Title: Fruit Quality Assessment using Machine Learning Model

Author: Naphat Mahittikul

Date: April 30, 2024

Institute: Chulalongkorn University

Contact Information: naphatmahit@gmail.com

Abstract

This project aims to develop classification machine learning models for predicting fruit quality (good or bad) based on seven key features: size, weight, sweetness, crunchiness, juiciness, ripeness, and acidity. The target variable, quality, is assigned a value of 1 for good quality fruit and 0 for bad quality fruit. During model evaluation, the tuned Neural Network demonstrated the best performance with the highest accuracy (0.9512) on the validation dataset compared to Logistic Regression and SVM models. The Neural Network's success can be attributed to its ability to capture complex relationships within the data, potentially due to its non-linear activation functions. Despite a slight decrease in accuracy on unseen data from the test set, the Neural Network model's performance (0.9462) remains promising, indicating its ability to generalize well. However, the evaluation process primarily focused on accuracy, and future work may explore additional evaluation metrics such as precision, recall, and F1-score. Addressing limitations such as the unknown absolute values of certain features and limited knowledge about the data's background could further enhance the model's generalizability. Overall, the developed model holds significant potential for improving fruit grading, reducing food waste, and enhancing customer experience in real-world applications.

Keywords: fruit quality prediction, classification, machine learning, Neural Network, Logistic Regression, SVM, evaluation metrics, generalizability.

Table of Contents

[Table of Figures 3](#_Toc165471217)

[Table of Tables 3](#_Toc165471218)

[1. Problem Statement 4](#_Toc165471219)

[2. Literature Review 5](#_Toc165471220)

[Libraries 5](#_Toc165471221)

[Models 5](#_Toc165471222)

[Tools 6](#_Toc165471223)

[3. Methodology 7](#_Toc165471224)

[3.1 Data Preparation 7](#_Toc165471225)

[3.2 Creating machine learning model 11](#_Toc165471226)

[3.2.1 Logistic Regression 11](#_Toc165471227)

[3.2.2 Support Vector Machine (SVM) 13](#_Toc165471228)

[3.2.3 Neural networks 14](#_Toc165471229)

[3.3 Metric assessment on testing dataset 14](#_Toc165471230)

[4. Results and Discussion 15](#_Toc165471231)

[4.1 Results from model 15](#_Toc165471232)

[4.1.1 Logistic Regression result 15](#_Toc165471233)

[4.1.2 Support Vector Machine 16](#_Toc165471234)

[4.1.3 Neural Network 17](#_Toc165471235)

[4.2 Comparing between model 18](#_Toc165471236)

[4.3 Discussion 18](#_Toc165471237)

[References 19](#_Toc165471238)

# Table of Figures

[Figure 1. Data dimension 7](#_Toc165471239)

[Figure 2. Data type of each column 7](#_Toc165471240)

[Figure 3. Descriptive Statistics of dataset 7](#_Toc165471241)

[Figure 4. Number of missing data in each column 8](#_Toc165471242)

[Figure 5. Each fruit quality count 8](#_Toc165471243)

[Figure 6. Histogram and boxplot of features 9](#_Toc165471244)

[Figure 7. Correlation Matrix of Features 10](#_Toc165471245)

[Figure 8. Learning Curve of Logistic Regression Model 16](#_Toc165471246)

[Figure 9. Learning Curve of Support Vector Machine 17](#_Toc165471247)

# Table of Tables

[Table 1. Results from Logistics Regression Model 15](#_Toc165471248)

[Table 2. Result from SVM 16](#_Toc165471249)

[Table 3. Results from Neural Network 17](#_Toc165471250)

[Table 4. Table comparing accuracy 18](#_Toc165471251)

# 1. Problem Statement

This project focuses on creating classification machine learning models to predict fruit quality (good or bad) from a dataset which has seven features: size, weight, sweetness, crunchiness, juiciness, ripeness, and acidity. Each attribute can be described as:

Features

* Size: Size of the fruit
* Weight: Weight of the fruit
* Sweetness: Degree of sweetness of the fruit
* Crunchiness: Texture indicating the crunchiness of the fruit
* Juiciness: Level of juiciness of the fruit
* Ripeness: Stage of ripeness of the fruit
* Acidity: Acidity level of the fruit

Target Variable: Quality – 1 if the fruit quality is good, 0 if the fruit quality is bad.

