**National Research University Higher School of Economics**
**Faculty of Computer Science**
**Bachelor's Programme Data Science and Business Analytics**
**01.03.02 Applied Mathematics and Information Science**

# Internship report

## Processing and Analysis of Spectra Using Machine Learning and Neural Networks

**Fulfilled by**

**Barinov Kirill Alekseevich**
*(Surname, Given Name, Middle Name if any)*

______________________________
*(signature)*

**Supervised by**

*Candidate of Technical Sciences,*
*Head of the lab C-1 , IITP RAS*
*(job or academic title)*

*Kurochkin.I.I*

*(surname, initials)*

______________________
*(signature)*

**Moscow, 2024**

# Table of contents

# Abstract

The thesis, devoted to the application of spectral analysis and machine learning to diagnose plant conditions, describes significant achievements and applications of these technologies in modern agriculture, focusing on the presymptomatic detection of plant diseases, which is an important area for improving agricultural productivity and sustainability.

In the modern agro-industry, one of the most important tasks is the fight against plant diseases, which can significantly reduce the yield and quality of crops. Traditionally, the diagnosis of plant diseases has largely relied on visual examinations and laboratory tests, which not only take a lot of time, but also often do not allow the detection of diseases at an early stage. The advent of spectral analysis technologies combined with the power of machine learning represents a fundamental solution to these problems.

The integration of machine learning with spectral analysis for early detection of plant diseases is considered. This approach is very useful because it allows for rapid, non-invasive and accurate diagnosis, which allows preventive measures to be taken before diseases manifest themselves noticeably.

The goals of this study focus on the development and validation of a machine learning model that can accurately classify plants as healthy or diseased based on their spectral data. This includes preprocessing spectral data to ensure consistency and quality, applying dimensionality reduction techniques to improve computational efficiency, and implementing various machine learning models to predict the presence of diseases.

The empirical results of the study indicate a high accuracy of disease detection. The SVM model, used in combination with LDA reduction and StandardScaler pretreatment, has achieved test accuracy of almost 99.7%, highlighting the potential of this technology in the revolutionary fight against plant diseases.

# Introduction

In the modern world, agriculture faces many problems, one of which is the fight against plant diseases. These diseases can seriously threaten the yield and quality of agricultural products, a significant part of the harvest is lost due to diseases [1]. The development of effective and operational methods for diagnosing plant health is becoming particularly relevant. One of the promising approaches in this field is the use of spectral characteristics analysis and machine learning, which allows to obtain detailed information about the condition of plants.

Traditional methods of diagnosing plant diseases often require considerable time, specialized knowledge and may be ineffective in detecting the early stages of diseases. In contrast, spectral analysis provides an opportunity for fast and non-invasive research that can be automated using machine learning methods.

The use of machine learning for spectrum analysis is an approach that can radically improve the process of diagnosing plant health. Machine learning models can process large amounts of data and identify complex patterns. This opens up new opportunities for early detection and prevention of the spread of plant diseases, as well as for improving the overall efficiency of the agricultural sector.

The use of unmanned aerial vehicles to fly over fields and collect spectral data further enhances the potential of this technology. This approach not only increases the efficiency of data collection, but also minimizes interference with the natural environment of plants, providing more accurate and objective information. These innovations can significantly improve agricultural decision-making processes by providing farmers with up-to-date crop status data for timely correction of agricultural missions.

Thus, the key perspective of this work is the possibility of detecting the disease in the early stages, before the appearance of visual symptoms in affected plants using machine learning and spectral analysis methods. This ensures not only timeliness, but also increases the effectiveness of the measures applied, contributing to the sustainability and productivity of agricultural crops.

## Goals and Objectives

Main goal: To develop and validate a machine learning model capable of classifying plants into healthy and diseased plants with high accuracy based on their spectral data. This will help to detect diseases in the early stages, before the appearance of visual symptoms in affected plants, which will allow taking measures to protect the crop in the early stages of the disease development.

Objectives:

Preprocessing of spectral data:

- Standardization of data to ensure data quality and consistency before using it in machine learning models.
- The use of dimensionality reduction methods and their optimization to identify the most informative features, achieve maximum efficiency with minimal loss of information and reduce computational complexity.

Application of machine learning models:

- Development and training of various machine learning models for classifying plant conditions.

Evaluating the performance of models:

- Using the accuracy metric to evaluate the effectiveness of classification models.
- Analyzing the results to identify best practices and approaches, as well as to identify potential improvements in data processing and analysis techniques.

## Literature review

Thanks to recent advances in agricultural technology, spectral analysis combined with machine learning techniques has shown promising results in the early detection of plant diseases. Two fundamental studies in this field made significant contributions to the diagnosis of cassava and rice plant diseases, respectively, using spectral data. These articles not only correspond to our research goals, but also contain fundamental methodologies and results, and will serve as the basis for our own research.

The research “Ovomugisha et al. (2018) - Machine learning for the diagnosis of plant diseases using spectral data[3]” conducted by Ovomugisha and his colleagues was aimed at automating the detection of diseases in cassava plants using spectral data. The main goal was to identify diseases at the pre-symptomatic stage, which would allow for earlier interventions and potentially significantly reduce crop losses. This study used spectral data collected from both clearly affected and seemingly healthy parts of cassava leaves.

The methods used in this study included the use of prototype-based classifiers along with more traditional machine learning models such as support vector machines (SVM) and random forest. These models were trained to distinguish between healthy and diseased plants based on their spectral characteristics, which are the absorption and reflection properties of light changed as a result of pathogenic processes in plant tissues.

The results of this study were very promising. Prototype-based classifiers, in particular, have shown a high level of accuracy, reaching 95% accuracy in identifying diseased plants. This high level of accuracy has demonstrated the potential of spectral analysis for early detection of diseases, which allows for more timely and effective methods of disease control.

Konrad and his colleagues focused their research “Conrad et al. (2020) - Machine Learning-Based Presymptomatic Detection of Rice Sheath Blight Using Spectral Profiles[4]” on detecting rice scutes in the pre-symptomatic stages using near-infrared spectroscopy. The study was motivated by the need to improve disease control strategies by identifying infected plants before they show visible symptoms. Accurate early detection can lead to more targeted processing and potentially save a significant portion of the crop from destruction.

