Co-funded by the European Union

MODULE 6/Machine Learning in R

Carlo Drago PhD

University Niccolo Cusano, Rome

MASSIVE OPEN ONLINE COURSE (MOOC)

Project N. 2023-1-IT02-KA220-HED-000161770
ANALYST - A New Advanced Level for Your Specialised Training

Co-funded by
the European Union

UNSUPERVISED LEARNING IN R

Unsupervised learning in R involves techniques used to analyze and cluster unlabeled datasets. These methods help in identifying patterns or structures without predefined labels. Common unsupervised learning techniques in R include:

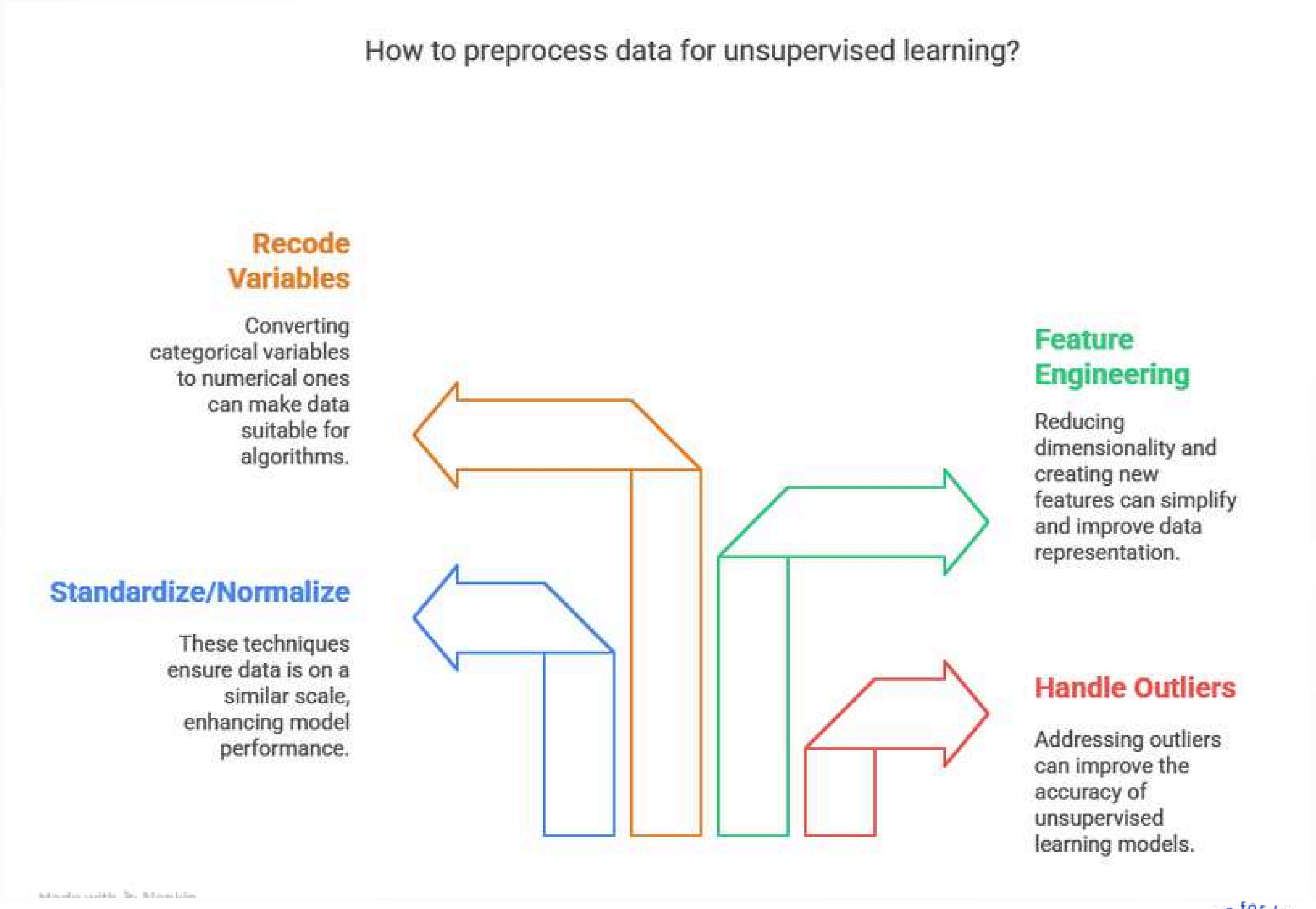
1. **Clustering:** This involves grouping data points into clusters based on their similarities. Popular clustering methods in R include K-means, Hierarchical Clustering, and DBSCAN. The `stats` package provides functions like `kmeans()` for K-means clustering, while `hclust()` is used for hierarchical clustering.

