 iterations include 'mean_variance_ratio', calculated as the mean divided by the variance, and 'contrast_energy_ratio', computed as the contrast divided by the energy. Incorporating the features generated by CAAFE into TPOT improved the accuracy from 93.02% achieved using TPOT alone on the validation dataset to 95.42%, as shown in Table 2.

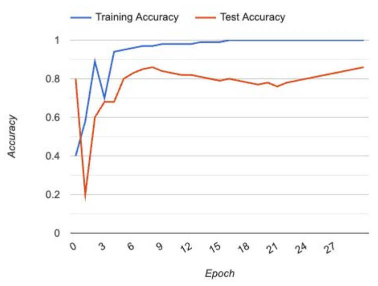

Figure 3: ResNet-18 Accuracy for RustNet dataset

4.4. AutoML

The results, evaluated using TPOT and ResNet-18, are detailed in Table 2. For TPOT, two variants are considered: the baseline TPOT and TPOT with Context-Aware Automated Feature Engineering (CAAFE), referred to as TPOT (FE). The comparative results demonstrate that both variants of our proposed framework—TPOT and TPOT (FE)—outperform the baseline ResNet-18 model. TPOT achieved an accuracy of 93.02%, which was further enhanced to 95.35% with the integration of CAAFE. In contrast, ResNet-18 achieved a lower accuracy of 85.2% on the same dataset, highlighting the superior performance of our proposed approach. The limited number of epochs achieved within the allocated time budget highlights the substantial computational effort required.

Dataset	Model	Accuracy	Precision	Recall	F1-Score
RustNet	TPOT	93.02	92.99	92.90	92.80
	TPOT (FE)	**95.35**	**95.79**	**94.85**	**95.22**
	ResNet-18	85.20	86.13	86.54	86.15

Table 2
Performance of TPOT and ResNet-18 on RustNet dataset

5. Conclusion

This study introduces a novel approach to detect stripe rust in wheat crops, using AutoML and rigorous feature engineering techniques. By extracting a comprehensive set of statistical features from original images and employing Context-Aware Automated Feature Engineering, we enhance the discriminative power of the extracted features. Our iterative feature generation process aims to capture subtle patterns and nuances, leading to superior effectiveness compared to state-of-the-art deep learning techniques. The considered wheat rust disease problem has already been tackled in the literature by employing several ML techniques, such as feed forward NNs, KNN, SVM, RF. All of them populated the search space of TPOT, which helped to determine the best one for the considered case. We compared our results against the ones by ResNet-18, a state-of-the-art technique used for the same kind of problem, according to the most recent literature. The experiments were performed on a publicly available dataset retrieved from the relevant literature. Our approach outperformed the above-mentioned state-of-the-art technique, revealing a higher computational effort of the latter in the allotted computing time.

Acknowledgments

This work has been partially funded by the Estonian Research Council, grant PRG1604, through the funding of SusAn, FACCE ERA-GAS, ICT-AGRI-FOOD and SusCrop ERA-NET, and through the project *Increasing the knowledge intensity of Ida-Viru entrepreneurship* co-funded by the European Union.

References

[1] FAO, Fao cereal supply and demand brief, 2023, www.fao.org/worldfoodsituation/csdb/en/, https://www.fao.org/worldfoodsituation/csdb/en/, 2023. Accessed: 2024-11-22.

[2] S. Savary, L. Willocquet, S. Pethybridge, P. Esker, N. McRoberts, A. Nelson, The global burden of pathogens and pests on major food crops, Nature Ecology & Evolution 3 (2019) 1. doi:10.1038/s41559-018-0793-y.

[3] X. Chen, Pathogens which threaten food security: Puccinia striiformis, the wheat stripe rust pathogen, Food Security 12 (2020). doi:10.1007/s12571-020-01016-z.

[4] J. su, C. Liu, X. Hu, X. Xu, L. Guo, W.-H. Chen, Spatio-temporal monitoring of wheat yellow rust using uav multispectral imagery, Computers and Electronics in Agriculture (2019). doi:10.1016/j.compag.2019.105035.

[5] D. Basurto-Lozada, A. Hillier, D. Medina, D. Pulido, S. Karaman, J. Salas, Dynamics of soil surface temperature with unmanned aerial systems, Pattern Recognition Letters 138 (2020). doi:10.1016/j.patrec.2020.07.003.

[6] Z. Tang, M. Wang, S. Michael, K.-H. Dammer, X. Li, R. Brueggeman, S. Sankaran, A. Carter, M. Pumphrey, Y. Hu, X. Chen, Z. Zhang, Affordable high throughput field detection of wheat stripe rust using deep learning with semi-automated image labeling, 2022. doi:10.20944/preprints202204.0177.v1.

[7] U. Shafi, R. Mumtaz, Z. Shafaq, S. Zaidi, Z. Mahmood, S. Zaidi, Wheat rust disease detection techniques: a technical perspective, Journal of Plant Diseases and Protection 129 (2022). doi:10.1007/s41348-022-00575-x.

[8] T. Hayıt, H. Erbay, F. Varçın, F. Hayıt, N. Akci, The classification of wheat yellow rust disease based on a combination of textural and deep features, Multimedia Tools and Applications 82 (2023) 1–19. doi:10.1007/s11042-023-15199-y.

[9] R. Elshawi, S. Sakr, Automated machine learning: Techniques and frameworks, in: R.-D. Kutsche, E. Zimányi (Eds.), Big Data Management and Analytics, Springer International Publishing, Cham, 2020, pp. 40–69.