According to their approach, BIC spectra were obtained from rice leaves that were either artificially inoculated or infected with a fungus that causes late blight of the shells. The team used several machine learning algorithms, including SVM and Random Forest, to develop classification models that could distinguish between grafted and non-grafted plants based on the spectral data obtained. In addition, discriminant analysis using the sparse partial least squares (SPLs-DA) method was used to test the effectiveness of the models.

The specific results of this study highlighted the effectiveness of using machine learning to interpret spectral data for disease detection. The SVM model, in particular, was able to distinguish fictitiously grafted plants from inoculated ones with an accuracy of 86.1%. These results highlight the potential of machine learning models that can serve as reliable diagnostic tools for early detection of plant diseases.

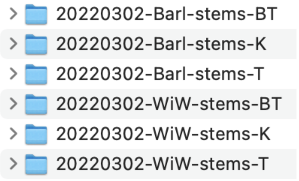
Owomugisha et al. (2018), and Conrad et al. (2020) demonstrate the potential of using spectral data and machine learning in disease control in agriculture. While Ovomugishi's research was a breakthrough in the detection of cassava diseases, achieving particularly high accuracy, Conrad's work applied similar concepts to rice with equally convincing results, emphasizing the universal applicability and scalability of these methods for various crops and diseases. Together, these studies expand our understanding of how early detection of plant diseases can be achieved through innovative technological integration, offering new opportunities to increase crop yields and reduce disease-related economic losses.

## Data

Spectral data is used in the research. Spectrum is the distribution of the intensity of electromagnetic radiation by frequency or wavelength[4]. In our case, the distribution of the intensity of electromagnetic radiation by wavelengths.

Spectral data, which is a key element for training our machine learning models, was collected using portable spectrometers in the laboratory. These devices are specifically designed to capture and analyze spectral data, which makes it possible to effectively collect detailed information about the condition of plants and their spectral characteristics.

To collect the data, a method was used to analyze the spectra of plant samples from six different groups: wheat without aphids, wheat with green aphids, wheat with black aphids, barley without aphids, barley with green aphids and barley with black aphids. Separate sets of spectra were formed for each of these groups, which were then saved in text files for subsequent processing. The files were organized and stored in folders, the names of which are formed according to the following principle: <date>-<Cultural attribute>-<A sign of aphid damage>. Figure 1 shows the folders that were used.

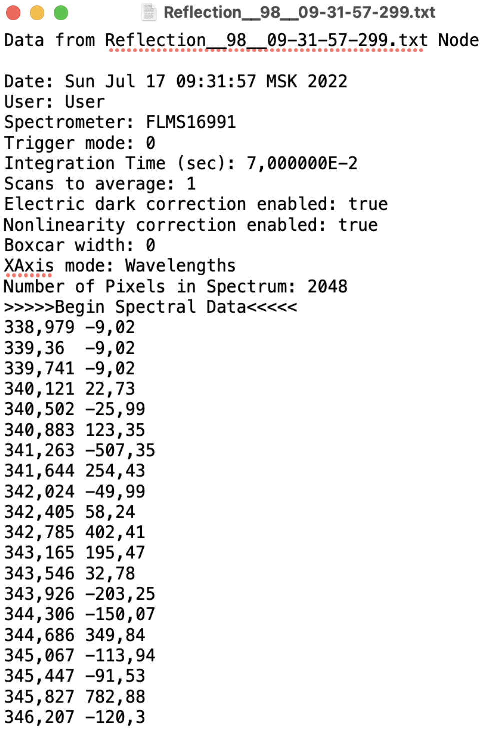


```
20220302-Barl-stems-BT
20220302-Barl-stems-K
20220302-Barl-stems-T
20220302-WiW-stems-BT
20220302-WiW-stems-K
20220302-WiW-stems-T
```

figure 1

The designations for the presence of aphids include: K - no aphid, T – green aphid, BT - black aphid. The attributes of the crop are divided into: WiW – wheat, Barl – barley.

The spectral data itself, which we receive from the spectrometer in the form of text files, are sequences of pairs of values: wavelength and corresponding intensity. An example of a text file is shown in Figure 2, and the interpretation of the spectrum values is shown in Figure 3.



```
Data from Reflection__98__09-31-57-299.txt Node

Date: Sun Jul 17 09:31:57 MSK 2022
User: User
Spectrometer: FLMS16991
Trigger mode: 0
Integration Time (sec): 7,000000E-2
Scans to average: 1
Electric dark correction enabled: true
Nonlinearity correction enabled: true
Boxcar width: 0
XAxis mode: Wavelengths
Number of Pixels in Spectrum: 2048
>>>>>Begin Spectral Data<<<<<
338,979 -9,02
339,36  -9,02
339,741 -9,02
340,121 22,73
340,502 -25,99
340,883 123,35
341,263 -507,35
341,644 254,43
342,024 -49,99
342,405 58,24
342,785 402,41
343,165 195,47
343,546 32,78
343,926 -203,25
344,306 -150,07
344,686 349,84
345,067 -113,94
345,447 -91,53
345,827 782,88
346,207 -120,3
```

figure 2

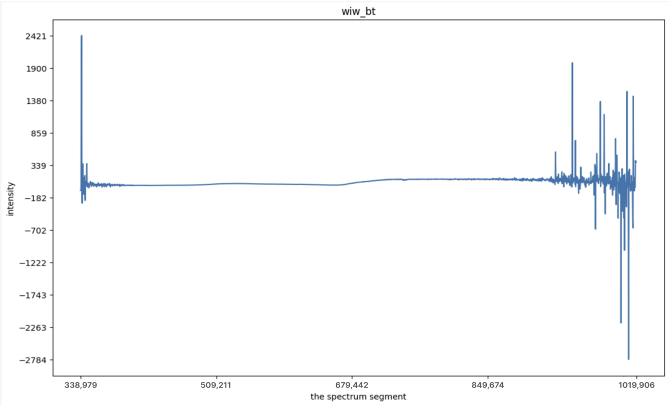


figure 3

Each line of the file represents a separate measurement, where the first number indicates the wavelength in nanometers in our case, this value is from 333.979 to 1019.906, and the second number indicates the intensity reflecting the amount of light reflected from or passed through the sample.