Co-funded by
the European Union

UNSUPERVISED LEARNING IN R

2. **Principal Component Analysis (PCA):** PCA is used for dimensionality reduction, helping to simplify data while preserving as much variance as possible. The `prcomp()` function in R is often used for PCA.

PREPROCESSING FOR UNSUPERVISED LEARNING



How to preprocess data for unsupervised learning?

Recode Variables

Converting categorical variables to numerical ones can make data suitable for algorithms.

Standardize/Normalize

These techniques ensure data is on a similar scale, enhancing model performance.

Feature Engineering

Reducing dimensionality and creating new features can simplify and improve data representation.

Handle Outliers

Addressing outliers can improve the accuracy of unsupervised learning models.

Made with Napkin

DISTANCES IN R

When performing clustering in R, understanding distances is crucial as they determine the similarity or dissimilarity between data points. In clustering, distance metrics are used to group similar data points together into clusters. Here are some common distance measures used in clustering within R:

1. **Euclidean Distance:** This is the most common distance measure and calculates the straight-line distance between two points in Euclidean space. It is suitable for continuous data and is often used in methods like K-means clustering.

DISTANCES IN R

2. **Manhattan Distance:** Also known as the L1 distance, it calculates the distance between two points by summing the absolute differences of their coordinates. This distance is useful when the data is sparse or when dealing with categorical data.

3. **Minkowski Distance:** A generalization of both Euclidean and Manhattan distances. By adjusting the order parameter 'p', you can switch between different types of distances. For example, when p=2, it becomes the Euclidean distance, and when p=1, it becomes the Manhattan distance.

Co-funded by
the European Union

DISTANCES IN R

4. **Cosine Similarity/Dissimilarity:** Often used in text mining and information retrieval, cosine similarity measures the cosine of the angle between two non-zero vectors. It is a measure of orientation rather than magnitude, making it useful when the magnitude of vectors varies significantly.

5. **Jaccard Distance:** Jaccard similarity measures how much two sets have in common by dividing the number of shared elements by the total number of elements in either set. That ratio always falls between 0 (no overlap) and 1 (identical sets). The Jaccard distance is simply one minus this similarity score, so identical sets have distance 0 and completely disjoint sets have distance 1.

Co-funded by
the European Union

DISTANCES IN R

In R, these distances can be computed **using functions from various packages**. For example, the `dist()` function in base R can calculate Euclidean, Manhattan, and Minkowski distances. The `proxy` package offers a wider range of distance and similarity metrics, while the `stats` package provides clustering algorithms that can utilize these distances.

Choosing the appropriate distance measure is crucial as it influences the clustering results. The choice often depends on the nature of the data and the specific characteristics that are important for the analysis.

K-MEANS

K-Means is a popular clustering algorithm used in data analysis. It aims to group a set of data points into a certain number of clusters defined by their centers. K-Means is a widely used clustering algorithm in data analysis that groups data points into predefined clusters based on their centers. The process involves:

- **Initialization:** Randomly selecting 'k' initial centroids.
- **Assignment step:** Assigning each data point to the nearest centroid to form clusters, typically using Euclidean distance.

