A Comprehensive Guide to Feature Selection, Classification, and Evaluation Metrics

MATLAB-based MLS: Feature Selection, Classification Process, and Evaluation Metrics

MLS is a powerful tool for analyzing complex datasets, and its effectiveness largely depends on the careful selection and processing of features. By following a structured approach, you can enhance the performance of your models and achieve reliable results. This guide outlines a step-by-step process for feature selection, classification, and evaluation in MLS, with a focus on biomedical signal analysis.

Step-by-Step Process Based on Feature Selection and Classification

Input and Labels

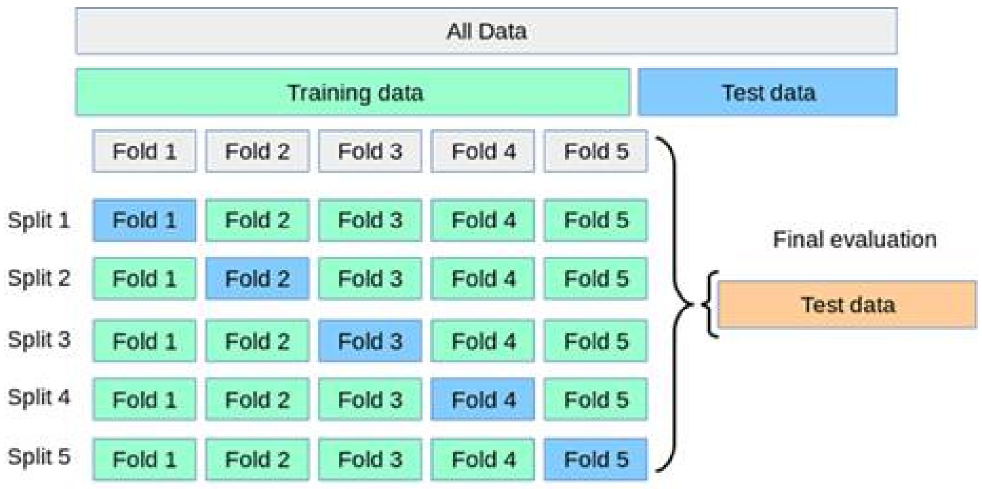
1. **Select Input**: Begin by loading your data. Choose the appropriate input type based on the preprocessing stage, such as:
   - Raw Signal
   - Filtered Signal
   - Wavelet Signal
   - Feature Signal
2. **Labels**: Assign labels to the data, which is crucial for supervised learning tasks. This involves selecting channels (e.g., Chi: and Chj:) to define the target classes for classification.

Cross Validation

1. **Select Validation Type**: Use k-fold cross-validation, which divides the dataset into k subsets (folds) for training and testing. This method assesses model performance robustly and helps prevent overfitting by evaluating the model on multiple data splits.
2. **Overfitting**: Occurs when the model learns training data noise too well, leading to poor generalization. Mitigate overfitting by:
   - Increasing the number of folds
   - Using regularization techniques
   - Reducing model complexity
3. **Underfitting**: Happens when the model is too simple to capture data patterns, resulting in low performance. Address underfitting by:
   - Increasing model complexity

- Extending the number of training epochs

4. **Role of Cross Validation**: Cross-validation helps identify overfitting (high variance) and underfitting (high bias) by partitioning the data into training and validation sets across multiple folds. This process provides a more reliable estimate of model performance and guides necessary adjustments.



Dimension Reduction

Choose a method to reduce dimensionality, which simplifies the dataset while retaining essential information:

1. **PCA (Principal Component Analysis)**: A linear dimension reduction technique that transforms the original features into a new set of uncorrelated principal components, ranked by variance. PCA is useful for reducing noise and computational load.
2. **FDA (Fisher Discriminant Analysis)**: Also known as Linear Discriminant Analysis in some contexts, FDA maximizes class separability by projecting data onto a lower-dimensional space. It is effective for tasks with distinct classes.