## Methods

### MinMax Scaling

It is a common preprocessing method used in preparing data for machine learning and data analysis applications. Scaling objects in a given range, usually from 0 to 1, ensures that the data will work well with algorithms that are sensitive to changes in scale[5].

How MinMaxScaler Works? MinMaxScaler operates by transforming each feature individually according to the formula: x_scaled = (x – x.min(axis=0)) / (x.max(axis=0) – x.min(axis=0)) * (max – min) + min

Where,

- min, max = feature_range
- x.min(axis=0) : Minimum feature value
- x.max(axis=0):Maximum feature value

Syntax: class sklearn.preprocessing.MinMaxScaler(feature_range=0, 1, *, copy=True, clip=False)

Parameters:

- feature_range: tuple (min, max), default=(0, 1)
  Desired range of scaled data. The default range for the feature returned by MinMaxScaler is 0 to 1. The range is provided in tuple form as (min,max).
- copy: bool, default=True
  If False, inplace scaling is done. If True , copy is created instead of in place scaling.
- clip: bool, default=False
  If True, scaled data is clipped to provided feature range.

MinMax scaler offers several benefits but also has limitations[6].

Advantages of MinMax Scaler:

- All objects are scaled to the same scale, which ensures that no object will dominate the machine learning process due to its scale.
- The distribution of objects remains the same, only the scale changes.

Limitation: MinMaxScaler does not handle outliers well, because minimum and maximum data values are used for scaling, outliers can significantly distort the scaled values, affecting these minimum and maximum values.

## Standart Scaling

It is a popular preprocessing method used to standardize data in preparation for working with machine learning algorithms. By converting the data so that it has a zero mean value and a standard deviation equal to one[7].

How StandardScaler Works? Standard Scaler standardizes a feature by deducting the mean and then dividing by the standard deviation: z = (x - u) / s

Where,

- z is scaled data.
- x is to be scaled data.
- u is the mean of the training samples
- s is the standard deviation of the training samples.

Syntax: class sklearn.preprocessing.StandardScaler(*, copy=True, with_mean=True, with_std=True)

Parameters:

- copy: bool, default=True
  If False, inplace scaling is done. If True , copy is created instead of inplace scaling.
- with_mean: bool, default=True
  If True, data is centered before scaling.
- with_std: bool, default=True
  If True, data is scaled to unit variance.

Standard scaler offers several benefits but also has limitations[8].

Advantages of a standard scaler:

- It transforms data at a scale suitable for algorithms that assume a normal distribution of data.
- By scaling objects, he ensures that they all contribute equally to the result, avoiding the dominance of one object due to its scale.

Limitation: StandardScaler is sensitive to emissions. If outliers are present, they can affect the mean and standard deviation, which may lead to less efficient scaling of the remaining data.

## PCA [9]

Principal Component Analysis (PCA) is a statistical method of reducing dimensionality while maintaining as much variability as possible. It converts a set of correlated variables into a smaller number of uncorrelated variables called principal components.

How does PCA work:

1) Standardization of data. Since PCA is scale dependent, it is necessary to scale the objects in the data before applying PCA. Standardization involves subtracting the mean and dividing by the standard deviation for each value of each variable.
2) The covariance matrix is calculated. The covariance matrix expresses the covariance of each pair variables in the dataset. Covariance indicates the level to which two variables change together. If the total dataset contains N objects, the covariance matrix will be equal to NxN.

$$\mathrm{cov}(x_i, x_j) = \frac{\sum_{k=1}^{n}(x_{i,k}-\overline{x_i})(x_{j,k}-\overline{x_j})}{n-1}$$

3) The eigenvalues and vectors of the covariance matrix are calculated. The eigenvectors indicate the directions of maximum variability in the data, and the eigenvalues show how important each of these directions is. The eigenvectors are then sorted by their eigenvalues in descending order. Thus, you can select the first few eigenvectors that reflect the largest variance in the data. The main components are selected based on the size of the eigenvalues.
4) The source data is projected onto the selected main components, which leads to a decrease in the dimension of the data. The obtained data already in a smaller number of measurements contain most of the information about the variability of the initial data.

Syntax: class sklearn.decomposition.PCA(n_components=None, *, copy=True, whiten=False, svd_solver='auto', tol=0.0, iterated_power='auto', n_oversamples=10, power_iteration_normalizer='auto', random_state=None)

Advantages of Principal Component Analysis:

- Dimensionality Reduction: By reducing the number of variables, PCA simplifies data analysis, improves performance, and simplifies data visualization.
- Feature selection: To select a feature, you can use principal component analysis, which is the process of selecting the most important variables in a dataset. This is useful in machine learning, where the number of variables can be very large and it is difficult to identify the most important variables.
- Noise reduction: Principal component analysis can be used to reduce the noise level in the data. By removing the low-variance principal components that are assumed to represent noise, principal component analysis can improve the signal-to-noise ratio and simplify the identification of the underlying structure in the data.
- Data Compression: By presenting data using fewer core components that reflect most of the differences in the data, PCA can reduce storage requirements and speed up processing.

Disadvantages of principal component analysis:

- Loss of information: Analysis of the main components can lead to loss of information. Although the analysis of the main components reduces the number of variables, it can also lead to loss of information. The degree of information loss depends on the number of selected main components. Therefore, it is important to carefully select the number of main components to be stored.
- Nonlinear relationships: The analysis of the main components assumes that the relationships between variables are linear. However, if there are non-linear relationships between the variables, the analysis of the main components may be ineffective.
- Overfitting: The analysis of the main components can sometimes lead to overfitting when the model fits the training data too well and does not work well on new data. This can happen if too many core components are used or if the model is trained on a small dataset.

## LDA [10]

Linear discriminant analysis (LDA) is a method of controlled dimensionality reduction. Unlike principal component analysis, which focuses on maximizing variance in data, LDA aims to maximize separation between different classes in a classification task.

Using LDA to reduce dimensionality includes:

1. Preprocessing the data. Make sure that there are no errors or missing values in the data. Although LDA is less sensitive to object scaling than methods like PCA, providing the same scale of objects can improve performance.
2. Calculate the average vectors and scattering matrices for each class in the dataset.
3. Find the eigenvectors and values of the generalized eigenvalue problem generated using scattering matrices.
4. Sort the eigenvalues in descending order and select the k upper eigenvectors corresponding to the k largest eigenvalues to form the transformation matrix.
5. Project Data: Project your data onto a new k-dimensional subspace formed by the transformation matrix.