See Witten et al. (2013)

K-MEANS IN R

To implement K-means clustering in R, it is necessary to follow these steps:

1. **Data Preparation:** it is necessary to ensure that the data is in a suitable format, typically as a data frame or matrix. Standardize the data if necessary to ensure that each feature contributes equally to the distance calculations.

2. **Choice of the Number of Clusters (K):** it is important to determine the number of clusters to identify in data. This can be done using methods like the elbow method, silhouette analysis, or domain knowledge.

K-MEANS IN R

3. **Use the `kmeans()` Function:** R provides a built-in function, `kmeans()`, to perform K-means clustering. The function requires at least two arguments: the data and the number of clusters (K).

```R
set.seed(123)  # Set seed for reproducibility
kmeans_result <- kmeans(your_data, centers = K)
```

K-MEANS IN R

4. **Analysis of the Output:** The `kmeans()` function returns a list with several components, such as:
   - `centers`: The coordinates of the cluster centers.
   - `cluster`: A vector indicating the cluster assignment for each observation.
   - `tot.withinss`: The total within-cluster sum of squares.
   - `size`: The number of points in each cluster.

K-MEANS IN R

5. **Visualization of the Clusters:** the use of the visualization techniques is necessary to understand clustering results better. It is possible to plot the data with the clusters highlighted. For simple two-dimensional data, the `ggplot2` package can be used.

~~~
   ```R
   library(ggplot2)
   ggplot(your_data, aes(x = feature1, y = feature2, color = as.factor(kmeans_result$cluster))) +
     geom_point() +
     geom_point(data = as.data.frame(kmeans_result$centers), aes(x = V1, y = V2), color =
'red', size = 3, shape = 4)
~~~

Co-funded by the European Union

K-MEANS IN R

6. **Evaluating the Clusters:** Assessing the quality of the clustering using metrics like the silhouette score or examining the within-cluster sum of squares.

By following these steps, it is possible to implement and analyze the results of K-means clustering in R effectively.

I

Co-funded by
the European Union

HIERARCHICAL CLUSTERING IN R

Hierarchical clustering is a cluster analysis technique that creates a hierarchy of clusters using either an agglomerative (bottom-up) or divisive (top-down) approach. The agglomerative method starts with each data point as a separate cluster and gradually merges them as it moves up the hierarchy. Conversely, the divisive approach begins with one large cluster and divides it into smaller subclusters.

For further details, please consult Witten et al. (2013).

Co-funded by the European Union

HIERARCHICAL CLUSTERING IN R

Hierarchical clustering in R is a statistical method used to group similar objects into clusters. This technique builds a hierarchy of clusters, which can be visualized as a tree-like diagram called a dendrogram. In R, hierarchical clustering can be performed using the `hclust()` function, which requires a dissimilarity matrix as input. This matrix is often created using the `dist()` function to calculate distances between data points.