After getting model which can accurately determine the quality of given fruit, we can potentially utilize the model for these benefits:

* Improving fruit grading
* Reducing food waste
* Enhancing customer experience

# 2. Literature Review

In the project, machine learning techniques were applied to develop predictive models. Machine learning involves the construction of algorithms that allow computers to learn patterns from data and make predictions or decisions without explicit programming. This field encompasses various algorithms and methods, such as supervised learning, unsupervised learning, and reinforcement learning [1].

The project addressed a classification problem where the objective was to categorize data points into distinct classes or categories. Classification problems are a common task in machine learning, and they involve predicting a discrete label or category based on input features. Various algorithms, such as logistic regression, support vector machines (SVM), and neural networks, can be used to solve classification problems depending on the nature of the data and the problem at hand [1].

## Libraries

Pandas and NumPy were utilized extensively for data preprocessing, manipulation, and analysis. NumPy provides essential functions for working with arrays and matrices, while Pandas offers data structures like DataFrames that facilitate data handling and manipulation. These libraries are fundamental tools in the Python ecosystem for tasks ranging from data cleaning to exploratory data analysis [2].

Matplotlib and Seaborn are Python libraries commonly used for data visualization. Matplotlib provides a wide range of plotting functions, while Seaborn offers a high-level interface for creating informative statistical graphics [3].

TensorFlow is an open-source machine learning framework developed by Google for building and training deep learning models. Scikit-learn is a versatile Python library for machine learning tasks, including classification, regression, clustering, and dimensionality reduction [4].

Optuna is a powerful hyperparameter optimization framework in Python. It offers an intuitive and efficient way to search for the best hyperparameters of machine learning models. Optuna uses various algorithms, including Bayesian optimization, to explore the hyperparameter space efficiently and find the optimal configuration for the given objective function [5].

## Models

Logistic regression is a statistical method used for binary classification tasks. It models the probability that a given input belongs to a particular class [6].

SVM (Support Vector Machine) is a supervised machine learning algorithm used for classification and regression tasks. It works by finding the optimal hyperplane that separates data points into different classes, maximizing the margin between classes. SVM can handle linear and non-linear data separations using different kernel functions [7].

Neural networks are computational models inspired by the structure and functioning of the human brain. They consist of interconnected nodes organized in layers, including input, hidden, and output layers. Neural networks are capable of learning complex patterns from data and are widely used for tasks such as classification, regression, and pattern recognition [8].

## Tools

The Elbow Method is a technique used to determine the optimal number of clusters in a dataset for clustering algorithms such as K-means. It involves plotting the within-cluster sum of squares (WCSS) against the number of clusters and identifying the "elbow" point, where the rate of decrease in WCSS slows down. The number of clusters at the elbow point is considered optimal [9].

StandardScaler is a preprocessing technique used to standardize features by removing the mean and scaling to unit variance. It transforms the data distribution to have a mean of 0 and a standard deviation of 1, ensuring that features are on a similar scale. StandardScaler is commonly used before applying machine learning algorithms to ensure that all features contribute equally to the model [6].

K-fold cross-validation is a resampling technique used to assess the performance of machine learning models. It involves dividing the dataset into k subsets, training the model on k-1 subsets, and evaluating it on the remaining subset. This process is repeated k times, with each subset serving as the validation set exactly once. K-fold cross-validation provides a more reliable estimate of model performance compared to a single train-test split [10].

Data imbalance occurs when one class in the dataset has significantly fewer samples than another. It can lead to biased models that perform poorly in the minority class. Techniques to address this problem include resampling methods like oversampling and undersampling, using algorithms robust to class imbalance, and performance metrics tailored to imbalanced datasets [11].

A correlation matrix is a table showing correlation coefficients between variables in a dataset. It helps identify relationships between variables, indicating how changes in one variable relate to changes in another. Correlation matrices are useful for feature selection, identifying multicollinearity, and understanding the underlying structure of the data [12].

# 3. Methodology

## 3.1 Data Preparation

This project will follow a methodology to develop and evaluate machine learning models for fruit quality prediction. The EDA stage:

1. Data Import and Loading: The process begins by importing the provided fruit quality dataset. Pandas library will be used to read the data into a DataFrame.
2. Data Inspection: The DataFrame will be examined to understand its dimensions (number of rows and columns). Then, the data types of each column will be verified to ensure proper handling during analysis.

A screenshot of a computer code

Description automatically generated

Figure 1. Data dimension

A screenshot of a computer

Description automatically generated

Figure 2. Data type of each column

1. Descriptive Statistics: Descriptive statistics will be computed to summarize the central tendency (mean, median, Q1, Q2, Q3) and standard deviation of all features.

A screenshot of a computer

Description automatically generated

Figure 3. Descriptive Statistics of dataset

1. Missing Value Analysis: The presence of missing values (NA values) will be investigated. Depending on the extent and distribution of missing data, appropriate imputation techniques may be applied to address them.

A screenshot of a computer

Description automatically generated

Figure 4. Number of missing data in each column

1. Target Variable Analysis: The value counts for the target variable (fruit quality) will be calculated. This analysis provides insights into the class imbalance (ratio of good vs. bad quality fruits) which can influence the choice of classification models and evaluation metrics.

A graph of a fruit quality

Description automatically generated

Figure 5. Each fruit quality count

Figure 6 shows that from all the data collected, 2004 was good quality and 1996 was bad. It is safe to say that this data will not face imbalance classification problem.

1. Data Visualization: Data visualization plays a crucial role in understanding the relationships between features. Histograms will be employed to visualize the distribution of each numerical feature, revealing potential skewness or outliers. Boxplots will be used to compare the distribution of features across the two quality classes (good and bad).

A blue and black graph

Description automatically generated with medium confidenceA blue and black graph

Description automatically generated with medium confidence

Figure 6. Histogram and boxplot of features

Apart from crunchiness, all features (based on the histogram) appear to fall within a nearly identical value range following the EDA step. implying that the data may already have been normalized and cleaned. Additionally, based on box plot analysis, certain characteristics of low-quality fruit have a narrower range than those of high-quality fruit. However, there isn't much of a difference in its census.

1. Correlation Analysis: Correlation Matrix is used to evaluate the correlation between each feature. This matrix helps identify the correlation and detect the correlation problem which will cause the issues when creating model.

A screenshot of a graph

Description automatically generated

Figure 7. Correlation Matrix of Features

From figure 7, it shows that there is at most weak correlation between features, if not no correlation at all.

## 3.2 Creating machine learning model

In this project, aiming to develop machine learning models to predict the quality of fruit (good or bad) based on a dataset containing various measurable attributes. While traditional methods rely on visual inspection, this project explores a more objective and automated approach.

The choice of models incorporates Logistic Regression, Support Vector Machines (SVM), and Neural Networks. Logistic Regression provides a solid foundation for binary classification tasks like this one. SVM is adept at handling non-linear relationships between features, which might exist in the fruit quality data. Finally, Neural Networks offer exceptional flexibility in modeling complex relationships, potentially capturing intricate patterns within the data that simpler models might miss. By evaluating these diverse models, we can determine the most effective approach for accurate fruit quality prediction.

Accuracy is the initial choice for evaluating the Logistic Regression model's performance. This metric simply reflects the proportion of fruit quality predictions (good or bad) that align with the actual labels. And since the dataset contains roughly equal numbers of good and bad quality fruit samples, accuracy directly reflects the model's ability to correctly classify both categories. In balanced data, a high accuracy score indicates the model performs well in identifying both good and bad quality fruits.

### 3.2.1 Logistic Regression

The baseline Logistic Regression model employs a baseline configuration where all features are included with their default hyperparameter values. This serves as a starting point for evaluating model performance.

1st model - Baseline Model:

* All features are included in the model.
* Default values are used for all hyperparameters. This serves as a starting point for further optimization.

2nd model - Grid Search for Initial Hyperparameters:

A grid search is performed to find a good initial set of hyperparameters for the logistic regression model. The grid search explores different values for the following hyperparameters:

* 'C': This is the regularization parameter that controls the trade-off between model complexity and fitting the training data. The grid search considers a range of C values (0.001, 0.01, 0.1, 1, 10, 100)
* 'solver': This specifies the algorithm used to solve the logistic regression problem. We consider 'lbfgs', 'liblinear', 'newton-cg', 'newton-cholesky', 'sag', and 'saga' to be evaluated in the grid search.

Polynomial Feature Selection: A loop iterates through polynomial degrees from 1 to 4.

Inside the loop:

* The 'liblinear' solver is used for logistic regression with polynomial features of the current degree.
* The mean squared error (MSE) is calculated on a 5-fold cross-validation.
* The calculated MSE is stored in a list for later analysis.

After the loop is completed, a plot is created to visualize the relationship between polynomial degree and the average MSE across cross-validation folds. This helps identify the polynomial degree that leads to the lowest average error.

3rd model - Final Model Tuning is based on the results from step 3, the optimal polynomial degree is chosen. A new grid search is performed using the chosen polynomial degree and the 'liblinear' solver (since it only has L2 regularization).

* 'C': The regularization parameter with a wider range from 0.001 to 10000 for more precise tuning.
* 'max\_iter': The maximum number of iterations for the solver, exploring values of 100, 500, and 1000 to ensure convergence.
* The accuracy is calculated using 5-fold cross-validation for each hyperparameter combination.

The hyperparameter combination that leads to the highest average accuracy is selected as the final model.

### 3.2.2 Support Vector Machine (SVM)

Data Preprocessing: Standardize the data using a standard scaler. This ensures all features are on a similar scale, improving the performance of some kernel functions.

1st model - Baseline model: SVM model which hyperparameter is not tuned. All the hyperparameter are default values.

2nd model - Initial Hyperparameter Tuning: A grid search is performed to find a good starting point for the SVM hyperparameters. The grid search explores different values for the following hyperparameters:

* 'C': The regularization parameter that controls the complexity of the model. We try values of 0.1, 1, and 10 to cover a range of regularization strengths.
* 'kernel': This specifies the type of kernel function used by the SVM. We explore four options: 'linear', 'rbf', 'poly', and 'sigmoid'. Depending on the chosen kernel, additional hyperparameters might be relevant.
* 'gamma': This hyperparameter controls the influence of training points in the RBF kernel. We try values of 0.01, 0.1, and 1 to explore different levels of influence.

The accuracy of each hyperparameter combination is evaluated using 5-fold cross-validation. The combination with the highest average accuracy across folds is selected as the initial model.

Polynomial Feature Exploration: A loop iterates through polynomial degrees from 1 to 4.

Inside the loop:

* The SVM model is trained with a polynomial kernel of the current degree.
* The accuracy is calculated using 5-fold cross-validation.
* The average accuracy across folds is stored for later analysis.

After the loop is completed, a plot is created to visualize the relationship between polynomial degree and the average accuracy. This helps identify the polynomial degree that leads to the highest average accuracy.

3rd model - Final Model Tuning:

Based on the results from step 3, the optimal polynomial degree (if applicable) is chosen.

A new grid search is performed using the chosen kernel function (including polynomial if applicable) and the following hyperparameters:

* ‘C’: The regularization parameter with the same range as before (0.1, 1, 10).
* ‘kernel': The chosen kernel function from step 2 or 3.
* 'gamma': The hyperparameter for RBF kernel with the same range as before (0.01, 0.1, 1).

The accuracy is calculated using 5-fold cross-validation for each hyperparameter combination. The hyperparameter combination that leads to the highest average accuracy is selected as the final SVM model.

### 3.2.3 Neural networks

1st model - Baseline Model:

* A simple neural network with one hidden layer is created as a baseline.
* The model is trained with 50 epochs and a batch size of 32 to get a preliminary understanding of the achievable accuracy.

2ndmodel - Hyperparameter Optimization with Optuna:

* Optuna is used to optimize the hyperparameters of the neural network for better performance. The following hyperparameters are tuned within specified ranges:
  + 'units': The number of neurons in the hidden layer. This is searched as integers from 16 to 192 with a step size of 16.
  + 'dropout\_rate': The dropout rate for regularization. This is explored as floats between 0.0 and 0.5 with a step size of 0.1.
  + 'learning\_rate': The learning rate for the optimizer. Currently, it is in the range of 1e-2, 1e-3, and 1e-4.
  + 'num\_layers': The number of hidden layers in the network. This is searched as integers from 1 to 5.
* Noted that the number of hidden layers, units, number of layers are keeping low to prevent overfitting due to its nature of this model is prone to be overfit.

3rd model - Evaluation of Optimized Model:

* After Optuna finds the best hyperparameter configuration, the neural network is retrained with those parameters.
* The accuracy of the optimized model is evaluated using accuracy to assess its performance.

## 3.3 Metric assessment on testing dataset

Following the evaluation of individual models (Logistic Regression, SVM, and Neural Network), their performance will be compared using accuracy as the primary metric. Accuracy reflects the overall proportion of correctly predicted fruit quality labels (good or bad) on the testing set. The model that achieves the highest accuracy on the testing data will be deemed the most effective for fruit quality prediction in this context.

This approach prioritizes overall classification accuracy, ensuring the chosen model excels at correctly identifying both good and bad quality fruits. By evaluating on the unseen testing set, we obtain a more reliable assessment of the model's generalizability to real-world scenarios beyond the training data. The model with the strongest performance on this final test becomes our champion, ready for potential deployment in a fruit quality assessment system.

# 4. Results and Discussion

## 4.1 Results from model

### 4.1.1 Logistic Regression result

The baseline model's initial hyperparameter values are all set to default, allowing for the prediction of the model's expected performance. To ensure consistency in the outcome, the random state is set to 42. Additionally, the average accuracy from K-fold cross validation on the data (K=5) is used to calculate the accuracy of the models that were developed.

Table 1. Results from Logistics Regression Model

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Solver** | **Penalty** | **C**  **(penalty strength)** | **Max iteration** | **Degree of polynomial** | **Model accuracy** |
| 1st Baseline model (default) | lbfgs | l2 | 1.0 | 100 | 1 | 0.7262 |
| 2nd Grid search for hyperparameter  tuning | lbfgs | l2 | 0.01 | 100 | 1 | 0.7312 |
| 3rd Final model | liblinear | l1 | 0.1 | 100 | 3 | 0.8850 |

The learning curve, which is depicted in Figure 8, is used to move from the second model to the third model in order to ascertain which degree of polynomial should be employed in the model. As the elbow point with the lowest loss among the four degree polynomials tested in this method, the elbow method can indicate that the number three is the ideal point. Figure 8's loss is computed using mean squared error (MSE).

A graph with a line going up

Description automatically generated

Figure 8. Learning Curve of Logistic Regression Model

The final model, which is 0.8850 of the trained model, has the highest accuracy, according to the results of the accuracy check. Next, the model produces an average accuracy of 0.8942 after k-fold cross validation accuracy checking at k=5, which is objectively better than the accuracy of the baseline model (accuracy = 0.7262).

### 4.1.2 Support Vector Machine

SVM suggests first normalize the data for this model before moving forward. The data is normalized using a standard scaler before being entered into a model with a random state of 42 to produce results that are reliable. This baseline model phase does not modify the hyperparameters. The average accuracy from K-fold cross validation on the data (K=5) is used to calculate the accuracy of this model.

Table 2. Result from SVM

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Model** | **Kernel** | **C** | **Gamma** | **Degree of polynomial** | **Accuracy** |
| 1st Baseline model | rbf | 1.0 | \*default\* | 1 | 0.8854 |
| 2nd Grid search for hyperparameter  tuning | rbf | 1.0 | 1.0 | 1 | 0.8992 |
| 3rd Final Model | rbf | 1.0 | 1.0 | 3 | 0.9037 |

A learning curve is used to identify the degree of polynomial that is appropriate for the mode between the second and third models. Figure 9 illustrates the degree of polynomial of 3 with the lowest MSE obtained using the elbow approach. As a result, the polynomial degree in the finished model will be 3.

A graph with a line and a line

Description automatically generated

Figure 9. Learning Curve of Support Vector Machine

The final model, which is 0.9037 of the training model, has the highest accuracy, according to the results of the accuracy check. Next, after examining the k-fold cross validation accuracy at k=5, the model produces an average accuracy of 0.8992, which is demonstrably better than the accuracy of the baseline model (accuracy = 0.8854).

### 4.1.3 Neural Network

In order to prevent overfitting, start with the baseline model, which consists of an input layer, an output layer, and one hidden layer with a dropout layer. Adam is the model's optimizer; the loss function is binary cross entropy; the accuracy metric must be tuned; there are 50 epochs and 32 batches. This model has an accuracy of 0.8788 with a loss of 0.2826. The output layer's activation function is sigmoid, and the activation function for all node is relu.

Table 3. Results from Neural Network

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Model** | **Epochs** | **Batch Size** | **Units** | **Dropout rate** | **Learning rate** | **Number of hidden layer** | **Accuracy** |
| 1st Baseline model | 50 | 32 | 16 | 0.2 | 0.01 | 1 | 0.8788 |
| 2nd Model with early stopping | 150 | 32 | 64 | 0.2 | 0.01 | 1 | 0.9237 |
| 3rd Model tuned from optuna | 50 | 32 | 144 | 0.5 | 0.01 | 2 | 0.9512 |

The metric that is selected at the outset of the process allows for a comparison of each model's top features based on accuracy. The optimized neural network model, based on validated data, achieves the highest accuracy of 0.9512. Utilizing the test dataset, the accuracy is computed in the last step. The trained neural network's accuracy on test datasets is 0.9462. which, considering its small accuracy differential, is practical.

## 4.2 Comparing between model

Table 4. Table comparing accuracy

|  |  |
| --- | --- |
| **Model** | **Accuracy** |
| Logistics Regression | 0.8946 |
| SVM | 0.9037 |
| Neural Network | 0.9512 |

From all the best of each model, it can be compared with accuracy, which is the metric that is chosen from the start of methodology. Result is tuned neural network model has the best accuracy of 0.9512 from the validate data.

The last step is to calculate the accuracy using the test dataset. The accuracy of the tuned neural network on test datasets is 0.9462. Which is usable due to its negligible difference on accuracy.

## 4.3 Discussion

As highlighted in Table 4, the tuned Neural Network model emerged as the champion with the highest accuracy (0.9512) on the validation data. This indicates its superior ability to correctly classify fruit quality (good or bad) compared to Logistic Regression (0.8946) and SVM (0.9037). The Neural Network's success can be attributed to its flexibility in capturing complex relationships within the data, potentially due to its non-linear activation functions and ability to learn intricate feature representations.

The final evaluation involved testing the tuned Neural Network model on unseen data from the test set. Despite a slight decrease in accuracy (0.9462 compared to 0.9512 on the validation set), the performance remains highly promising. This minor difference suggests the model generalizes well to unseen data, making it a strong candidate for real-world fruit quality prediction tasks.

While this project achieved promising results, some limitations offer avenues for future exploration. The evaluation relied primarily on accuracy, and incorporating additional metrics like precision, recall, and F1-score could provide a more comprehensive understanding of model performance. Additionally, while the data underwent cleaning procedures, the absolute values used in certain features remain unknown. Furthermore, limited knowledge regarding the data's background and the specific methods used to measure the features could potentially influence model generalizability. Future work can address these limitations by exploring a broader range of evaluation metrics, potentially delving deeper into the data's origins and measurement techniques.

# References

1. Penmetsa, C.C. Machine learning Classification Problem end-to-end example using Scikit-Learn, Pandas, NumPy and Matplotlib. Available online: <https://medium.com/codenx/machine-learning-classification-problem-end-to-end-example-using-scikit-learn-pandas-numpy-and-ea47104eddd5> (accessed on April 30, 2024).

2. Toomet, O. Machine learning in python. Available online: <https://faculty.washington.edu/otoomet/machinelearning-py/numpy-and-pandas.html> (accessed on April 30, 2024).

3. Galarnyk, M. Logistic Regression using Python (scikit-learn). Available online: <https://towardsdatascience.com/logistic-regression-using-python-sklearn-numpy-mnist-handwriting-recognition-matplotlib-a6b31e2b166a> (accessed on April 30, 2024).

4. TensorFlow. Logistic regression for binary classification with Core APIs : TensorFlow Core. Available online: <https://www.tensorflow.org/guide/core/logistic_regression_core> (accessed on April 30, 2024).

5. Akiba, T.; Sano, S.; Yanase, T.; Ohta, T.; Koyama, M. Optuna: A Next-generation Hyperparameter Optimization Framework. In Proceedings of the Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 2019; pp. 2623–2631.

6. Bronwlee, J. How to Fix k-Fold Cross-Validation for Imbalanced Classification. Available online: <https://machinelearningmastery.com/cross-validation-for-imbalanced-classification/> (accessed on April 30, 2024).

7. Wang, Z.; Zhan, Q.; Tong, B.; Yang, S.; Hou, B.; Huang, H.; Saykin, A.; Thompson, P.; Davatzikos, C.; Shen, L. Distance-Weighted Sinkhorn Loss for Alzheimer’s Disease Classification. *iScience* **2024**, *27*, 109212, doi:10.1016/j.isci.2024.109212.

8. Karani, D. Optuna Guide: How to Monitor Hyper-Parameter Optimization Runs. Available online: <https://neptune.ai/blog/optuna-guide-how-to-monitor-hyper-parameter-optimization-runs> (accessed on April 30, 2024).

9. Turruellas, J.C.O. A Comprehensive Guide to Stratified K-Fold Cross-validation for Unbalanced Data. Available online: <https://www.linkedin.com/pulse/comprehensive-guide-stratified-k-fold-unbalanced-data-juan-carlos-n6aee/> (accessed on April 30, 2024).

10. Barnaghi, P. Machine Learning for Beginners Tutorial and Assessment/Machine Learning for Beginners. Available online: <https://github.com/PBarnaghi/ML4NS/blob/main/00-%20Tutorials/Machine%20Learning%20for%20Beginners%20Tutorial%20and%20Assessment/Machine%20Learning%20for%20Beginners%20(run).ipynb> (accessed on April 30, 2024).

11. Akiode, A. How to carry out k-fold cross-validation on an imbalanced classification problem. Available online: <https://medium.com/analytics-vidhya/how-to-carry-out-k-fold-cross-validation-on-an-imbalanced-classification-problem-6d3d942a8016> (accessed on April 30, 2024).

12. Manunza, G. German Credit Dataset Analysis. Available online: <https://www.kaggle.com/code/gilbertomanunza/german-credit-dataset-analysis/notebook> (accessed on April 30, 2024).