Feature Selection Algorithms

Feature selection is crucial for refining your dataset to include only the most relevant features:

1. **T-test**: A statistical test comparing means between classes, useful for identifying features with strong discriminative power.
2. **ANOVA**: Assesses variance between group means to determine feature significance across multiple classes (Useful for multi-class problems, it compares within-group variance to between-group variance).
3. **MI (Mutual Information)**: Mutual Information measures the dependency between features and the target variable. It is a non-linear method that selects features with the highest mutual information, making it suitable for complex datasets where linear methods like t-test or ANOVA may fail.
4. **FDR (False Discovery Rate)**: FDR controls the expected proportion of false positives among selected features. It is a statistical approach to adjust p-values in multiple hypothesis testing, ensuring that the number of incorrectly selected features is minimized, which is valuable in high-dimensional datasets.

Select Classification Type

Once features are optimized, choose a classification algorithm that suits your specific needs:

1. **MLP (Multi-Layer Perceptron):** A type of feedforward artificial neural network with multiple layers (input, hidden, output). It uses backpropagation and a learning function (e.g., trainlm) to adjust weights, making it suitable for complex, non-linear classification tasks like image or speech recognition**.**
2. **Bayesian:** This method calculates the probability of a class given the input features. It assumes feature independence and is effective for text classification or datasets with high dimensionality, though it may struggle with complex dependencies.
3. **SVM (Support Vector Machine):** A supervised learning method that finds the optimal hyperplane to separate classes in a high-dimensional space. It uses kernel tricks (e.g., linear, polynomial) to handle non-linear data, making it ideal for binary classification tasks with clear margins.
4. **KNN (K-Nearest Neighbors):** A non-parametric method that classifies data points based on the majority class of their k nearest neighbors in the feature space. It's simple and effective for small datasets, but can be computationally expensive for large ones.
5. **LDA (Linear Discriminant Analysis):** A dimensionality reduction and classification technique that maximizes class separability by projecting data onto a lower-dimensional space. It assumes normal distribution and equal covariance, suitable for linearly separable data with multiple classes.
6. **RBF (Radial Basis Function):** Often used as a kernel in SVM or as a network layer, RBF classifies data using radial basis functions that measure the distance from a data point to a center. It's effective for non-linear problems but requires careful tuning of centers and spreads.
7. **PNN (Probabilistic Neural Network):** A feedforward neural network based on Bayesian classification and probability density functions. It's fast for training and good for pattern recognition, but can be memory-intensive due to its structure.
8. **ELM (Extreme Learning Machine):** A single-hidden-layer feedforward network where hidden node parameters are randomly assigned and output weights are analytically determined. It's extremely fast for training and suitable for regression and classification tasks with large datasets.
9. **DT (Decision Tree):** A tree-like model that makes decisions by splitting data based on feature values, leading to class predictions at the leaves. It's intuitive and works well for categorical data, but can overfit if not pruned.

Training and Evaluation

Configure your chosen algorithm and its parameters, then run the classification model with the selected cross-validation settings. The Training Data Results Display and Test Data Results Display will show k-fold cross-validation metrics, allowing you to review and analyze the model's performance comprehensively.

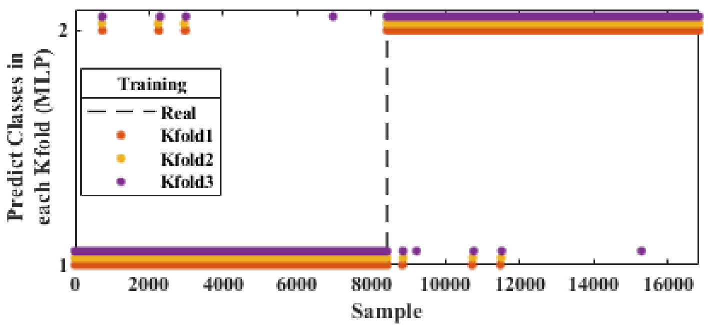
Evaluation Metrics

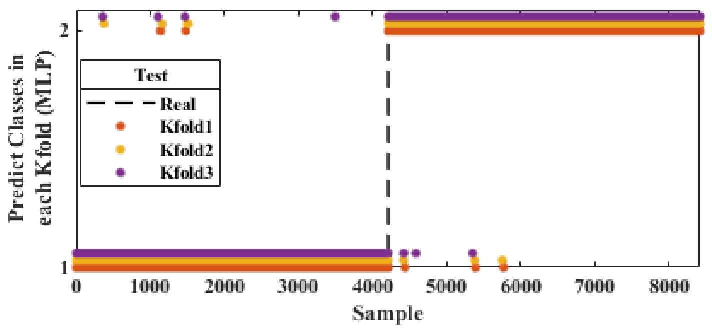
To accurately assess model performance, consider these crucial evaluation metrics:

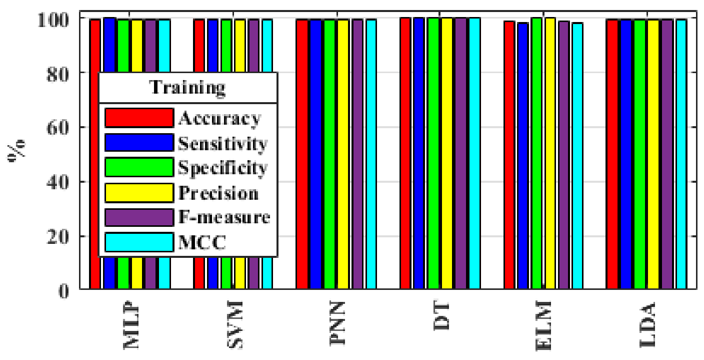
- **Accuracy**: Ratio of correctly predicted instances to total instances.
- **Sensitivity (Recall)**: Ratio of correctly predicted positives to actual positives.
- **Specificity**: · The ratio of correctly predicted negative instances to all actual negatives. It is important in scenarios where false positives need to be minimized.
- **Precision**: The ratio of correctly predicted positive instances to the total predicted positives. It is valuable when the cost of false positives is high.
- **F-Measure (F1-Score)**: The harmonic mean of precision and recall, providing a single metric that balances both. It is used when both false positives and false negatives are important, common in information retrieval.
- **MCC (Matthews Correlation Coefficient)**: A balanced measure that considers true and false positives and negatives, even in imbalanced datasets. It is particularly useful for binary classification tasks to assess overall performance.

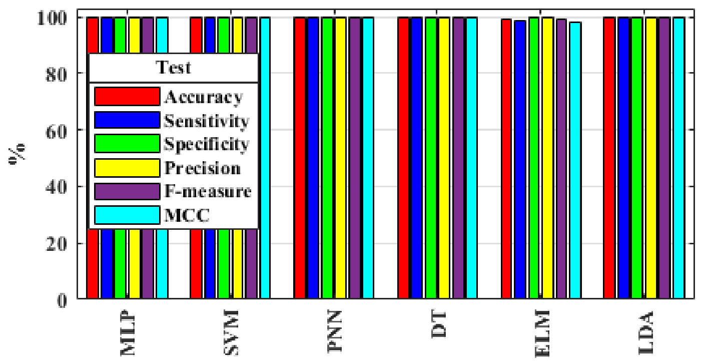
Metric Name	Formula
Accuracy	((TP + TN) / (TP + TN + FP + FN))
Sensitivity (Recall)	(TP / (TP + FN))
Specificity	(TN / (TN + FP))
Precision	(TP / (TP + FP))
F-Measure (F1-Score)	2 × (Precision × Recall) / (Precision + Recall))
Matthews Correlation Coefficient	(TP × TN - FP × FN) / √((TP + FP)(TP + FN)(TN + FP)(TN + FN))

- **TP**: Correctly predicted positives.
- **TN**: Correctly predicted negatives.
- **FP**: Negatives incorrectly predicted as positives.
- **FN**: Positives incorrectly predicted as negatives.









The figure analyzes the performance of machine learning models (MLP, SYM, PNN, DT, ELM, LDA) using 3-fold cross-validation. The top row shows predicted vs. actual class distributions for training and test data, revealing some misalignment, especially in test predictions. The bottom row presents performance metrics (Accuracy, Sensitivity, Specificity, Precision, F-Measure, MCC) on training and test sets, with LDA consistently outperforming others

By meticulously following these steps within the MLS, you can ensure high-quality, reliable machine learning outcomes for your biomedical signal analysis.

🔗 Explore MATLAB MLS repository to enhance your projects: https://lnkd.in/dDJUyNtc

Stay updated by following @rezasaadatyar