Co-funded by
the European Union

HIERARCHICAL CLUSTERING IN R

There are two main types of hierarchical clustering: **agglomerative and divisive**. Agglomerative clustering starts with each object as a separate cluster and merges them step by step, while divisive clustering begins with all objects in one cluster and splits them iteratively. The `hclust()` function in R primarily supports agglomerative clustering, with various linkage methods like single, complete, average, and ward.D, which determine how the distance between clusters is calculated.

HIERARCHICAL CLUSTERING IN R

To visualize the results, you can use the `plot()` function to create a **dendrogram**. This visualization helps in deciding the number of clusters by cutting the tree at a specific level using the `cutree()` function, which assigns each object to a cluster.

Hierarchical clustering in R is particularly useful when you need to understand the relationships between data points and how they group together at different levels of granularity.

SUPERVISED LEARNING IN R

Supervised learning refers to a machine learning method where an algorithm is trained on a dataset that includes the correct answers.

This approach contrasts with unsupervised learning, where algorithms analyze data that is not labeled.

For further reading, refer to Bishop (2006) and Witten et al. (2013).

SUPERVISED LEARNING IN R

Supervised learning algorithms acquire knowledge by identifying patterns within the data (the training set) and correlating inputs with the appropriate outputs. After training, these algorithms can apply their learned insights to make predictions or classify new, unseen data (the test set).

Refer to Bishop (2006) and Witten et al. (2013).

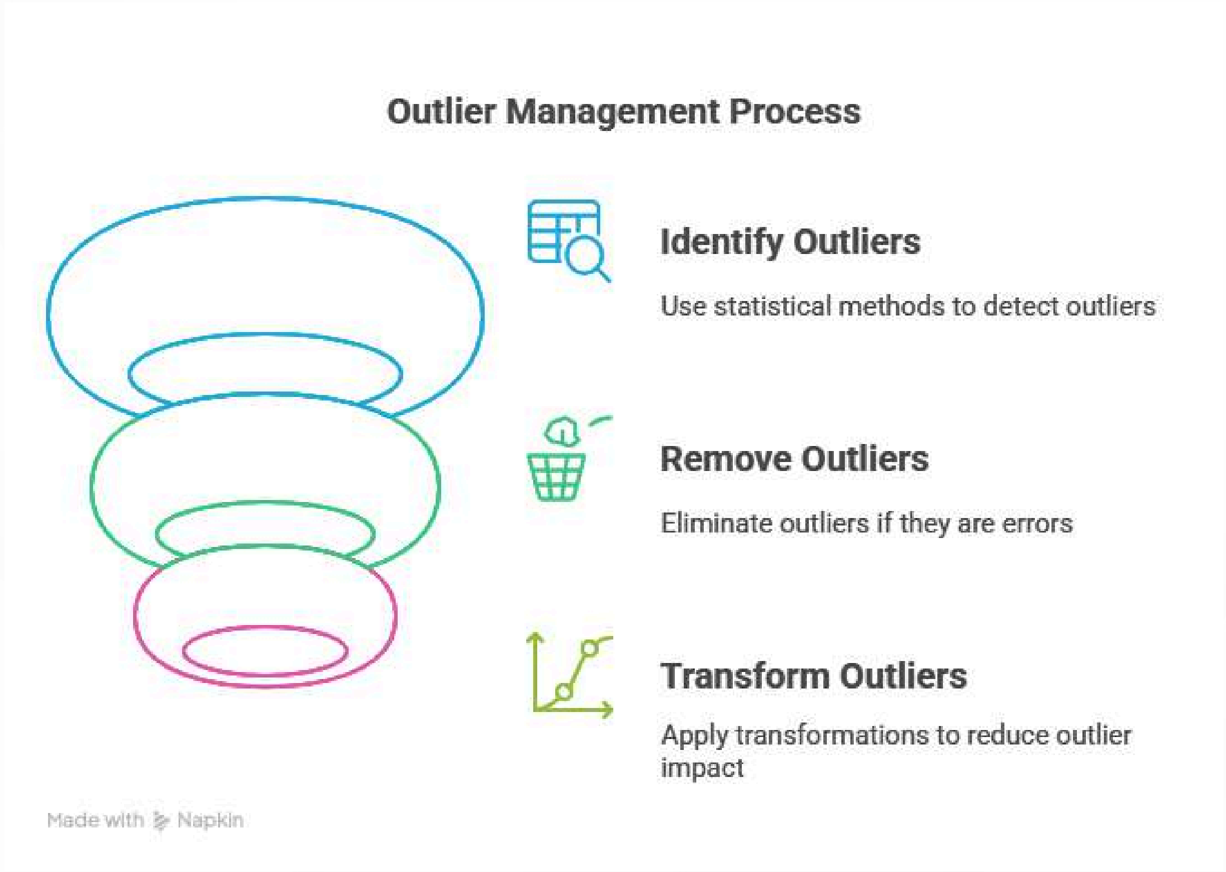
Co-funded by
the European Union

PREPROCESSING FOR SUPERVISED LEARNING IN R

Outliers play a significant role in this scenario. In supervised learning, properly preprocessing the data is essential for the model's effective learning. When handling outliers, preprocessing involves recognizing and addressing these extreme values that may skew the results.

Co-funded by
the European Union

PREPROCESSING FOR SUPERVISED LEARNING IN R


Outlier Management Process
Identify Outliers
Use statistical methods to detect outliers
Remove Outliers
Eliminate outliers if they are errors
Transform Outliers
Apply transformations to reduce outlier impact
Made with Napkin

Co-funded by
the European Union

PREPROCESSING FOR SUPERVISED LEARNING IN R

To perform preprocessing for supervised learning in R,it is possible to follow these steps:

Outliers identification: the use of statistical methods can be used to detect outliers in the considered dataset. Common techniques include calculating the Z-score or using the Interquartile Range (IQR).

