Brief Description

This document presents a brief theoretical overview of the project workflow found in `https://github.com/krl21/cls-as-webservice`.

The problem consists of having one or more models to classify integers according to a divisibility criterion. The classifications are:

- "Fizz" if the number is divisible by 3.
- "Buzz" if the number is divisible by 5.
- "FizzBuzz" if the number is divisible by 3 and 5.
- "None" otherwise.

To train the model and find the "best classifier", a workflow was defined. The models used, how the data was processed and the workflow are explained below.

Used models

For the experiments, 6 different classification models were used. These were:

- Logistic regression
- Support Vector Machine (SVM)
- Decision tree
- Random forest
- KNN
- Naive Bayes

Each of these supervised models was trained with integers from the interval 1000 to 2000. This interval was taken arbitrarily and because of the way the values are represented, any other interval of numbers can be taken.

The parameters taken from each model were those that come by default in the library used, since after verifying the metrics good results were obtained, so it was not necessary to adapt the model parameters to the problem.

Data processing

Training a model to classify individual natural numbers is not feasible or effective due to:

- Each integer represents a point in an infinite and discrete space, which means that the feature space is infinite dimensional. This makes it extremely difficult for a model to learn distinctive patterns between groups.
- A number, or a vector of size 1, does not provide enough contextual information for a model to classify it correctly, based on its divisibility with 3 and 5.

Using this representation would not exceed a value of 50 % accuracy, at best.

Therefore, the representation used belongs to a Boolean space of size 2: each position of the vector indicates whether the value is divisible by 3 and 5. For example, be some integers, their representation and classification:

Integer	Representation	Classification
0	[True, True]	"FizzBuzz"
1	[False, False]	"None"
3	[True, False]	"Fizz"
5	[False, True]	"Buzz"

Data Pipeline

The workflow is described below, but is essentially divided into 2 stages: the training and model selection stage, and the prediction or classification stage.

The training and model selection stage

While the server loads, the six models described at the beginning of the document are trained using the range of integers from 1000 to 2000. These numbers are represented in the Boolean space of dimension 2 and each model is trained using k-fold (doing mechanism: K random subgroups of the data set, the model trains with k-1 subgroups and evaluates with the remaining ones; this is repeated k times).

The system shows the process of this stage and shows the accuracy metric for each model. A possible system output for this stage is this:

```
### BUILDING MODELS ###
  Using K-fold. Model: logistic_regression. Average accuracy: 1.0
  Using K-fold. Model: svc. Average accuracy: 1.0
  Using K-fold. Model: decision_tree. Average accuracy: 1.0
  Using K-fold. Model: random_forest. Average accuracy: 1.0
  Using K-fold. Model: knn. Average accuracy: 1.0
  Using K-fold. Model: naive_bayes. Average accuracy: 1.0
```

Then, the models with the highest accuracy value are taken and trained with values from 1 to 100. These are the ones that will define the system classifier.

A possible system output for the second training stage is:

```
### TAKING THE BEST MODELS ###
  Using original test data. Model: logistic_regression. Accuracy: 1.0
  Using original test data. Model: svc. Accuracy: 1.0
  Using original test data. Model: decision_tree. Accuracy: 1.0
  Using original test data. Model: random_forest. Accuracy: 1.0
  Using original test data. Model: knn. Accuracy: 1.0
  Using original test data. Model: naive_bayes. Accuracy: 1.0
```

The prediction or classification stage

The API makes available the list of classifiers it manages. The client has the option to specify which one they want to classify with or leave it to the API's decision.

In the case of the second option, the system simulates a Random Forest model, that is, it classifies with all the "good" models and determines as the final classification the most common classification

determined by each model. This is the principle of the Random Forest model: having several decision tree models so that each one captures different characteristics and the final classification is an aggregation function with the result of all the models.