Introduction to Machine Learning - Phase 1
Master HPC-IA, Mines Paris

Ugo Pelissier
Dimitrios Tsitsos

December 7, 2022

Preliminary analysis of the dataset and analysis plan.

1. How many proteins are in this dataset?

To begin with, we load our data and we print the length of the dataset.

```
protein_labels = pd.read_csv("labels.csv",index_col=0)
print("There are", len(protein_labels), "proteins in this dataset.")
```

There are 1210590 proteins in this dataset.

2. What is the proportion of positive labels in the dataset? What is the proportion of negative labels? What metric(s) would be appropriate to validate the machine learning pipeline? Justify each of the metrics used.

```
def labels_proportion(protein_labels):
    n_true = (1*protein_labels["label"]).sum()
    p_true = n_true/len(protein_labels)
    p_false = 1-p_true
    plt.bar(["True","False"], [p_true,p_false])
    plt.show()
    print("p_true =",round(p_true,3))
```

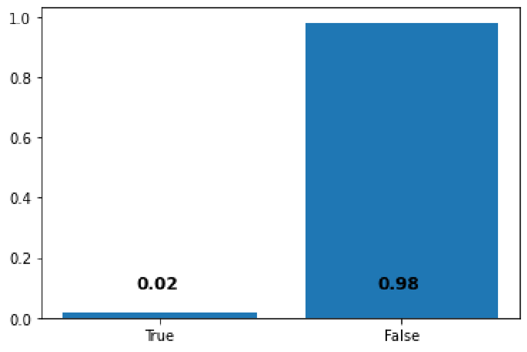

Figure 1: Proportion of true and false labels in the train data set

From the different metrics we have seen, we can first think of *accuracy* and *ROC_AUC*. However, these two metrics are better suited for balanced data set. In our case, we have to choose metrics that will take into account our uneven data set. From our research, both *imbalanced accuracy* and *F1-score* are good metrics when class metrics is uneven. To better understand the difference between these two metrics, we should write their formula:

$$\text{F1} = 2 * \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{Balanced Accuracy} = \frac{\text{specificity} + \text{recall}}{2}$$

From these formula, we can understand the following points:

- F1 score doesn't care about how many true negatives are being classified. When working on an imbalanced dataset that demands attention on the negatives, Balanced Accuracy does better than F1.
- In cases where positives are as important as negatives, balanced accuracy is a better metric for this than F1.
- F1 is a great scoring metric for imbalanced data when more attention is needed on the positives.

From this point, we believe that we need a similar attention to positive and negative labels so we prefer choosing balanced accuracy as our main metric, but we also plan on computing F1-score to have a comparison.

3. Now, looking at the complete list of labels, what is the number of elements in each category? What is the proportion of proteins labeled in two categories?

Looking at the *complete_labels.csv* file, we can notice that we have 134 columns, which represent subsets of the original secretion system families. We can then try to group our columns by families to know how many proteins belong to each big family.

```
def secretion_system_name(column_name):
    name = ""
    i = 0
    while(i<2):
        name+=column_name[i]
        i+=1
    return name
```

```
def secretion_system_list(complete_labels):
    column_name = complete_labels.columns
    secretion_system = np.array([])
    index = np.array([])
    i = 0
    for column in column_name:
        temp = secretion_system_name(column)
        if temp not in
             secretion_system.tolist():
            secretion_system =
                np.append(secretion_system,
                temp)
            index = np.append(index, i)
        i += 1
    index = np.append(index, i)
    index = index.astype(int)
    return secretion_system, index
```

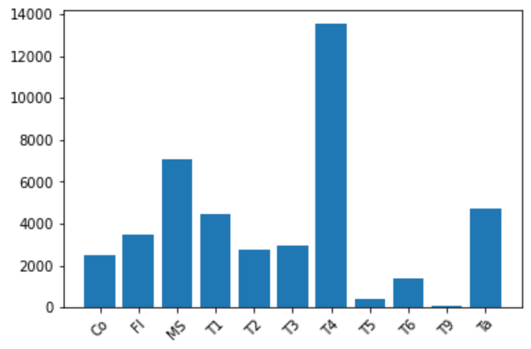

Figure 2: Number of elements in each secretion system family

We have 11 secretion system families, while 12 have been identified at least. Our data set is missing one or several secretion system families, which we need to keep in mind.

Then, we can compute the proportion of proteins labeled in two categories.

```
def double_proportion(complete_labels):
    complete_labels = complete_labels.values
    n = 0
    for i in range(len(complete_labels)):
        if ((complete_labels[i,:].sum())>1):
            n += 1
    print("p_double =",round(n/len(complete_labels),3))
```

Proportion of proteins labeled in two categories is 0.007

4. Identify and describe three challenges when working on this machine learning problem.

Looking at our datasets we can identify the following challenges:

- Imbalanced Dataset.
  As we can see in the second question the proportion of negative labels is way bigger than the positive one. This can lead to difficulties both on finding the best possible metric to validate the problem, and on generally processing the data. This represents a challenge for our cross-validation strategy that we will have the chance to explain in the following question.

- Large Number of Data.
  The number of data that we need to work on is really big which leads difficulties while processing them. The time for training a Machine Learning pipeline on this data set will be long, so we need to prepare carefully each attempt not to loose too much time with useless tries.

- Limited Number of Secretion Systems.
  In the *complete_labels.csv* dataset we can see that we don't have all the 12 possible Secretion Systems that we expect to identify. This is also an important challenge since we'll have to focus our efforts on the generalization ability of our Machine Learning algorithm, so that it is able to detect unknown secretion system families.

5. Devise (on paper) a cross-validation strategy that would enable to check that the machine learning pipeline would extend to secretion systems not annotated in our data set. You can write a pseudo-algorithm describing the cross validation strategy, or draw a schema explain your though. Explain how this cross validation strategy is a good way to check for generalization in this setting.

Our cross-validation strategy could be split in two parts.

The first one would be to create representative folders in regard of the proportion of positive and negative values in the initial data set. To be more specific, not only do we want to avoid having one folder with only false labels, but we also want that each folder have a proportion of true and false labels close to what we computed initially. That strategy is called **stratified k-folds cross validation**.

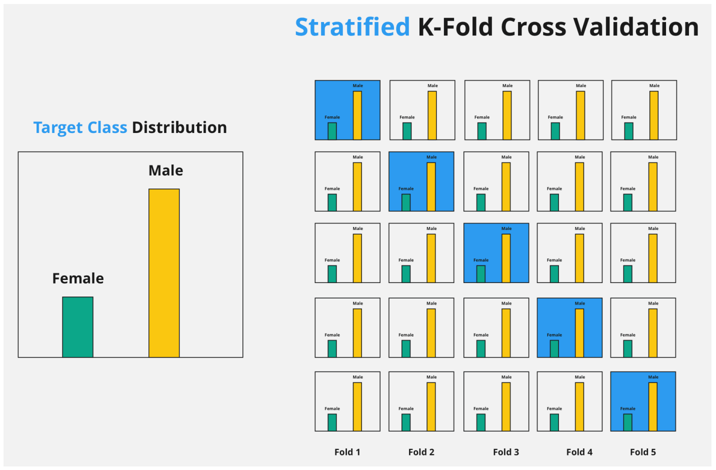

Figure 3: Stratified k-folds cross validation

In the same idea, we can stratify in order to have the same proportion of each secretion system family in each k-fold so that the pipeline does not over fit one family and keeps a good generalization ability.