Syntax: class sklearn.discriminant_analysis.LinearDiscriminantAnalysis (solver='svd', shrinkage=None, priors=None, n_components=None, store_covariance=False, tol=0.0001, covariance_estimator=None)

Linear Discriminant Analysis (LDA) offers several benefits but also has limitations that affect its performance in certain scenarios[11].

Advantages of using LDA:

- LDA is easy to implement and computationally efficient, which makes it suitable for environments with limited computing resources.
- It works well even when the number of objects far exceeds the number of training samples, effectively eliminating the dimension problem.
- LDA is resistant to multicollinearity by finding linear combinations of correlated features that maximally separate classes.

Disadvantages of using LDA:

- LDA assumes that the data within each class is normally distributed and all classes have the same covariance matrices. These assumptions may not match the actual data, which affects performance.
- LDA is looking for a linear decision boundary, which may not be enough for datasets where classes are non-linearly separated.
- Despite its relative efficiency in high-dimensional spaces, LDA can still suffer from retrofitting and may require high computational costs in very high-dimensional spaces.

## Kernel PCA

Kernel Principal Component Analysis is a method used in machine learning for nonlinear dimensionality reduction. It is an extension of the classical Principal Component Analysis Algorithm (PCA), which is a linear method that allows you to identify the most significant functions or components of a dataset. KPCA applies a non-linear mapping function to the data before applying PCA, which allows you to capture more complex and non-linear relationships between data points[12].

Steps for implementing Kernel PCA[13]:

1) Select an appropriate kernel function:

- Linear kernel: $k(x, y) = x^T y$
- Polynomial kernel: $k(x, y) = (x^T y + c)^d$
- Radial Basis Function (RBF) kernel: $k(x, y) = \exp(-\gamma \|x - y\|^2)$
- Sigmoid kernel: $k(x, y) = \tanh(\alpha x^T y + c)$

2) Compute the Gram Matrix: Calculate the Gram matrix K where each element is defined by a kernel function applied to all pairs of data points: $K_{ij} = k(x_i, x_j)$

3) Center the Gram matrix in the feature space. This is done by subtracting the average value of each row and column of the original Gram matrix:

$$K'_{ij} = K_{ij} - \frac{1}{n}\sum_{j=1}^{n} K_{ij} - \frac{1}{n}\sum_{i=1}^{n} K_{ij} + \frac{1}{n^2}\sum_{i,j=1}^{n} K_{ij}$$

where n is the number of data points.

4) Calculate the eigenvalues and vectors of the normalized Gram matrix K'. This step is important because it defines the main components in the feature space.

5) Arrange the eigenvalues in descending order and rank the corresponding eigenvectors. In this order, priority is given to the main components that reflect the largest variance in the dataset.

6) For a given input value x, the projection onto the main component associated with the eigenvector α is calculated as follows:

$$\text{projection}(x) = \sum_{i=1}^{n} \alpha_i k(x, x_i)$$

Here ai are the coefficients from the eigenvector corresponding to one of the largest eigenvalues, and k(x,xi) is the kernel function that calculates the relationship between the new point x and the learning points xi.

Syntax: class sklearn.decomposition.KernelPCA(n_components=None, *, kernel='linear', gamma=None, degree=3, coef0=1, kernel_params=None, alpha=1.0, fit_inverse_transform=False, eigen_solver='auto', tol=0, max_iter=None, iterated_power='auto', remove_zero_eig=False, random_state=None, copy_X=True, n_jobs=None)

Kernel PCA offers several benefits but also has limitations[12].

Advantages of using Kernel PCA:

- KPCA can capture complex non-linear patterns in data that traditional PCA may miss, due to its ability to display data in a multidimensional space.
- It is useful for datasets with a large number of functions, where it can reduce the dimension without losing important information.

Disadvantages of using Kernel PCA:

- Choosing the right kernel function and its parameters is crucial and can be a difficult task because there is no universal solution. The effectiveness of KPCA largely depends on this choice.
- KPCA can be computationally difficult, especially for large datasets, because it involves calculating a kernel matrix that scales quadratically depending on the number of data points.

### Isomap [14]

Isomap is a well-known nonlinear dimensionality reduction method. Designed as an alternative to traditional linear methods such as principal component Analysis (PCA), Isomap eliminates the limitations of linear methods by preserving the internal geometry of the data.

Steps for implementing Isomap:

1) The algorithm starts by calculating the Euclidean distances between data points.
2) Find the nearest neighbors according to these distances: For each data point, k nearest neighbors are determined by this distance.
3) Create a neighborhood graph: The edges of each point are aligned with their nearest neighbors, which creates a diagram representing the regional data structure.
4) Calculation of geodetic distances: Floyd's algorithm sorts all pairs of data points in the neighborhood graph and finds the most remote paths. Geodetic distances are represented by these shortest paths.
5) Performing dimensional reduction: The classic multi-scaling MDS is used for geodetic distance matrices, which lead to the introduction of low-dimensional data.

Syntax: class sklearn.manifold.Isomap(*, n_neighbors=5, radius=None, n_components=2, eigen_solver='auto', tol=0, max_iter=None, path_method='auto', neighbors_algorithm='auto', n_jobs=None, metric='minkowski', p=2, metric_params=None)

Advantages of Isomap:

- Where linear methods such as PCA do not work, Isomap can capture nonlinear relationships in data by approximating geodesic rather than Euclidean distances.
- Preserving the global structure: The program maintains the overall structure of diversity by providing an accurate representation of the original data distribution, which is crucial for holistic data analysis.
- Based on the built neighborhood graph, Isomap optimizes implementation on a global scale, providing the most accurate representation.

Disadvantages of Isomap:

- Calculating geodetic distances, especially using Floyd's algorithm, is computationally expensive and is not well suited for very large datasets.
- Complex topologies: Isomap may have problems with data containing complex topological structures (for example, holes), which can be misleading when calculating distances and lead to inaccurate implementation.

