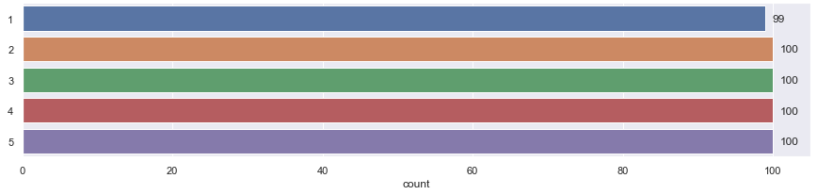
# Data Science Assignment - Final Report

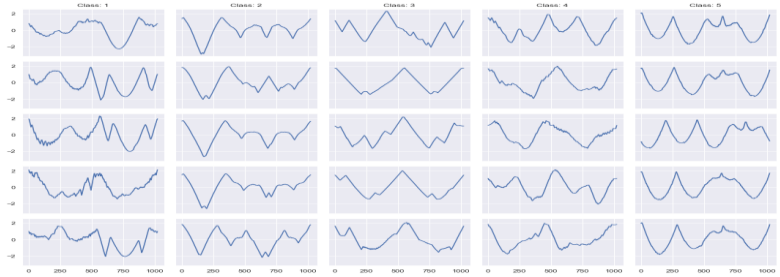
## Data Analysis

### EDA main findings:

1. The dataset contains 499 samples and 1025 columns.
2. Column names are float numbers with string type. They are not sorted but sorting them did not have any noticeable effect.
3. The target class column has the title "1" and it's the only one with dtype=="int64".
4. The dataset is balanced.




5. All sequences have the same length, so no need for special processing (e.g. padding).
6. Plotting a sample of the data revealed that each class has a distinguishable pattern.
   Some classes (e.g. class 1) might have multiple subgroups of similar patterns.



7. Looks like the similarity between samples of the same class is in the low-frequency part of the sequence (some samples has high-frequency information, which might be considered as noise). Therefore:
   a. Applying a low-pass filter could be useful.
   b. Dimensionality reduction can be applied without losing significant information.
8. No duplicates.
9. No missing values.

## Data Modeling

### Preprocessing:

1. Low Pass Filter (smoothing) – no performance gain was noticed after applying different window sizes.
2. Dimensionality reduction (down sampling) – some performance degradation was noticed after downsampled by a factor of 4, resulting in sequences with length of 256.
3. Label encoding – each class label was subtracted by 1 (from classes 1-5 to classes 0-4).

### Models:

1. Three experiments were conducted to test different approaches of time-series classification.
2. Each model hyperparameters were tuned using a grid search on a 5-fold cross-validation split.
3. The metric used to evaluate and compare the performance of the models was F1-score macro average.
4. Experiments:
   a. **Naïve KNN classifier** on the time series. The Dynamic Time Warping (DTW) metric was used to calculate the distance between two samples.
      Hyperparameters:
      - KNeighborsClassifier: n_neighbors, weights
      - DTW: window size
      - Distance metric: DTW, Euclidean (sklearn default)

b. **DWT Feature extraction and Random Forest Classifier.**
Feature extraction – features of each time-series sample were extracted using a Discrete Wavelet Transform (DWT) in order to reduce the time-series dimensionality and create a high-level representation of the sample.
Classifier – a Random Forest classifier was applied on the features dataset.
Hyperparameters:
- DWT: decomposition level
- RandomForestClassifier: n_estimators, max_depth

c. **LSTM + Fully Connected classifier**
A PyTorch implementation of an LSTM with FC layers and a softmax classifier.
Hyperparameters:
- LSTM: number of stacked LSTMs, hidden dimension
- FC layers: number of layers

## Results

| Model | Validation set Best Macro avg F1-score | Best parameters | Test set Macro avg F1-score | Test set inference time (seconds) |
|---|---|---|---|---|
| KNN | DWT=0.91<br>Euclidean =0.90 | n_neighbors=1<br>weights=uniform<br>metric=DWT.distance<br>window=60 | DWT=0.90<br>Euclidean=0.90 | DWT=489.57<br>Euclidean=1.64 |
| RandomForest | 0.90 | level=5<br>n_estimators=100<br>max_depth=16 | 0.88 | 0.9 |
| LSTM | **0.92** | LSTM num_layers=3<br>hidden_dim=256<br>FC num_layers=1 | **0.91** | **0.71** |

## Discussion

In many Machine Learning tasks, there is a tradeoff between performance and explainability. For systems in production, additional measures such as inference time, memory usage, hardware needs, code maintenance and debugging should be taken in to account.

As can be seen in the results – the best performing classifier is the Deep Learning LSTM model. However, the difference in performance between the winning model and the simple KNN is not big.

The simplicity of the KNN provides a model that is easy to understand and for production performance is preferable over the LSTM model. The simple Euclidean metric outperformed the supposedly better DWT distance. That could be a result of parameters tuning or actually a better metric for this task. Either way, Euclidean inference time is much faster.

As a conclusion, I would suggest compromising a bit in model accuracy and use the simple KNN model with Euclidean distance for production.

## Future Work

There is much more that can be done to improve performance.
Few things to consider in the future:

1. Feature engineering: look into what shape each sequence represent and try to detect meaningful features in the time-series representation (symmetry, curves, complexity of shapes…)
2. Augmentations: rotations and flips are easy to implement.
3. Reconstruct the shapes from the time-series representation and use computer vision methods to classify the shapes.