~~~
```R
# Calculate Z-scores
z_scores <- scale(your_data)
outliers_z <- which(abs(z_scores) > threshold)
~~~

PREPROCESSING FOR SUPERVISED LEARNING

```
# Calculate IQR
Q1 <- quantile(your_data, 0.25)
Q3 <- quantile(your_data, 0.75)
IQR <- Q3 - Q1
outliers_iqr <- which(your_data < (Q1 - 1.5 * IQR) | your_data > (Q3 + 1.5 * IQR))
```
```

**Handling Outliers:** It is necessary to decide on a method to deal with the identified outliers. It is possible to remove them, transform them, or replace them with statistical measures like the median or mean.
```

PREPROCESSING FOR SUPERVISED LEARNING

- **Removing Outliers**:

  ```R
  cleaned_data <- your_data[-outliers_z] # For Z-score method
  ```

PREPROCESSING FOR SUPERVISED LEARNING

- **Replacing Outliers:**

```R
your_data[outliers_iqr] <- median(your_data, na.rm = TRUE) # Replacing with median
```

Transforming and Normalizing outliers: Applying transformations to the entire dataset to minimize the impact of outliers and improve the learning algorithm’s performance.

ANALYST

Co-funded by the European Union

PREPROCESSING FOR SUPERVISED LEARNING

```R
normalized_data <- scale(your_data) # Normalize data
```

Building a Robust Model: With the preprocessed data, it si possible to proceed to train the supervised learning model, ensuring that data is clean and ready for analysis, thereby reducing noise and focusing on meaningful patterns.

```R
model <- lm(target ~ ., data = normalized_data)
```

PREPROCESSING FOR SUPERVISED LEARNING

These steps will help ensure your model is less affected by outliers and more representative of the underlying data patterns.

Co-funded by the European Union

STEPWISE LOGISTIC REGRESSION IN R

Stepwise regression in R is a method used to select the most significant variables in a regression model by adding or removing predictors based on specific criteria. This approach can be applied to logistic regression to identify which variables are most predictive of the outcome.

Co-funded by the European Union

STEPWISE LOGISTIC REGRESSION IN R

Here's a step-by-step process of performing stepwise logistic regression in R:

1. **Preparing the Data:** Ensuring the data is clean and properly formatted. Defining the response variable as a binary outcome, which is necessary for logistic regression.

2. **Fit the Initial Model:** it is necessary to start with a full model that includes all potential predictor variables. It is possible to use the `glm()` function for logistic regression with the `family=binomial` argument.

Co-funded by
the European Union

STEPWISE LOGISTIC REGRESSION IN R

3. **Using the Stepwise Selection:** Using the `step()` function to perform stepwise selection. This function iteratively adds or removes predictors based on a selection criterion (usually AIC - Akaike Information Criterion).

```R
stepwise_model <- step(full_model, direction="both")
```

Co-funded by the European Union

STEPWISE LOGISTIC REGRESSION IN R

- **Direction Options:**
  - `"forward"`: Starts with no predictors and adds them one by one.
  - `"backward"`: Starts with all predictors and removes them one by one.
  - `"both"`: Combines both approaches, adding and removing predictors as needed.

STEPWISE LOGISTIC REGRESSION IN R

4. Evaluating the Model: After running the stepwise regression, it is possible to evaluate the model's performance. It is necessary to check the summary of the selected model to interpret the coefficients and assess which variables are significant.

```R
summary(stepwise_model)
```

STEPWISE LOGISTIC REGRESSION IN R

5. Validate the Model: It's important to validate the model using a separate dataset or cross-validation to ensure that the model performs well on unseen data.

By following these steps, you can effectively apply stepwise logistic regression in R to identify the most influential variables in your dataset.

CROSS-VALIDATION IN R

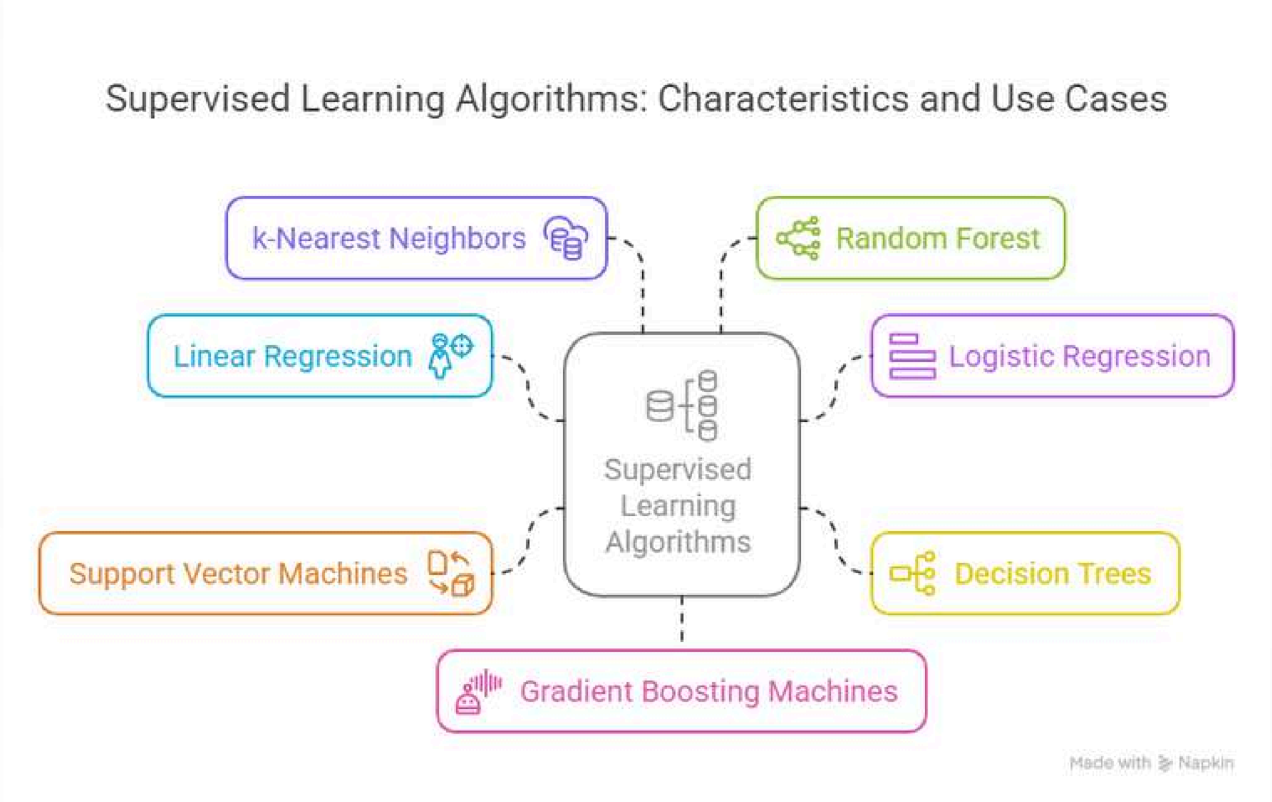
- **Cross-validation** is a method used in supervised learning to evaluate the performance and robustness of a model within the R programming environment. This technique involves dividing the dataset into multiple subsets, commonly known as "folds."
- In R, the guideline is again splitting the dataset typically involves **assigning 80% of the observations to the training set and 20% to the test set.** The observations are randomly distributed among the subsets to ensure an unbiased assessment of the model.

CROSS-VALIDATION IN R

- In R, the model is trained using the training set, while the remaining portion is designated as the testing set.
- The results provide an **estimation of the model's performance**. Cross-validation is crucial in R for identifying potential overfitting and ensuring that the model generalizes effectively to new, unseen data.

Co-funded by
the European Union

SUPERVISED LEARNING ALGORITHMS IN R



NEURAL NETWORKS

How to create a neural network in R

```

library(neuralnet)

```

4. **Define the Neural Network Model**

   - Specify the formula, data, hidden layers, and other parameters.

```R
nn <- neuralnet(target ~ predictors, data=train, hidden=c(5,3), linear.output=TRUE)
```

Co-funded by the European Union

NEURAL NETWORKS

5. **Evaluate the Model**

   - Predict and assess performance on the test set.

   ```R
   predictions <- compute(nn, test[,predictors])$net.result
   ```

6. **Visualize the Neural Network**

   ```R
   plot(nn)
   ```

Co-funded by
the European Union

HYPERPARAMETER TUNING IN R

To carry out **hyperparameter tuning in R**, it is necessary to follow these steps:

- **Defining Hyperparameters**: Determining which hyperparameters to optimize, such as learning rate or number of layers.
- **Choose a Search Strategy:** Selecting a method, for example:
  - Grid Search: Using the `expand.grid` function to test all possible combinations.
  - Random Search: Utilizing the `runif` or `sample` functions to randomly sample options.
  - Bayesian Optimization: to be Implemented with packages like `rBayesianOptimization` to focus on promising areas.
- **Set Up Cross-Validation:** Using packages like `caret` or `mlr` to implement cross-validation for accurate model performance assessment.

ANALYST
Co-funded by the European Union

HYPERPARAMETER TUNING IN R

- **Evaluate Performance:** It is necessary to train the model and evaluate its performance using metrics like accuracy with functions such as `caret::train`.
- **Select Best Hyperparameters:** It is necessary to identify the hyperparameters that produce the best results using functions like `caret::train`.
- **Validate the Model:** then testing on a separate dataset to ensure robust performance on unseen data using functions like `predict` and `confusionMatrix`.
- **Iterate if Necessary**: If results are not satisfactory, it is necessary to consider further exploration using different strategies or parameter ranges.

See Yang & Shami (2020)

