# MLEng Assignment 4: Advanced Parameter Search and Classifier Optimization

Highlight Reel

## 1 Objectives

By the end of this assignment, you will be able to:

- Apply parallel parameter search techniques to optimize your classification model
- Understand the trade-offs between different hyperparameter choices
- Visualize model performance across parameter spaces
- Fine-tune your best classifier from Assignment 3
- Create informative visualizations that demonstrate model improvements
- Apply distributed computing concepts using Ray for efficient parameter search

## 2 Background

In Assignment 3, you implemented an alternative classification algorithm that improved upon the baseline Random Forest classifier for time series data. Now, we'll take your model to the next level by applying systematic parameter optimization techniques.

Understanding how different hyperparameters affect model performance is crucial for developing effective machine learning solutions. In this assignment, you'll use distributed computing with Ray to efficiently search through parameter combinations and identify optimal configurations for your chosen classifier.

## 3 Understanding the Provided Parameter Search Code

The file `time_classification.py` has been updated to include a parameter search implementation using Ray for distributed computing. Let's understand how it works:

### 3.1 Ray for Distributed Computing

Ray is a framework that makes it simple to scale Python applications. In our parameter search:

- The `@ray.remote` decorator is used to define tasks that can be executed in parallel
- `ray.init()` initializes the Ray runtime
- `ray.wait()` is used to process results as they become available
- This allows us to evaluate multiple parameter combinations simultaneously

### 3.2 Parameter Grid Search

The code implements a grid search by:

- Defining a parameter grid with different window sizes and estimator counts
- Running each parameter combination as a separate task
- Collecting and visualizing results in a heatmap
- Identifying the best parameter combination

### 3.3 Visualization and Analysis

The results are visualized using:

- A heatmap showing performance across the parameter space
- Detailed metrics computed on the test set using the optimal parameters
- Output predictions saved to a CSV file for further analysis

## 4 Your Task: Optimize Your Best Classifier

Your task is to adapt the parameter search approach to optimize the best classifier you developed in Assignment 3. You will:

1. Integrate your classifier from Assignment 3 into the provided parameter search framework
2. Define appropriate hyperparameters to optimize for your specific algorithm
3. Implement the distributed parameter search
4. Analyze and visualize the results
5. Compare the optimized performance with your previous results

### 4.1 Required Steps

#### 4.1.1 Analyze the provided code

- Run the provided `time_classification.py` to understand how the parameter search works
- Identify the components you'll need to modify for your classifier

#### 4.1.2 Adapt the code for your classifier

- Replace the RandomForest with your best classifier from Assignment 3
- Modify the parameter grid to include relevant hyperparameters for your algorithm
- Update the `create_model` function to work with your classifier

#### 4.1.3 Expand the parameter search

- Include a hyperparameter specific to your chosen algorithm as well as window size
- Define appropriate ranges for each parameter
- Ensure the parameter space is meaningful but computationally feasible

#### 4.1.4 Analyze and document your findings

- Compare the optimized classifier with both your previous implementation and the baseline
- Document the impact of different parameters on model performance
- Explain any surprising or counterintuitive results

## 5 Submission Requirements

Submit your work as follows:

1. To your GitHub repository (shared with https://github.com/paulkefer):
    - Your modified `time_classification.py` file that includes:
        - Integration of your best classifier from Assignment 3
        - Implementation of the parameter search for your specific algorithm
        - Code for generating visualizations of the parameter space
    - If you implemented any additional files or utilities, include those as well
2. To our course Discord in the Assignment 4 thread:
    - All visualizations you've generated
    - A brief description (2-3 paragraphs) of your findings, including:
        - Which parameters had the most significant impact on performance
        - Any surprising or counterintuitive results
        - How your optimized classifier compares to your previous implementation and the baseline
    - A summary of the best parameter combination and the corresponding performance metrics

## 6 Tips for Success

- Start by running the provided code to understand the parameter search implementation
- Focus first on adapting the code for your classifier before expanding the parameter grid
- Choose parameter ranges wisely - too many combinations can be computationally expensive
- Create visualizations that highlight the most important parameter relationships
- Document your findings carefully, especially any interesting patterns or anomalies

## 7 Deadline

Submit all required materials by our next workshop.
Good luck!