### Logistic Regression [15]

Logistic regression is a controlled machine learning algorithm used for classification tasks, the purpose of which is to predict the probability of an instance belonging to a given class or not. Logistic regression is a statistical algorithm that analyzes the relationship between two data factors.

Syntax: class sklearn.linear_model.LogisticRegression(penalty='l2', *, dual=False, tol=0.0001, C=1.0, fit_intercept=True, intercept_scaling=1, class_weight=None, random_state=None, solver='lbfgs', max_iter=100, multi_class='auto', verbose=0, warm_start=False, n_jobs=None, l1_ratio=None)

How does logistic regression work? The logistic regression model converts a continuous value output by a linear regression function into a categorical value output using a sigmoid function that converts any real-valued set of input independent variables to a value from 0 to 1. This function is called the logistic function.

Let the independent input characteristics be:

$$X = \begin{bmatrix} x_{11} & \cdots & x_{1m} \\ x_{21} & \cdots & x_{2m} \\ \vdots & \ddots & \vdots \\ x_{n1} & \cdots & x_{nm} \end{bmatrix}$$

and the dependent variable is Y having only binary value i.e. 0 or 1.

$$Y = \begin{cases} 0 & \text{if } Class\ 1 \\ 1 & \text{if } Class\ 2 \end{cases}$$

then, apply the multi-linear function to the input variables X.

$$z = \left(\sum_{i=1}^{n} w_i x_i\right) + b$$

Here xi is the ith observation of X, $w_i = [w_1, w_2, w_3, \cdots, w_m]$ is the weights or Coefficient, and b is the bias term also known as intercept. simply this can be represented as the dot product of weight and bias.

$$z = w \cdot X + b$$

Types of logistic regression:

- Binomial: In binomial logistic regression, there can be only two possible types of dependent variables, such as 0 or 1, success or failure, etc.
- Multinomial: In a multinomial logistic regression, there may be 3 or more possible disordered types of dependent variable, such as “cat", “dogs” or “sheep”.
- Ordinal: In ordinal logistic regression, there may be 3 or more possible ordered types of dependent variables, such as “low", “Medium”, or “High”.

Multinomial logistic regression:

The target variable can have 3 or more possible types that are not ordered (i.e., the types do not have a quantitative value), for example, “disease A”, “disease B” and “disease C".

In this case, the softmax function is used instead of the sigmoid function. The Softmax function for K classes will be: $\text{softmax}(z_i) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$

Here K represents the number of elements in the vector z, and i, j iterates over all the elements of the vector.

Then the probability for class c will be: $P(Y = c | \overrightarrow{X} = x) = \frac{e^{w_c \cdot x + b_c}}{\sum_{k=1}^{K} e^{w_k \cdot x + b_k}}$

## KNN [16]

The K-Nearest Neighbor Algorithm (KNN) is a supervised machine learning method that is used to solve classification and regression problems. KNN algorithm operates on the principle of similarity, where it predicts the label or value of a new data point by considering the labels or values of its K nearest neighbors in the training dataset.

Syntax: class sklearn.neighbors.KNeighborsClassifier(n_neighbors=5, *, weights='uniform', algorithm='auto', leaf_size=30, p=2, metric='minkowski', metric_params=None, n_jobs=None)

Workings of KNN algorithm:

1) Choosing the optimal K value: K is the number of nearest neighbors that must be taken into account when predicting.

2) Distance calculation: The Euclidean distance is used to measure the similarity between the target and training data points. The distance is calculated between each of the data points in the dataset and the target point.

3) Search for nearest neighbors: The nearest neighbors are the k data points with the smallest distances to the target point.

4) Voting for classification or getting an average value for regression: In the classification task, class labels are determined by majority vote. The class with the most matches among neighbors becomes the predicted class for the target data point. In a regression problem, the class label is calculated by averaging the target values of the K nearest neighbors. The calculated average value becomes the predicted result for the target data point.

Let X be a training dataset with n data points, where each data point is represented by a d-dimensional feature vector Xi, and Y by the corresponding labels or values for each data point in X. Given a new data point x, the algorithm calculates the distance between x and each data point Xi in X. using a distance metric such as Euclidean distance:

$$\text{distance}(x, X_i) = \sqrt{\sum_{j=1}^{d} (x_j - X_{i,j})^2}]$$

The algorithm selects K data points from X that have the shortest distances to x. For classification problems, the algorithm assigns the label y, which is most common among the K nearest neighbors of x. For regression problems, the algorithm calculates the average or weighted average of the y values from the K nearest neighbors and assigns it as the predicted value for x.

## SVM [17]

The support vector machine (SVM) method is one of the most popular learning methods used to solve classification and regression problems. The main idea of the method is to construct a hyperplane separating the sample objects in an optimal way. The algorithm works under the assumption that the greater the distance (gap) between the separating hyperplane and the objects of the shared classes, the smaller the average error of the classifier will be.

Syntax: class sklearn.svm.SVC(*, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=-1, decision_function_shape='ovr', break_ties=False, random_state=None)

How SVM works:

1) Determining the optimal hyperplane: SVM searches for the hyperplane with the maximum gap between data classes. This hyperplane functions as a crucial boundary that separates classes.
2) Maximizing the gap: The hyperplane is chosen in such a way as to maximize the distance (gap) between the points of different classes closest to this hyperplane, which are called support vectors.
3) Linear and nonlinear classification: In the linear case, SVM seeks to find a hyperplane in the original feature space. In the nonlinear case, core functions are used to map data to a higher dimensional space where data can be divided linearly.
4) Sound functions: Sound functions allow SVM to work efficiently in high-dimensional spaces without the need to explicitly calculate coordinates in these spaces. The most commonly used kernels are polynomial, radial basis (RBF), and sigmoid.
5) Mathematical formalization: The optimization problem for SVM is to minimize the norm of the weight vector, taking into account the conditions that ensure that each sample is classified correctly with a given gap.

### Gaussian Native Bayes [18]

The Gaussian naive Bayesian algorithm is a simple but powerful algorithm used for classification tasks. It belongs to the family of naive Bayes algorithms, which is based on Bayes' theorem. Given the class label, it is assumed that the features follow a Gaussian distribution and are conditionally independent.

Syntax: class sklearn.naive_bayes.GaussianNB(*, priors=None, var_smoothing=1e-09)

Parameters:

- priors: array-like of shape (n_classes,), default=None
  Prior probabilities of the classes. If specified, the priors are not adjusted according to the data.
- var_smoothing: float, default=1e-9
  Portion of the largest variance of all features that is added to variances for calculation stability.

How Gaussian Native Bayes works:

1) Calculate the previous probabilities for each class. This is the frequency of each class divided by the total number of samples
2) Calculate the mean value and variance of each feature for each class
3) Apply the Gaussian probability density function to calculate the probability of each feature for each class. The Gaussian probability density function for a given class k and feature j is equal to:

$$P(x_{i,j}|C_k) = \frac{1}{\sqrt{2\pi\sigma_{k,j}^2}} e^{-\frac{(x_{i,j}-\mu_{k,j})^2}{2\sigma_{k,j}^2}}$$

   where xi,j is the feature value to predict.
4) Classify new instances. Compute the posterior probability for each class using Bayes' Theorem:

$$P(C_k|x_i) \propto P(C_k) \prod_{j=1}^{n} P(x_{i,j}|C_k)$$

Choose the Ck class that maximizes this a posterior probability.

## Train Test Split [19]

The train_test_split() method is used to divide our data into training and test sets.

First we need to divide our data into objects (X) and labels (y). The dataframe is divided into X_trail, X_test, y_train and y_test. The X_train and y_train sets are used for training and fitting the model. The X_test and y_test sets are used to test the model if it predicts the correct output/labels. We can explicitly test the size of the train and test suites. It is recommended that our train sets be larger than the test sets. The figure 4 shows clearly how the method works.

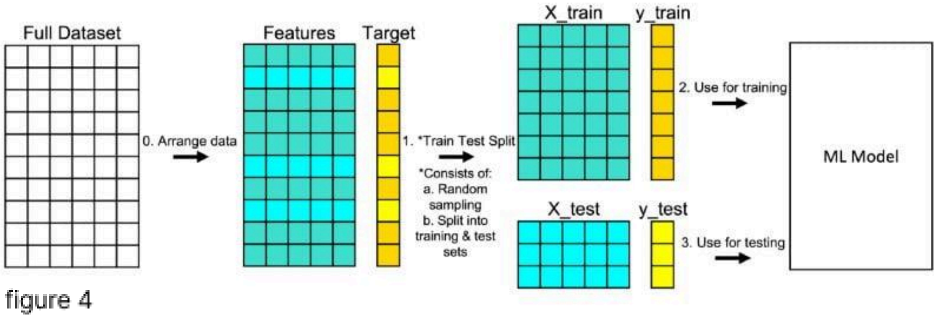


figure 4

Syntax: sklearn.model_selection.train_test_split(*arrays, test_size=None, train_size=None, random_state=None, shuffle=True, stratify=None)

Parameters:

- *arrays: a sequence of indexed objects. Lists, numpy arrays, scipy sparse matrices, and pandas data frames are all valid inputs.
- test_size: int or float, None by default. If the value is float, it should be in the range from 0.0 to 1.0 and represent the percentage of the dataset for the test split. If an int value is used, it refers to the total number of test samples. If the value is None, the addition to the train size is used. The value will be 0.25 if the train size is also None.
- train_size: int or float, None by default.
- random_state : int, not by default. Controls how data is shuffled before splitting is performed. For repeatable output with multiple function calls, pass the int value.
- shuffle: A logical object, the default value is True. Whether to shuffle the data before splitting. The Stratify value must be None if shuffle=False.
- stratify: An array-like object is set to None by default. If None is selected, the data is stratified using them as class labels.

## GridSearchCV

GridSearchCV is a method that accepts a model and various hyperparameter values (a grid of hyperparameters) as input. When searching through the grid, the model and various hyperparameter values are used as input data. Then, for each possible combination of hyperparameter values, the method calculates the error and finally selects the combination in which the error is minimal. Cross-validation is used when training the model[20].

Syntax: class sklearn.model_selection.GridSearchCV(estimator, param_grid, *, scoring=None, n_jobs=None, refit=True, cv=None, verbose=0, pre_dispatch='2*n_jobs', error_score=nan, return_train_score=False)

How does Grid Search CV work? Grid Search CV uses an exhaustive search strategy, systematically examining various combinations of specified hyperparameters and their default values. This approach involves tuning parameters such as learning rate using a cross-validation model that evaluates performance under different parameter settings[21].

## Research and results

The study includes several key stages:

1) Data parsing. First, I parsed text files from folders 20220302-Barl-stems-BT; 20220302-Barl-stems-K; 20220302-Barl-stems-T; 20220302-WiW-stems-BT; 20220302-WiW-stems-K; 20220302-WiW-stems-T.
2) Creating a dataframe and labeling. I created a single dataframe where the rows represented a plant sample and the columns represented the intensity value at a specific wavelength. In this dataframe, I also created a new column to label the data, assigning numeric identifiers to each plant state as follows: barl_bt = 0; barl_k = 1; barl_t = 2; wiw_bt = 3; wiw_k = 4; wiw_t = 5.
3) Data separation. After that, I used the `train_test_split` method to split the data into training and test sets, which allows me to train models on the same dataset and test them on an independent set.
4) Data preprocessing and dimensionality reduction. I have applied the following methods of preprocessing spectral data:`MinMax Scaler` for data normalization and `Standard Scaler` for data standardization. To reduce the dimension, I used `PCA` (Principal Component Analysis), where to select the number of main components, I used the method of the cumulative proportion of the explained variance, striving to preserve at least 95% of the variance in order to avoid excessive noise influence; `LDA` (Linear Discriminant Analysis), where the number of main components was limited to 5, which was determined by the minimum value of the number of classes and the number of features, minus one; `Kernel PCA`, where the number of main components and the choice of the core were determined using `GridSearchCV`; `Isomap`, where the number of main components and the number of neighbors were also determined using `GridSearchCV'. The following hyperparameter values were obtained for dimensionality reduction methods: for PCA(minmax scaled) n_components = 180 and PCA(standard scaled) n_components = 322 ; for LDA(minmax scaled) n_components = 5 and LDA(standard scaled) n_components = 5; for KernalPCA(minmax scaled) n_components = 2304, kernel = linear and KernalPCA(standard scaled) n_components = 3000, kernel = linear; for Isomap(minmax scaled) n_components = 16, n_neighbors = 30 and Isomap(standard scaled) n_components = 32, n_neighbors = 30
5) Choosing machine learning models:

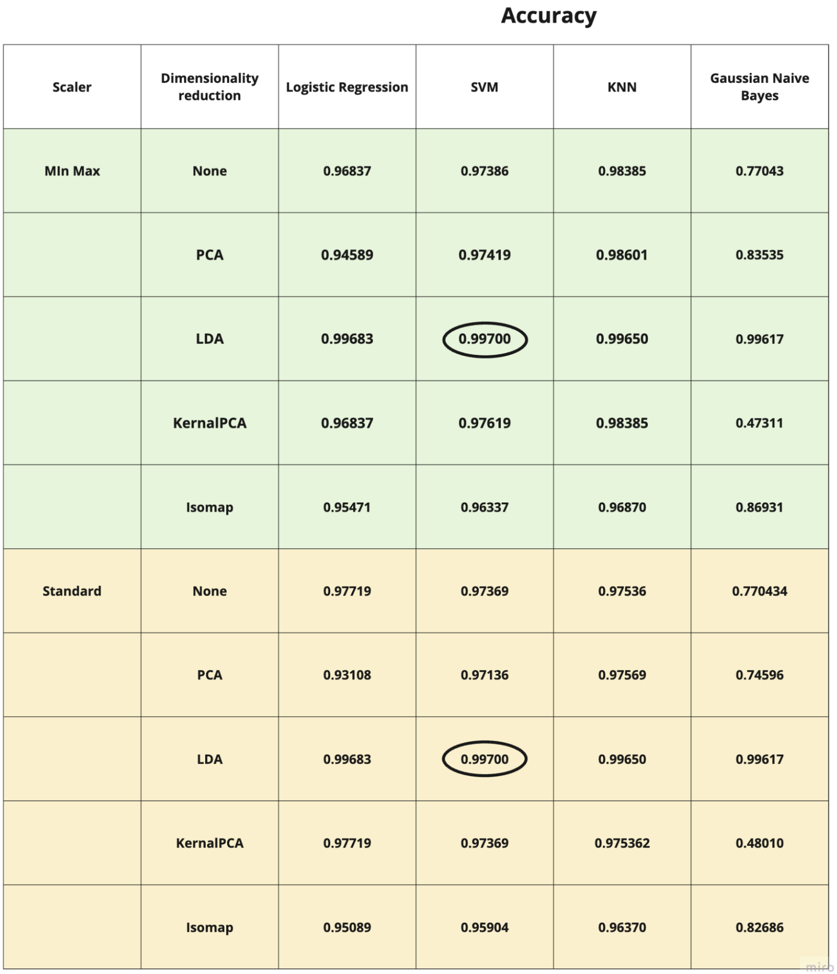
- `Logistic Regression`: The spectral data, after preprocessing and scaling, correspond well to the assumptions of logistic regression about the linear separability of classes in the new feature space. This makes logistic regression an appropriate choice for the initial assessment of class separability.

- `KNN` (K-nearest neighbors): Spectral data contains complex but locally consistent structures, KNN was chosen because of its ability to adapt to such local features without first assuming a global data structure.
- `SVM` (Support Vector Machine): Spectral data are high-dimensional datasets, SVM with its nuclear tricks was chosen to work efficiently with such data, ensuring high accuracy of class separation even in complex cases.
- `Gaussian Naive Bayes`: It was chosen due to its simplicity and efficiency when working with data that can be reasonably approximated by Gaussian distributions, which facilitates the processing of spectral data, where the intensity at certain wavelengths can follow a normal distribution.

6) Choice of metrics. The accuracy metric was used to evaluate the performance of the model. The choice of a metric is determined by several factors:

- Accuracy is one of the simplest and most understandable metrics. It is calculated as the ratio of the number of correctly classified samples to the total number of samples. In the context of plant classification, this means the percentage of samples whose health status and plant species are correctly determined.
- Since each class is represented in approximately equal proportions, using accuracy as a metric becomes effective. In situations where classes are not balanced, the accuracy metric can give a distorted view of the model's performance, since the model can simply predict a larger class and still achieve high accuracy.

7) Validation. Subsequent validation of all combinations of models with data preprocessing and dimensionality reduction methods made it possible to analyze and compare the results to determine the optimal combination of the model with the methods. Table 1 shows the results that were obtained.



Accuracy

| Scaler | Dimensionality reduction | Logistic Regression | SVM | KNN | Gaussian Naive Bayes |
|---|---|---|---|---|---|
| MIn Max | None | 0.96837 | 0.97386 | 0.98385 | 0.77043 |
| | PCA | 0.94589 | 0.97419 | 0.98601 | 0.83535 |
| | LDA | 0.99683 | 0.99700 | 0.99650 | 0.99617 |
| | KernalPCA | 0.96837 | 0.97619 | 0.98385 | 0.47311 |
| | Isomap | 0.95471 | 0.96337 | 0.96870 | 0.86931 |
| Standard | None | 0.97719 | 0.97369 | 0.97536 | 0.770434 |
| | PCA | 0.93108 | 0.97136 | 0.97569 | 0.74596 |
| | LDA | 0.99683 | 0.99700 | 0.99650 | 0.99617 |
| | KernalPCA | 0.97719 | 0.97369 | 0.975362 | 0.48010 |
| | Isomap | 0.95089 | 0.95904 | 0.96370 | 0.82686 |

table 4

The accuracy of the model in the test sample was 0.997 in the StandardScale + LDA + SVM bundle, as well as in the MinMaxScale + LDA + SVM bundle, which indicates a high overall ability of the models to correctly identify the health and species of the plant.

## Conclusion

The methods of dimensionality reduction (PCA, LDA, Kernel PCA, Isomap) were successfully used in the work, which made it possible to improve the quality of classification models. The study confirmed that the use of machine learning to analyze spectral data makes it possible to effectively classify plants into healthy and diseased, achieving a high accuracy of 99.7%. The obtained results open up opportunities for further research in the direction of preparing a model for the detection of various diseases and application on various types of crops, which can radically improve technologies in agriculture.

