MACHINE LEARNING - 2021

Stud: Manuel-Alexandru Dragomir, group 246, ICA

Prof: Prof. Gabriela Czibula

Second software project

Notebook:

https://colab.research.google.com/drive/1kAlmRK_mmnWlSDFX5m-skWydQjdrBPkc?usp=sharing

(2) comments about the solution - problem analysis

- description and analysis of the used data:

Since we deal with a multiclassification problem, first of all we want to analyse what are the classes, view a histogram and some general information about the features (mean, maximum, minimum).

There are 11 classes that are abbreviated as following:

1. 'arc'(archaea)
2. 'bct'(bacteria)
3. 'phg'(bacteriophage)
4. 'plm' (plasmid)
5. 'pln' (plant)
6. 'inv' (invertebrate)
7. 'vrt' (vertebrate)
8. 'mam' (mammal)
9. 'rod' (rodent),
10. 'pri' (primate)
11. 'vrl'(virus).

The data is distributed as follows:

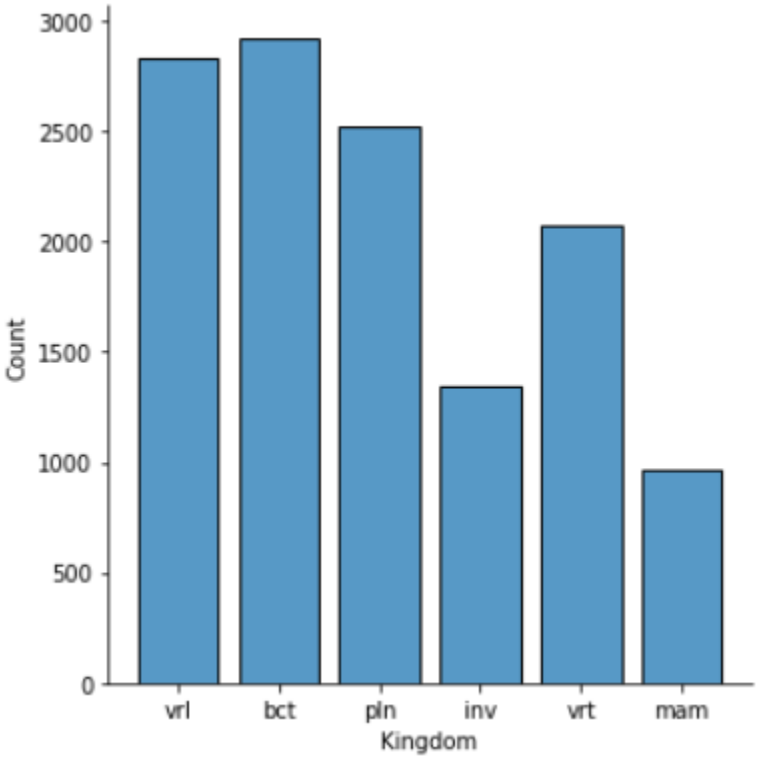
```
bct    2920
vrl    2832
pln    2523
vrt    2077
inv    1345
mam     572
phg     220
rod     215
pri     180
arc     126
plm      18
```

As it can be seen, we face a class imbalace issue. To solve this, based on the semantic of the problem we can merge some groups without altering the underlying structure. We concatenate pri (primates) + rod (rodents) + mam (mammals) into one big class mamals. Also, we would remove some irrelevant classes as arc (archaea), plm (plasmid) and phg (bateriophage). As a result, we finally have our final classes:

numeric data:

```
bct    2920
vrl    2832
pln    2523
vrt    2077
inv    1345
mam     967
```

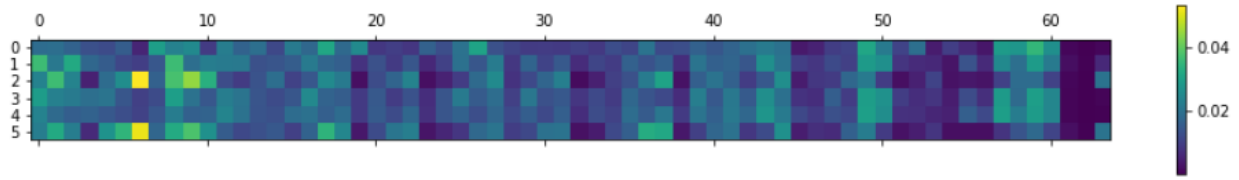
histogram:



view what are the most important features for each class:

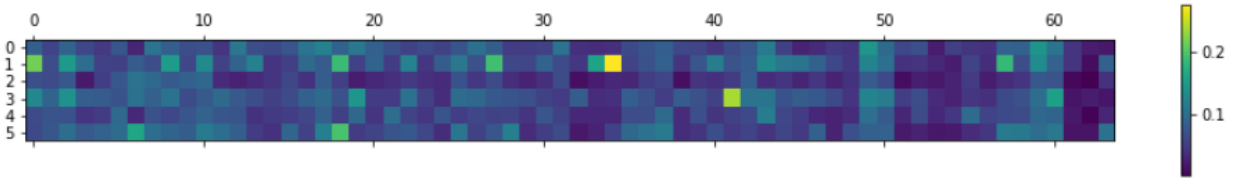
mean or Mean:

```
plt.matshow(df3.groupby('Kingdom').mean())
plt.colorbar()
plt.show()
```



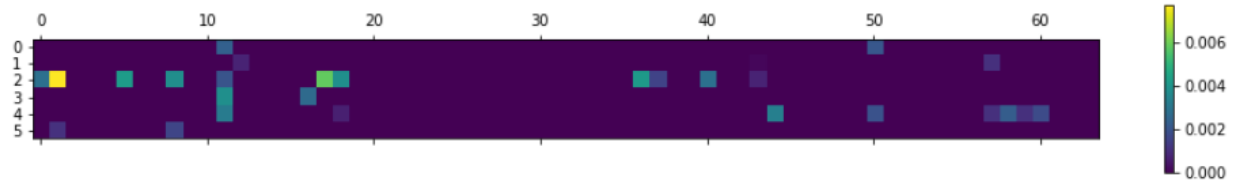
max or Maximum:

```
plt.matshow(df3.groupby('Kingdom').max())
plt.colorbar()
plt.show()
```



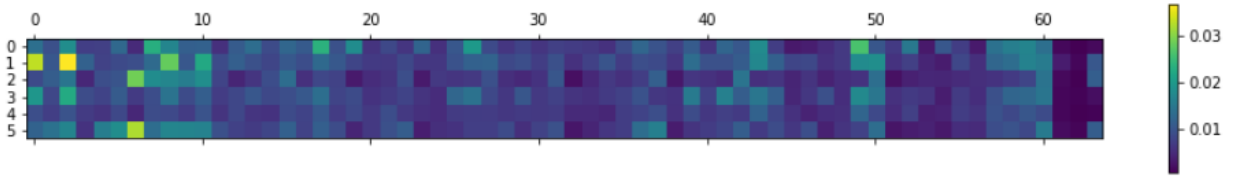
min or Minimum:

```
plt.matshow(df3.groupby('Kingdom').min())
plt.colorbar()
plt.show()
```



std or Standard Deviaton:

```
plt.matshow(df3.groupby('Kingdom').std())
plt.colorbar()
plt.show()
```