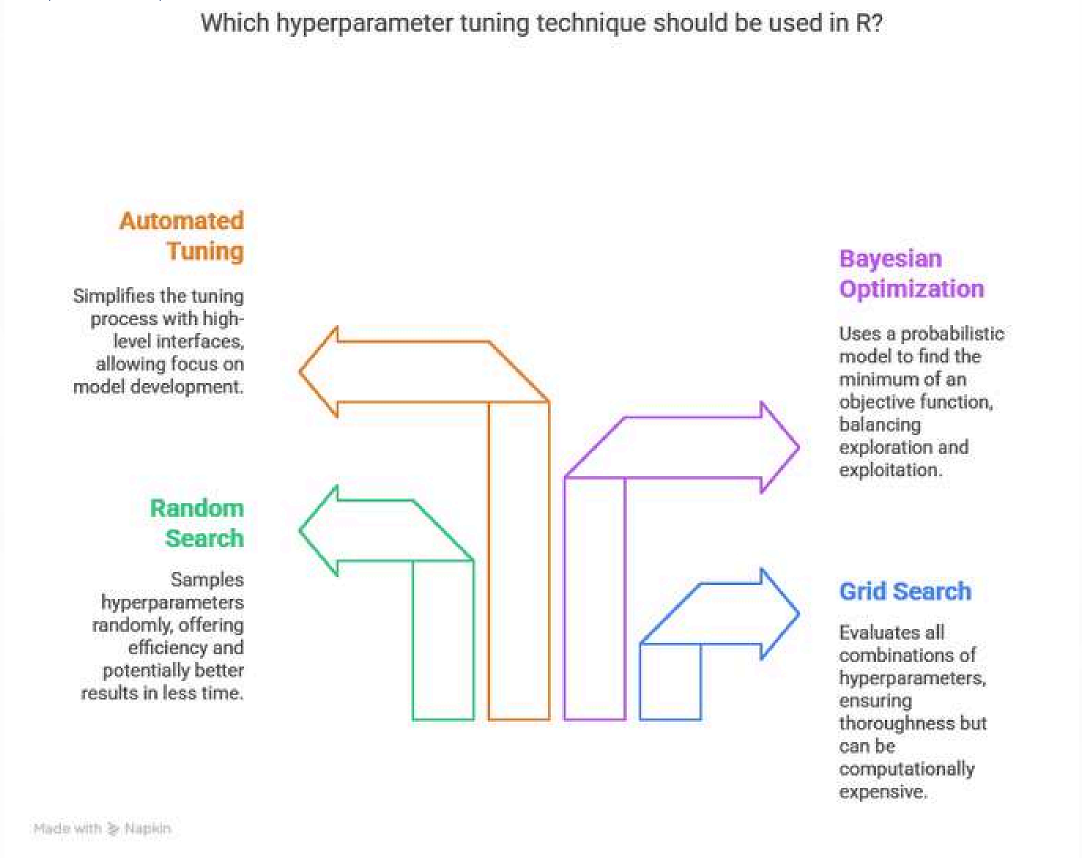
HOZINT HORIZON INTELLIGENCE fyg consultores UNIVERSITY OF WESTERN MACEDONIA Centro Studi AMIStaDeS APS UNIVERSITÀ CUSANO European Institute for Local Development UNIVERSIDAD DE ALMERÍA

ANALYST

Co-funded by the European Union

HYPERPARAMETER TUNING IN R

See Yang & Shami (2020)


Which hyperparameter tuning technique should be used in R?

Automated Tuning

Simplifies the tuning process with high-level interfaces, allowing focus on model development.

Random Search

Samples hyperparameters randomly, offering efficiency and potentially better results in less time.

Bayesian Optimization

Uses a probabilistic model to find the minimum of an objective function, balancing exploration and exploitation.

Grid Search

Evaluates all combinations of hyperparameters, ensuring thoroughness but can be computationally expensive.

Co-funded by the European Union

ENSEMBLE LEARNING IN R

Ensemble learning is a supervised method in R that combines multiple classification algorithms to improve prediction accuracy. Instead of using just one algorithm, ensemble learning runs **several algorithms simultaneously** and merges their predictions using a specific combination rule.

A common approach is the "Majority Rule," where the final prediction is determined by selecting the class most frequently predicted by all the individual algorithms. This method increases the reliability and accuracy of the results by integrating multiple algorithms. For further details, see Witten et al. (2013).

Co-funded by the European Union

REFERENCES

- Bishop, C. M. (2006), Pattern Recognition and Machine Learning, Springer, ISBN 978-0-387-31073-2
- Hassan, B. A., Tayfor, N. B., Hassan, A. A., Ahmed, A. M., Rashid, T. A., & Abdalla, N. N. (2024). From A-to-Z review of clustering validation indices. Neurocomputing, 601, 128198.
- Little, R. J., & Rubin, D. B. (2019). Statistical analysis with missing data. John Wiley & Sons.
- Tufte, E. R (1983). The visual display of quantitative information (Vol. 2, No. 9). Cheshire, CT: Graphics press.
- James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112, No. 1). New York: springer.

REFERENCES

- Yang, L., & Shami, A. (2020). On hyperparameter optimization of machine learning algorithms: Theory and practice. Neurocomputing, 415, 295-316.