Considering the limitations of the study, mention should be made of the limited size and diversity of the dataset, which may affect the generalizing ability of the model. In addition, the study focused on certain plant species and diseases, which limits its applicability to other species and conditions.

It is worth mentioning possible improvements, given that the four models considered showed high efficiency, the study of additional machine learning algorithms, such as gradient boosting or deep learning models, may possibly provide even higher classification accuracy. In the future, it is possible to expand the data set to include more plant species and diseases, as well as conduct additional experiments in various climatic conditions. It is also worth exploring the possibility of integrating spectral analysis with other diagnostic methods, for example, using high-resolution images.

## References

1. Savary S. The global burden of pathogens and pests on major food crops [Text] / Savary S., Willocquet L., Pethy- bridge S.J., Esker P., McRoberts N., Nelson A. // Nat. Ecol. Evolut. – 2019., – No. 3. – pp. 430-439.
2. Owomugisha G., Melchert F., Mwebaze E., Quinn J. A., Biehl M. Machine Learning for diagnosis of disease in plants using spectral data [Text] / Owomugisha G., Melchert F., Mwebaze E., Quinn J. A., Biehl M. // Int'l Conf. Artificial Intelligence | ICAI. – 2018. – University of Groningen, Johann Bernoulli Institute for Mathematics and Computer Science, P.O. Box 407, 9700 AK Groningen, The Netherlands; Fraunhofer Institute for Factory Operation and Automation IFF, Sandtorstrasse 22, 39106 Magdeburg, Germany; Makerere University, School of Computing & Informatics Technology P.O. Box 7062 Kampala, Uganda; Busitema University, Faculty of Engineering P. O. Box 236, Tororo, Uganda.
3. Conrad A.O., Li W., Lee D.-Y., Wang G.-L., Rodriguez-Saona L., Bonello P. Machine Learning-Based Presymptomatic Detection of Rice Sheath Blight Using Spectral Profiles [Text] / Conrad A.O., Li W., Lee D.-Y., Wang G.-L., Rodriguez-Saona L., Bonello P. // Department of Plant Pathology and Department of Food Science and Technology, The Ohio State University, Columbus, Ohio, USA. – Received 6 April 2020; Accepted 4 August 2020; Published 12 October 2020
4. Wikiquote/Спектр: [website]. URL: https://ru.wikiquote.org/wiki/%D0%A1%D0%BF%D0%B5%D0%BA%D1%82%D1%80#
5. Geeks for geeks/Data Pre-Processing with Sklearn using Standard and Minmax scaler: [website]. URL: https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/
6. Scikit learn/sklearn.preprocessing.MinMaxScaler: [website]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html
7. Geeks for geeks/Data Pre-Processing with Sklearn using Standard and Minmax scaler: [website]. URL: https://www.geeksforgeeks.org/data-pre-processing-wit-sklearn-using-standard-and-minmax-scaler/
8. Scikit learn/sklearn.preprocessing.StandardScaler: [website]. URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.StandardScaler.html#sklearn.preprocessing.StandardScaler
9. Geeks for geeks/Principal Component Analysis(PCA): [website]. URL: https://www.geeksforgeeks.org/principal-component-analysis-pca/
10. Medium/Understanding Linear Discriminant Analysis (LDA) for Dimensionality Reduction: [website]. URL:

https://medium.com/@conniezhou678/understanding-linear-discriminant-analysis-lda-for-dimensionality-reduction-part-3-d1b5c664b52c

11. Geeks for geeks/Linear Discriminant Analysis in Machine Learning: [website]. URL: https://www.geeksforgeeks.org/ml-linear-discriminant-analysis/?ref=header_search
12. Geeks for geeks/ML | Introduction to Kernel PCA: [website]. URL: https://www.geeksforgeeks.org/ml-introduction-to-kernel-pca/
13. Medium/Kernel PCA. Principal component analysis (PCA) is a…: [website]. URL: https://medium.com/@phsamuel.work/kernel-pca-f7a40f0b4265
14. Geeks for geeks/Isomap | A Non-linear Dimensionality Reduction Technique: [website]. URL: https://www.geeksforgeeks.org/isomap-a-non-linear-dimensionality-reduction-technique/
15. Geeks for geeks/Logistic Regression in Machine Learning: [website]. URL: https://www.geeksforgeeks.org/understanding-logistic-regression/
16. Geeks for geeks/K-Nearest Neighbor(KNN) Algorithm: [website]. URL: https://www.geeksforgeeks.org/k-nearest-neighbours/
17. Ifmo/Метод опорных векторов (SVM): [website]. URL: https://neerc.ifmo.ru/wiki/index.php?title=%D0%9C%D0%B5%D1%82%D0%BE%D0%B4_%D0%BE%D0%BF%D0%BE%D1%80%D0%BD%D1%8B%D1%85_%D0%B2%D0%B5%D0%BA%D1%82%D0%BE%D1%80%D0%BE%D0%B2_(SVM)&mobileaction=toggle_view_desktop#.D0.9C.D0.B5.D1.82.D0.BE.D0.B4_.D0.BE.D0.BF.D0.BE.D1.80.D0.BD.D1.8B.D1.85_.D0.B2.D0.B5.D0.BA.D1.82.D0.BE.D1.80.D0.BE.D0.B2_.D0.B2_.D0.B7.D0.B0.D0.B4.D0.B0.D1.87.D0.B5_.D0.BA.D0.BB.D0.B0.D1.81.D1.81.D0.B8.D1.84.D0.B8.D0.BA.D0.B0.D1.86.D0.B8.D0.B8
18. Geeks for geeks/Gaussian Naive Bayes using Sklearn: [website]. URL: https://www.geeksforgeeks.org/gaussian-naive-bayes-using-sklearn/
19. Geeks for geeks/How To Do Train Test Split Using Sklearn In Python: [website]. URL: https://www.geeksforgeeks.org/how-to-do-train-test-split-using-sklearn-in-python/amp/
20. Ifmo/Настройка гиперпараметров: [website]. URL: https://neerc.ifmo.ru/wiki/index.php?title=Настройка_гиперпараметров
21. Analytics Vidhya/Tune Hyperparameters with GridSearchCV: [website]. URL: https://www.analyticsvidhya.com/blog/2021/06/tune-hyperparameters-with-gridsearchcv/