Paper Title\* (use style: paper title)

\*Note: Sub-titles are not captured in Xplore and should not be used

line 1: 1st Given Name Surname   
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 4th Given Name Surname  
line 2: *dept. name of organization*  
*(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCIDline 1: 2nd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 5th Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCIDline 1: 3rd Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

line 1: 6th Given Name Surname  
line 2: *dept. name of organization   
(of Affiliation)*  
line 3: *name of organization   
(of Affiliation)*line 4: City, Country  
line 5: email address or ORCID

# Introduction

In this case we choose soft margin SVM to investigate the classification of time series in the big data context. It seeks to find a decision boundary, a linear model represented by a line in 2D or a hyperplane in higher dimensions, that maximizes the margin between the two classes, ensuring that all data points are correctly classified but at the same time allows for some points to be on the wrong side of the decision boundary, introducing a margin of tolerance for classification errors. SVM is also superior in handling of outliers and noisy data due to its focus on maximizing the margin and its dependence on support vectors. Outliers have less influence on the decision boundary as long as they are not among the support vectors. SVM has better performance in memory usage for it only needs to store the support vectors, making it memory-intensive, especially for large datasets.

Building upon this foundation, we integrate Spark into our big data preprocessing, using its parallelized processing capabilities to streamline the handling of massive time-series datasets, ensuring optimal performance on large-scale datasets.

In order to focus on the scalability and efficiency of the classification to accommodate larger-scale datasets, we introduce the Random Forest algorithm as an alternative to SVM. Random Forest can be easily parallelized, for each decision tree in the forest can be trained independently, and predictions can be made concurrently, leading to significant improvements in processing time. Random Forests can efficiently handle a large number of features without the need for feature selection or dimensionality reduction techniques. Moreover, they provide a measure of feature importance, allowing us to identify the most relevant features for classification. Moreover, Random Forest algorithms can be efficiently implemented in Apache Spark, as models can be trained and deployed on massive datasets with millions of samples under Spark clusters, making them suitable for large-scale classification tasks, while there isn't a native implementation of SVM in MLlib.

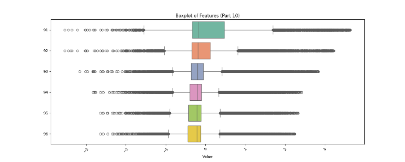
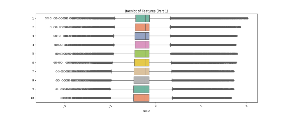
# Preprocession

## Maintaining the Integrity of the Specifications

Here we use utility function to extract numeric values from input data, handle any errors by correcting them to NaN, then forward fill any missing values with the most recent non-null value that occurred before it in the same column or row, aiming to facilitate data cleaning and standardization tasks within a data processing pipeline. then we record the characteristics of the cleaned data, including the number of time-series, their lengths, number of unique values, and any other relevant features.

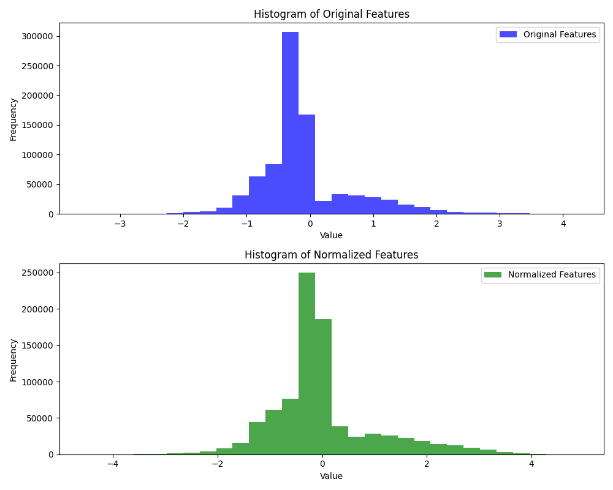
However, the cleaned data still have redundant data, which makes it necessary to find the starting line number of actual data in a text file and skip any comment lines. We use series of no-label function to assure that it starts from the identified data start line, followed by parsing the data, determining the number of columns, creating a DataFrame, and converting all feature columns to numeric types. And extract function are applied again to each cell, converting the entire DataFrame to numeric types, and filling missing values in both the training and test datasets with the mean of each column. Lastly, we merge the cleaned text and arff datasets, ensuring consistent column names and removing duplicated features from the arff dataset.

We divide the features into subsets to create multiple subplots, each containing boxplots as visualization of the distribution of a subset of features, which helps in identifying outliers and understanding the spread of data within each feature.( and other eight pics I sent on the GROUP)



We transform both the training and testing features using the computed mean and standard deviation, and replace the original feature values with the standardized values, known as mean feature values, in both the training and testing datasets.

We visualize the standardized features by plotting histograms of the original and standardized feature values, which helps in understanding how the distribution of feature values changes after standardization, making it easier to compare features and detect any shifts in data distribution.



# Integrate DTW with SVM

## Distance Calculation

Implement the Dynamic Time Warping (DTW) algorithm to compute the distance between pairs of time series arrays, combined with soft margin Support Vector Machine (SVM) for classification. First we initialize the parameters with provided default values, and choose radial basis function (RBF) kernel to initialize SVM model. Then we try to train the model using the input data (x) and corresponding labels (l). Then we calculate the DTW distance matrix between every pair of samples in the training data, which is then used as input features to train the SVM model with same method.

We take a set of testing data x as input, then use the trained SVM model to predict the class labels for the testing data based on the computed DTW distances.

## Prediction

We randomly select 50 samples from the training and test datasets, extract features and labels from the sampled training and testing datasets.

Fits the model using the sampled training data, for which predict labels with trained models

# Evaluation

We compute a classification report, which includes precision, recall, F1-score, and support for each class, which allows for assessing the performance of the classifier based on this subset of data. (Table for precision, recall, f1-score, support). Meanwhile the confusion matrix is also used to provide a summary of the predictions versus the actual labels.

| **Class** | **Evaluation Metrics** | | | |
| --- | --- | --- | --- | --- |
| ***Precision*** | ***Recall*** | ***f1-score*** | ***Support*** |
| 1 | 0.00 | 0.00 | 0.00 | 4 |
| 2 | 0.79 | 0.92 | 0.85 | 12 |
| 3 | 0.00 | 0.00 | 0.00 | 2 |
| 4 | 0.32 | 0.86 | 0.46 | 7 |
| 5 | 0.53 | 0.82 | 0.64 | 11 |
| 6 | 0.00 | 0.00 | 0.00 | 7 |
| 7 | 0.00 | 0.00 | 0.00 | 7 |
| ***Accuracy*** |  |  | 0.52 | 50 |
| ***macro avg*** | 0.23 | 0.37 | 0.28 | 50 |
| ***weighted avg*** | 0.35 | 0.52 | 0.41 | 50 |

# Inprovement with Random Forest

**IEEE conference templates contain guidance text for composing and formatting conference papers. Please ensure that all template text is removed from your conference paper prior to submission to the conference. Failure to remove template text from your paper may result in your paper not being published.**