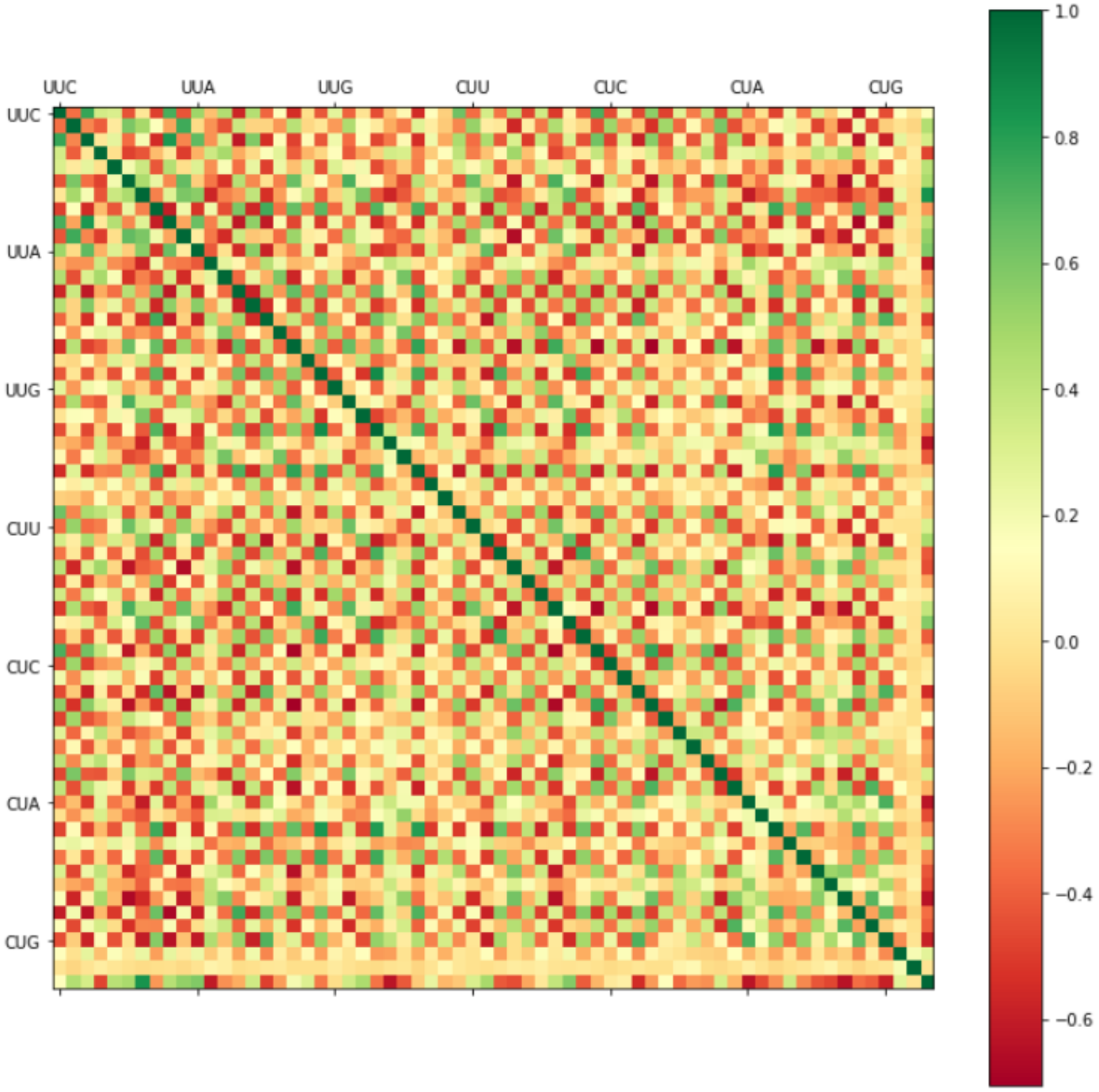


These are mainly used to identify what features described in the most insightful way some classes. For instance, looking at the min, we can see that feature 11 covers 4 out of 6 classes with nonzero values, which may hide some biological importance for that codon. Same for feature 50 that looks similar for class 0 and class 4.

Moreover, as we already mentioned, but it is important to reiterate, there are 64 features with values in interval [0, 1]

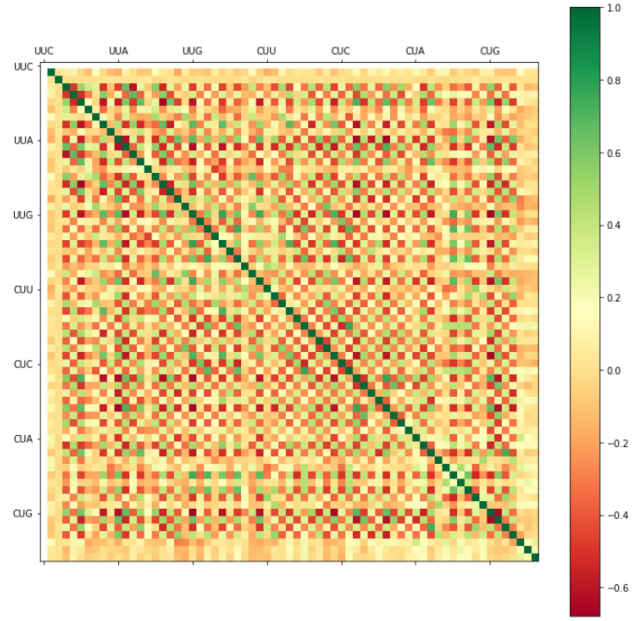
- description and analysis of the features used in learning as correlation, independence:

In this section, we will take a look if there is any kind of correlation (independence) between features for all the dataset and then for each class separately. This is the correlation matrix:

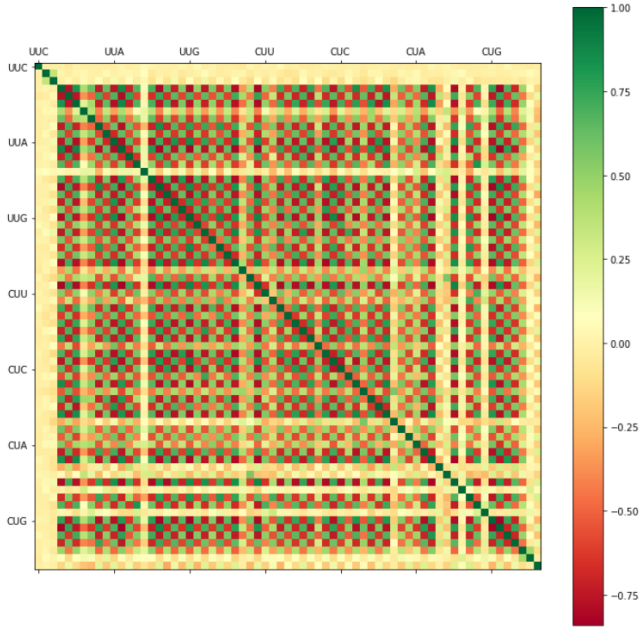


It mainly suggests as that most of data is not very correlated. But since there are 64 features it is very hard to assert that only by this visualisation. We will take a look on some classes independently – only on those that provide some visual results.

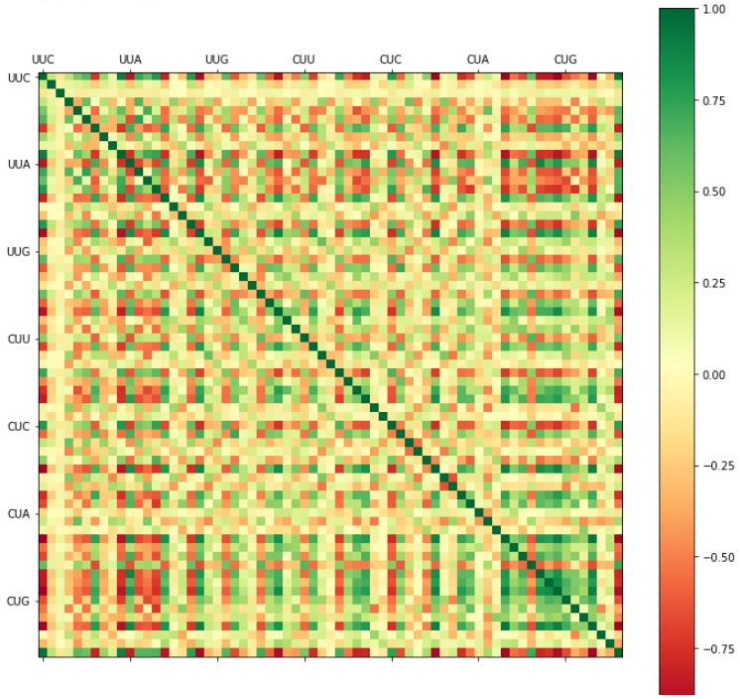
Virus – it is even highly uncorrelated but it does follow a pattern, margins are the most uncorrelated.



Bacteria – even two features are hiighly correlated, or they are independet (it mainly follows extreme values – except the margins)



Mammals: different from previous, the margins now are at extreme values not independent as for previous classes: they are mainly either green or red.