[10] H. Eldeeb, M. Maher, O. Matsuk, A. Aldallal, R. El Shawi, S. Sakr, Automlbench: A comprehensive experimental evaluation of automated machine learning frameworks, 2022. doi:10.2139/ssrn.4516282.

[11] N. Hollmann, S. Müller, F. Hutter, Large language models for automated data science: Introducing CAAFE for context-aware automated feature engineering, in: Thirty-seventh Conference on Neural Information Processing Systems, 2023. URL: https://openreview.net/forum?id=9WSxQZ9mG7.

[12] R. Olson, J. Moore, Tpot: A tree-based pipeline optimization tool for automating machine learning, 2019. doi:10.1007/978-3-030-05318-5_8.

[13] C. Gu, T. Cheng, N. Cai, W. Li, G. Zhang, X.-G. Zhou, D. Zhang, Assessing narrow brown leaf spot severity and fungicide efficacy in rice using low altitude uav imaging, Ecological Informatics 77 (2023) 102208. doi:10.1016/j.ecoinf.2023.102208.

[14] Y. Liu, L. An, N. Wang, W. Tang, M. Liu, G. Liu, H. Sun, M. Li, Y. Ma, Leaf area index estimation under wheat powdery mildew stress by integrating uav-based spectral, textural and structural features, Computers and Electronics in Agriculture 213 (2023) 108169. URL: https://www.sciencedirect.com/science/article/pii/S0168169923005574. doi:https://doi.org/10.1016/j.compag.2023.108169.

[15] W. Zhu, Z. Feng, S. Dai, P. Zhang, X. Wei, Using uav multispectral remote sensing with appropriate spatial resolution and machine learning to monitor wheat scab, Agriculture 12 (2022) 1785. doi:10.3390/agriculture12111785.

[16] Y. Xiao, Y. Dong, W. Huang, L. Liu, H. Ma, Wheat fusar-

ium head blight detection using uav-based spectral and texture features in optimal window size, Remote Sensing 13 (2021). URL: https://www.mdpi.com/2072-4292/13/13/2437. doi:10.3390/rs13132437.

[17] H. Zhang, L. Huang, W. Huang, Y. Dong, S. Weng, J. Zhao, H. Ma, L. Liu, Detection of wheat fusarium head blight using uav-based spectral and image feature fusion, Frontiers in Plant Science 13 (2022) 1004427. doi:10.3389/fpls.2022.1004427.

[18] L. Liu, Y. Dong, W. Huang, X. Du, H. Ma, Monitoring wheat fusarium head blight using unmanned aerial vehicle hyperspectral imagery, Remote Sensing 12 (2020) 3811. doi:10.3390/rs12223811.

[19] A. Guo, W. Huang, Y. Dong, H. Ye, H. Ma, B. Liu, W. Wu, Y. Ren, C. Ruan, Y. Geng, Wheat yellow rust detection using uav-based hyperspectral technology, Remote Sensing 13 (2021) 123. doi:10.3390/rs13010123.

[20] C. Nguyen, V. Sagan, J. Skobalski, J. Severo, Early detection of wheat yellow rust disease and its impact on terminal yield with multi-spectral uav-imagery, Remote Sensing 15 (2023) 3301. doi:10.3390/rs15133301.

[21] T. Zeng, J. Fang, C. Yin, Y. Li, W. Fu, H. Zhang, J. Wang, X. Zhang, Recognition of rubber tree powdery mildew based on uav remote sensing with different spatial resolutions, Drones 7 (2023) 533. doi:10.3390/drones7080533.

[22] S. Ding, J. Jing, S. Dou, M. Zhai, W. Zhang, Citrus canopy spad prediction under bordeaux solution coverage based on texture- and spectral-information fusion, Agriculture 13 (2023). URL: https://www.mdpi.com/2077-0472/13/9/1701. doi:10.3390/agriculture13091701.

[23] S. Wold, K. Esbensen, P. Geladi, Principal component analysis, Chemometrics and Intelligent Laboratory Systems 2 (1987) 37–52. doi:10.1016/0169-7439(87)80084-9.

[24] H. Eldeeb, R. El Shawi, Empowering machine learning with scalable feature engineering and interpretable automl, IEEE Transactions on Artificial Intelligence PP (2024) 1–16. doi:10.1109/TAI.2024.3400752.

[25] W. Banzhaf, P. Nordin, R. Keller, F. Francone, Genetic programming: An introduction on the automatic evolution of computer programs and its applications, 1998.

[26] K. Deb, A. Pratap, S. Agarwal, T. Meyarivan, A fast and elitist multiobjective genetic algorithm: Nsga-ii, Evolutionary Computation, IEEE Transactions on 6 (2002) 182 – 197. doi:10.1109/4235.996017.

[27] OpenAI, Gpt-3 can't count syllables - or doesn't "get" haiku. https://community. openai.com/t/gpt-3-cant-count-syllables-or-doesnt-get-haiku/18733, 2021. accessed on: 2024-03-1, 2021.

[28] OpenAI, openai/openai-cookbook: Examples and guides for using the openai api. https://github.com/openai/openai-cookbook, 2023b. (accessed on 03/1/2023), 2023.

[29] N. Hollmann, S. Müller, K. Eggensperger, F. Hutter, Tabpfn: A transformer that solves small tabular classification problems in a second, 2022. URL: https://arxiv.org/abs/2207.01848. doi:10.48550/ARXIV.2207.01848.

[30] Mlpclassifier documentation, https://scikit-learn.org/stable/modules/generated/sklearn.neural_network.MLPClassifier.html, 2024. Accessed: 2024-11-22.