- description and analysis of the proposed solution: **Random Forest and Decision Trees**:

Decision Trees:

A decision tree is a tree that has as internal nodes rules based on features that separates data. While the internal nodes work as tests, the leaf nodes are the output classes. Usually, decision trees are binary trees, each internal node separates the data into two mutually exclusive subsets based on a certain rule – the separation is done to increase the *entropy* as much as possible at the current point. The entropy is a measure of uncertainty – the node should split the data in two subsets in such a way that the samples in each subset are highly correlated whereas the values from distinct subsets are very different. When a node divides data in multiple subsets then the tree is no longer a binary tree but K-tree, where K is the maximum number of subsets.

For the current task, the rules of the nodes will be based on the numerical features not on categorically features. Morever, for the current task we will try to use both entropy from information theory as a measure of uncertainty and also gini impurit.

The limitations of decision forest usually include overfitting and the fact that small perturbations in the data set or in the set of features would lead to significant changes in the form of the tree. To avoid having leafs with only one sample that is definitely a sign of overfitting, we will include some conditions. To avoid the second issue, we will use Random Forest.

Random Forest:

They are an ensemble of multiple invidual decision trees. Each decision tree predicts a class (also called a vote) and the final decision of the forest will the class with the most votes. The point is that the decision of multiple individuals is better than a single decision (wisdom of the crowds). As we presented before, a single decision tree could overfit or be mistaken, but multiple instances would change the result to the desired one as long as the trees are not very correlated. This is the core point – the trees should be low correlated. In order to do that, we use some randomisations processes.

One of them is called **Bagging** (=Bootstrap Aggregation). This mainly consists of getting a random sample size with replacement from the original data. For instance, if the data was X = [A, B, C, D], a possible result of the bagging could be Xb = [B, A, A, C]. So, there can be repetitions. This is mainly due to tackle one of the main issues of decision trees: small changes in the input dataset lead to significant differences regarding the tree.

Another random process is called **Feature Randomness** and it consists in choosing a random subset of feature for each tree to increase the variation of the result and the diversification of the forest. Typically, in a decision tree the best splitting decision is made at each node given the entire set of features which generally result in overfitting.

As a result, Random Forest's pieces are Decision Trees, Bagging and Feature Randomness.

In conclusion, we believe that Random Forest are a good choice for the current task – the classification of codons - because of their simplicity and their easy-understanding and visualisation of the final predictions. This could lead to some biological discoveries contrary to Neural Nets which mainly work as black-boxes when extracting the features and cannot be *debugged.*

(3) a design documentation

- the design of the learning task (target function to be learned, representation of the learned function, learning algorithm, learning hypothesis):

In the current section we will answer the questions firstly for the decision trees and then we will extend for the random forest ensemble.

The current target function has discrete output values.

- **target function to be learned:**
  - $f : codons \rightarrow class$, more exactly $f : [0, 1]^{64} \rightarrow \{0, 1, 2, 3, 4, 5\}$

As for the representation of the learned function, the internal nodes of the tree are the feature extractors whereas the leafs are the classes. Therefore we can formalize the learned function as $\theta = \{\theta_1, \theta_2, \dots, \theta_n\}$, where θ_1 is an internal node in the tree that covers a specific rule that written in the simplest form can be $\theta_1 = x_k \blacksquare v$, where x_k is the k-th feature, $\blacksquare$ is a comparison operator (as greater or equal than, smaller or equal than, or equals to) and v is a threshold real numeric value. Moreover, the order of the rules is important therefore in order to apply rule θ_p, all rules that appear on ascendant nodes should be applied before rule *p*. Regarding the learning, each rule is chosen in such a way that a measure of uncertainty (usually entropy or gini impurity) is minimised. In other words, the split done by the rule should obtain two subsets in such a way samples from same subsample are highly correlated whereas the values from distinct subsets are very different. We added the function for the entropy from (Introduction to Machine Learning by Nils J Nilsso) for a better understanding.

$$H(\Xi) = -\sum_i p(i|\Xi) \log_2 p(i|\Xi)$$

Given a rule, two subsets highly correlated will have a higher entropy, whereas two distinct subsets will have a lower entropy.

For random forests, the representation of the learned function consists of all the representations of its decision trees. Moreover, we apply the majority-rule to predict a class for a sample based on the votes of all the decision trees.

- the design of the application:

We plan to a minimally console interface where the user can select the hyperparameters, the training and the test path. There will be a simple menu where some commands can be send to the system – train model, test model, some visusalisations, save model, load model. We will use a OOP approach that can be viewed in the following diagram: