COIMBRA UNIVERSITY

[BACHELOR'S DEGREE IN ENGINEERING AND DATA SCIENCE]

WATER QUALITY PREDICTION REPORT

MADE BY:

CATARINA PIMENTA SIMÕES

COMPUTER ENGINEERING DEPARTMENT

Index

Content

Dataset Description

[1] Accessibility to potable water is essential for health and a basic human right, playing a crucial role in health and development at national, regional, and local levels. In some regions, it has been demonstrated that investments in water supply and sanitation can generate economic benefits, as the reductions in adverse health effects and health costs outweigh the costs of implementing interventions.

The file water_potability.csv contains metrics related to water quality for 3276 different water samples:

pH Value: pH is an important parameter in assessing the water's acid-base balance and indicating its acidic or alkaline condition. The World Health Organization (WHO) recommends a maximum pH limit of 6.5 to 8.5. The current investigation ranges from 6.52 to 6.83, within the WHO standards.

Hardness: Hardness is mainly caused by calcium and magnesium salts, dissolved from geological deposits through which water travels. The duration of water's contact with the hardness-producing material helps determine the hardness in raw water. Hardness was originally defined as water's ability to precipitate soap caused by calcium and magnesium.

Total Dissolved Solids (TDS): Water can dissolve a wide range of minerals or inorganic salts and some organic substances, producing an undesirable taste and diluted appearance. High TDS values indicate high mineralization. The desirable limit for TDS is 500 mg/L, with a maximum limit of 1000 mg/L prescribed for drinking.

Chloramines: Chlorine and chloramine are the main disinfectants used in public water systems. Chloramines are formed when ammonia is added to chlorine to treat drinking water. Chlorine levels up to 4 milligrams per liter (mg/L or 4 parts per million (ppm)) are considered safe in drinking water.

Sulfate: Sulfates are natural substances found in minerals, soil, and rocks, present in the air, groundwater, plants, and food. The sulfate concentration in seawater is about 2,700 milligrams per liter (mg/L), varying from 3 to 30 mg/L in most freshwater supplies.

Conductivity: Pure water is a poor conductor of electric current but a good insulator. Increased ion concentration enhances water's electrical conductivity. According to WHO standards, electrical conductivity (EC) should not exceed 400 μS/cm.

Organic Carbon: Total Organic Carbon (TOC) in source waters comes from the decomposition of natural organic matter (NOM) and synthetic sources. TOC is a measure of the total amount of carbon in organic compounds in pure water. According to the US EPA, it should be < 2 mg/L as TOC in treated/drinking water and < 4 mg/L in water sources used for treatment.

Trihalomethanes: THMs are chemicals found in water treated with chlorine. THM concentration in drinking water varies based on the organic matter level, chlorine amount needed for water treatment, and water temperature. THM levels up to 80 ppm are considered safe in drinking water.

Turbidity: Water turbidity depends on the amount of solid matter present in suspension. It measures the light-emitting properties of water and is used to indicate the quality of waste discharge regarding colloidal matter. The average turbidity value obtained for Wondo Genet Campus (0.98 NTU) is below the WHO recommended value of 5.00 NTU.

Potability: Indicates whether the water is safe for human consumption, where 1 means Potable and 0 means Non-Potable.

Exploration of Data

Initially, after performing all the necessary library imports for this project, I visualized the existing data and calculated some statistical information:

Figure 1. Dataset Overview

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
0	nan	204.890455471...	20791.3189807...	7.30021187318...	368.516441349...	564.308654172...	10.3797830780...	86.9909704615...	2.96313538063...	0
1	3.71608007538...	129.422920514...	18630.0578579...	6.635245883862	nan	592.885359134...	15.1800131163...	56.3290762845...	4.50065627494...	0
2	8.09912418929...	224.236259393...	19909.5417322...	9.27588360269...	nan	418.606213064...	16.8686369295...	66.4200925117...	3.05593374966...	0
3	8.31676588421...	214.373394085...	22018.4174407...	8.05933237743...	356.886135643...	363.266516164...	18.4365244954...	100.341674365...	4.62877053683...	0
4	9.09222345629...	181.101509236...	17978.9863389...	6.54659997420...	310.135737524...	398.410813381...	11.5582794434...	31.9979927274...	4.07507542543...	0
5	5.58408663845...	188.313323769...	28748.6877390...	7.54486878877...	326.678362911...	280.467915933...	8.39973464015...	54.9178618419...	2.55970822755...	0
6	10.2238621645...	248.071735270...	28749.7165435...	7.51340846583...	393.663395515...	283.651633507...	13.7896953175...	84.6035561740...	2.67298873693...	0
7	8.63584871850...	203.361522584...	13672.0917639...	4.56300868559...	303.309771159...	474.607644942...	12.3638166987...	62.7983089629...	4.40142471544...	0
8	nan	118.988579090...	14285.5838542...	7.80417355307...	268.646940746...	389.375565871...	12.7060489686...	53.9288457675...	3.59501718095...	0
9	11.1802844707...	227.231469237...	25484.5084909...	9.07720001691...	404.041634684...	563.885481481...	17.9278064112...	71.9766010322...	4.37056193665...	0
10	7.36064010583...	165.520797259...	32452.6144091...	7.55070090670...	326.624353455...	425.383419495...	15.5868104380...	78.7400156643...	3.66229178285...	0

3276 rows × 10 columns

Figure 2. Statistical Data of the Dataset

	ph	Hardness	Solids	Chloramines	Sulfate	Conductivity	Organic_carbon	Trihalomethanes	Turbidity	Potability
count	2785.0	3276.0	3276.0	3276.0	2495.0	3276.0	3276.0	3114.0	3276.0	3276.0
mean	7.08079450427...	196.369496017...	22014.0925260...	7.12227679342...	333.775776610...	426.205110682...	14.2849702476...	66.3962929467...	3.96678616979...	0.3901098901
std	1.59431951870...	32.8797614762...	8768.57082778...	1.58308488903...	41.4168404616...	80.8240640511...	3.30816199912...	16.1750084222...	0.78038240848...	0.4878491696
min	0.0	47.432	320.942611274...	0.35200000000...	129.000000000...	181.483753985...	2.19999999999...	0.73799999999...	1.45	0.0
25%	6.09309191422...	176.850537877...	15666.6902969...	6.12742075549...	307.699497834...	365.734414118...	12.0658013336...	55.8445356209...	3.43971086961...	0.0
50%	7.03675210383...	196.967626863...	20927.8336065...	7.13029897388...	333.073545745...	421.884968280...	14.2183379372...	66.6224850980...	3.95502756299...	0.0
75%	8.06206612314...	216.667456214...	27332.7621274...	8.11488703210...	359.950170384...	481.792304487...	16.5576515438...	77.3374729087...	4.50031978728...	1.0
max	13.9999999999...	323.124	61227.1960077...	13.1270000000...	481.030642305...	753.342619558...	28.3000000000...	124.0	6.739	1.0

8 rows × 10 columns

In Figure 2, we can observe, for example, the quantity of data in each column, concluding that these are not equal in all, proving the existence of missing values. However, we can also observe the mean values, standard deviations, maximum values, etc. Figure 1 not only details the data but also highlights the columns where missing values exist, enabling us to address them later.

During this stage, I delved deeper into the distribution of water (potable/non-potable) in the dataset and in each column. For the latter, I used the kdeplot method from the seaborn library, which involves a kernel density estimate (KDE) plot. This method allows visualizing the distribution of observations in a dataset, similar to a histogram. The KDE represents the data using a continuous probability density curve in one or more dimensions [2]:

Figure 3. Water Distribution in the Dataset

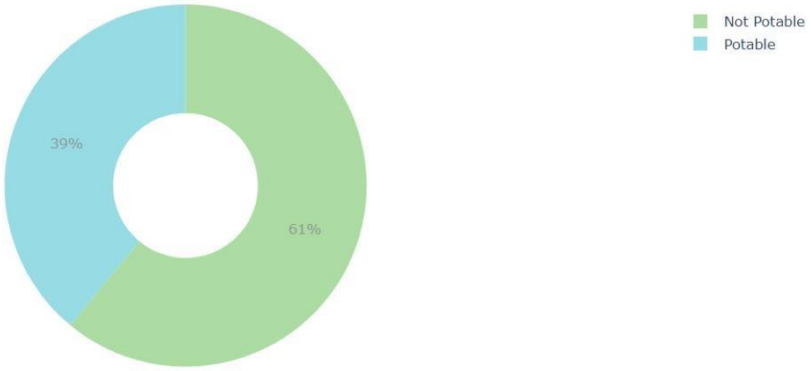
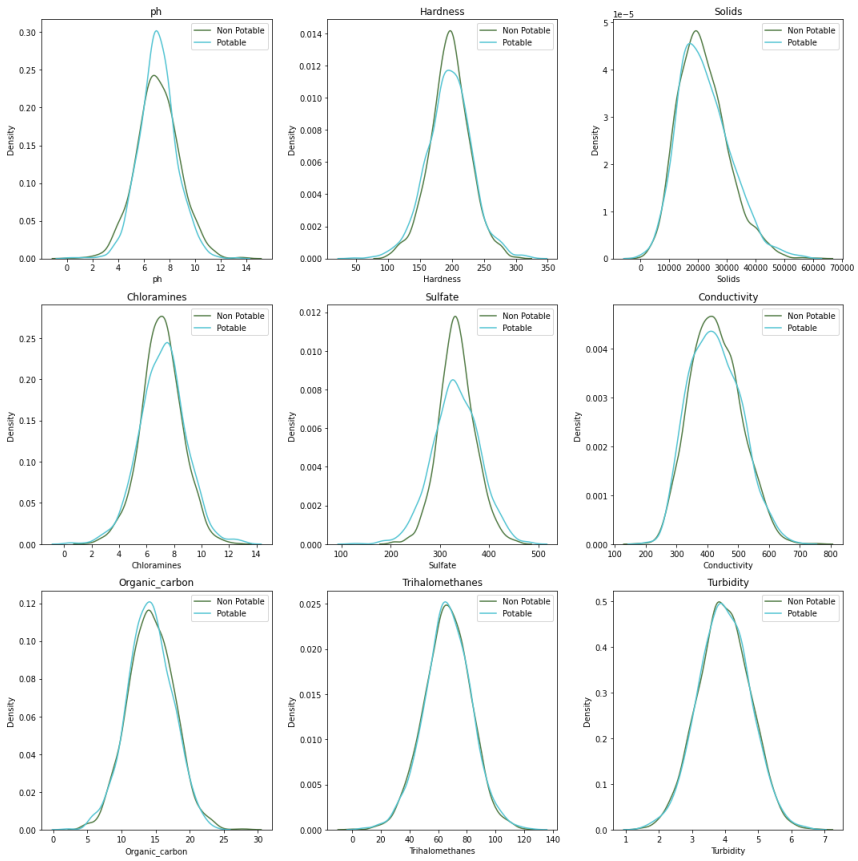


Figure 4. Water Distribution in Each Column



Consequently, it can be concluded that in the dataset, 39% of the water is potable, and 61% is non-potable. In Figure 4, we can observe that potable and non-potable water in the density estimate is visually distinguishable, where, for example, in pH, non-potable water reaches lower density values.

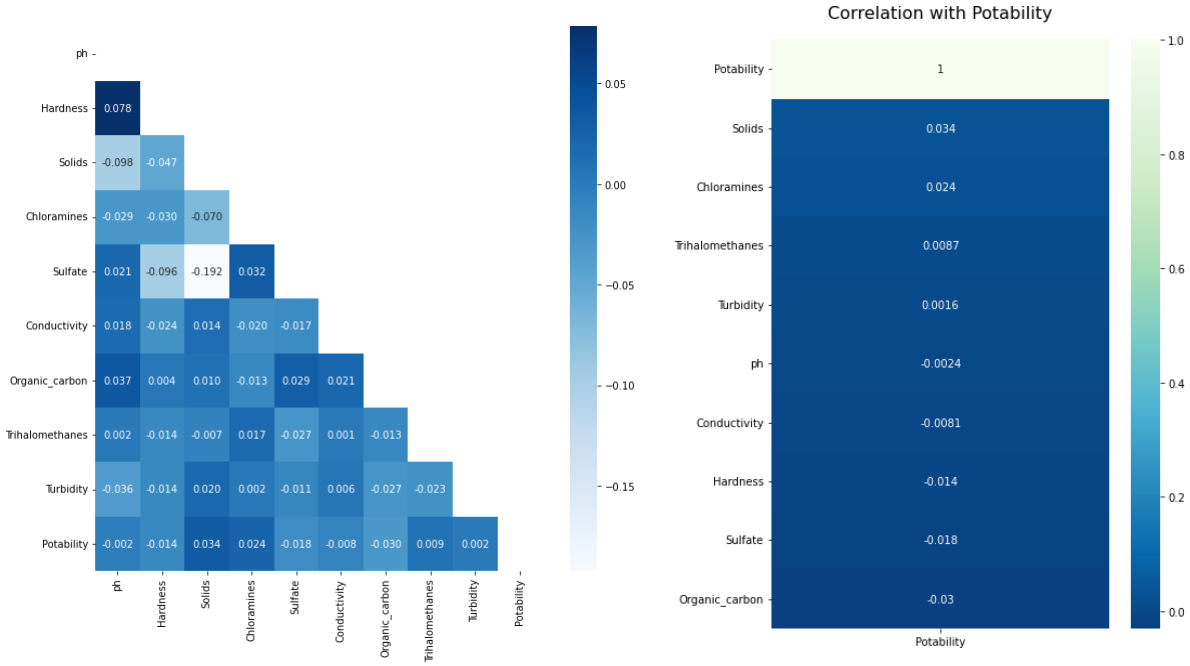
- **Data Correlation**

Correlation coefficients are statistical methods for measuring relationships between variables and what they represent.

Correlation aims to understand how one variable behaves in a scenario where another varies, identifying if there is any relationship between their variabilities. While it does not imply causality, the correlation coefficient expresses this relationship in numerical terms, quantifying the association between variables.

In this dataset, as observed in the figure below, there are no highly correlated data, with all coefficients remaining very close to 0. Therefore, I chose not to remove any variables:

Figure 5. Correlations



Preprocessing

- **Missing Values**

Previously, I discovered the presence of missing values in our dataset. Consequently, I meticulously analyzed the columns where these missing values are located and the quantity of null values within them:

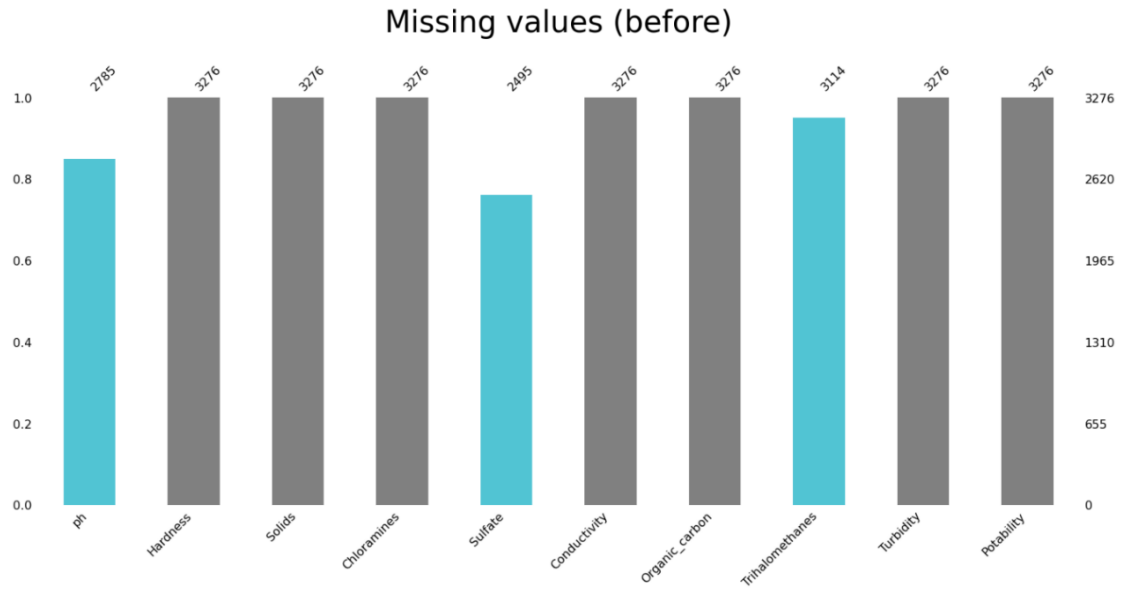
Figure 6. Specification of Missing Values

ph	491
Hard...	0
Solids	0
Chlor...	0
Sulfate	781
Cond...	0
Orga...	0
Trihal...	162
Turbi...	0
Potab...	0

10 rows × 1 columns

Next, I illustrated the observed, with columns containing missing values represented in blue and those without in gray. The number displayed above each bar corresponds to the quantity of data in each variable:

Figure 7. Illustration of Missing Data



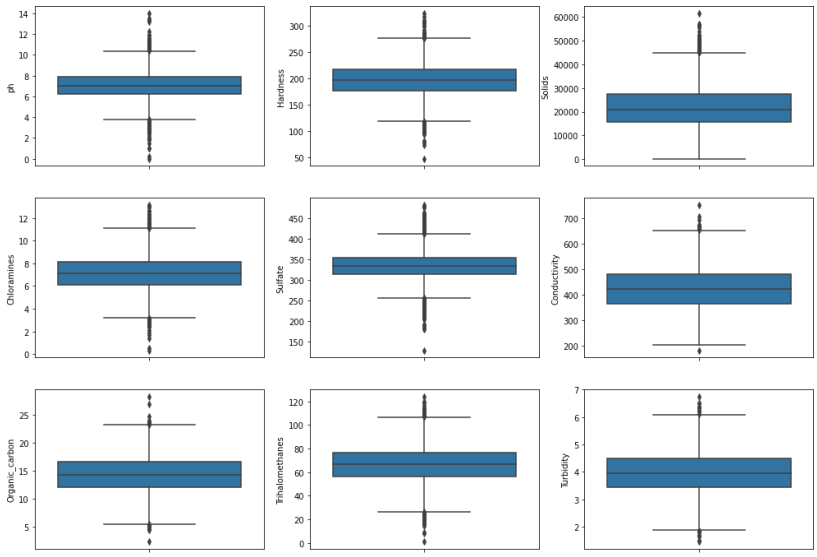
To handle these missing values, I used the *KNNImputer* method from the *scikit-learn* library. This method identifies neighboring points through a distance measure, and missing values can be estimated using complete values from nearby observations. The idea behind this method is to identify 'k' samples in the dataset that are similar or close in space. Then, we use these 'k' samples to estimate the value of missing data points. The missing values for each sample are imputed through the mean value of the 'k' neighbors found in the dataset [3].
In this project, I chose to use k=10, replacing the *KNNImputer* result in the dataset, as illustrated:

Figure 8. Missing Data after KNNImputer

- **Outliers**

As seen in Figure 9, there are outliers. However, I chose not to remove them since we are studying both potable and non-potable water, and, for example, it is plausible to have non-potable water with pH values higher than 12 or lower than 4.

Figure 9. Outliers



Models

In this stage, I began by splitting our data into X_train, X_test, y_train, and y_test, reserving 30% of the data for testing (X_test and y_test). Next, I presented the range of each variable to understand whether data scaling would be necessary.

From Figure 10, I concluded that the range for different variables is significantly distinct, such as the range for solids compared to pH. Consequently, I chose to scale X_train and X_test using the Standard Scaler, as it standardizes columns by removing the mean and scaling the variance to one unit, making them more manageable for our models [4]. After this, I proceeded to model implementation.

Figure 10. Range of Variables

	Name	Range
0	ph	0 to 14
1	Hardness	47 to 324
2	Solids	320 to 61228
3	Chloramines	0 to 14
4	Sulfate	129 to 482
5	Conductivity	181 to 754
6	Organic_carbon	2 to 29
7	Trihalomethanes	0 to 124
8	Turbidity	1 to 7

- **SVC**

Support Vector Machines (*SVMs*) are a set of supervised learning methods used for classification, regression, and outlier detection. Some advantages of *SVMs* include effectiveness in high-dimensional spaces and cases where the number of dimensions exceeds the number of samples. It uses a subset of training points in the decision function (called support vectors), making it memory-efficient, and is versatile as different kernel functions can be specified for the decision function. Common *kernels* are provided, but custom *kernels* can also be specified [5].

For this model, I chose to use the *GridSearchCV* method, which involves an exhaustive search over specified parameter values for an estimator. I specified parameters C and Kernel, resulting in an accuracy of 0.674.

Finally, I opted for cross-validation, a resampling method using different data subsets to test and train a model in different iterations, utilizing the best parameters from GridSearchCV, resulting in an accuracy of 0.61.

Figure 11. SVC Results with GridSearchCV and Cross-validation

```
{'C': 1.0, 'kernel': 'rbf'}
Test accuracy: 0.674
              precision    recall  f1-score   support

         0.0       0.67      0.93      0.78       603
         1.0       0.71      0.27      0.39       380

    accuracy                           0.67       983
   macro avg       0.69      0.60      0.58       983
weighted avg       0.68      0.67      0.63       983
```

```
[0.6097561  0.61068702 0.61068702 0.60916031 0.60916031]
0.61 accuracy with a standard deviation of 0.001
```

- **MLP Classifier**

MLPClassifier stands for *Multi-layer Perceptron Classifier*. Unlike other classification algorithms, such as Support Vectors or *Naive Bayes Classifier*, *MLPClassifier* relies on an underlying Neural Network to perform the classification task [6].

For this model, I again used the GridSearchCV method similar to the previous model. However, I specified parameters such as *Hidden_Layer_Sizes, Activation, Alpha, and Learning_Rate*, resulting in an accuracy of 0.677.

Finally, I once again performed cross-validation with the best parameters from GridSearchCV, resulting in an accuracy of 0.52.

Figure 12. MLP Results with GridSearchCV and Cross-validation

```
{'activation': 'relu', 'alpha': 0.05, 'hidden_layer_sizes':
Test accuracy: 0.677
              precision    recall  f1-score   support

         0.0       0.70      0.83      0.76       603
         1.0       0.62      0.43      0.51       380

    accuracy                           0.68       983
   macro avg       0.66      0.63      0.63       983
weighted avg       0.67      0.68      0.66       983
```

```
[0.43445122 0.60763359 0.39541985 0.58015267 0.56946565]
0.52 accuracy with a standard deviation of 0.09
```

- **Random Forest Classifier**

Finally, we opted to use the Random Forest Classifier, a generalization of the Decision Tree operation, where a set of (random) decision trees is used to minimize overfitting of each individual tree model generated for the input data [7]. I chose this model because it provides higher accuracy through cross-validation and handles missing values while maintaining accuracy for a significant portion of the data.

Similar to the previous models, I used GridSearchCV and cross-validation. In the first, I specified parameters *Max_Features and Max_Depth.* Ultimately, I achieved an accuracy of 0.678 and 0.64, respectively.

Figure 13. Random Forest Results with GridSearchCV and Cross-validation

```
{'max_depth': 10, 'max_features': 6}
Test accuracy: 0.678
              precision    recall  f1-score   support

         0.0       0.67      0.92      0.78       603
         1.0       0.70      0.29      0.41       380

    accuracy                           0.68       983
   macro avg       0.69      0.61      0.59       983
weighted avg       0.68      0.68      0.64       983
```

```
[0.6097561  0.64885496 0.65648855 0.61526718 0.66564885]
0.64 accuracy with a standard deviation of 0.02
```

Results and Discussion

There are several projects conducted with the dataset studied above, yielding various results. For example, in the study *"Water Potability Analysis"* [8], multiple models were explored, with the best being SVC, achieving an accuracy of 69% with GridSearchCV. Another study, *"Water Potability Prediction (Best Accuracy 69.5%)"* [9], reached an accuracy of 69.5% with the Random Forest Classifier.

In this project, the highest accuracy was obtained with the Random Forest model, achieving an accuracy of 0.678. Thus, it can be concluded that within existing works, the result falls within the expected range.

References

[1] Water Quality Dataset de https://www.kaggle.com/datasets/adityakadiwal/water-potability?datasetId=1292407&searchQuery=mlp

[2] Seaborn.kdeplot de https://seaborn.pydata.org/generated/seaborn.kdeplot.html

[3] KNNImputer: A robust way to impute missing values (using Scikit-Learn) de https://www.analyticsvidhya.com/blog/2020/07/knnimputer-a-robust-way-to-impute-missing-values-using-scikit-learn/

[4] Técnicas Standardscaler, Minmaxscaler E Robustscaler – ML de https://acervolima.com/tecnicas-standardscaler-minmaxscaler-e-robustscaler-ml/

[5] Support Vector Machines de https://scikit-learn.org/stable/modules/svm.html

[6] A Beginner's Guide To Scikit-Learn's MLPClassifier de https://analyticsindiamag.com/a-beginners-guide-to-scikit-learns-mlpclassifier/

[7] Random Forest de https://docs.lemonade.org.br/pt-br/spark/aprendizado-de-maquina/classificacao-random-forest.html

[8] Water Potability Analysis de https://www.kaggle.com/code/neesha12/water-potability-analysis

[9] Water Potability Prediction (Best Accuracy 69.5%) de https://www.kaggle.com/code/sinansaglam/water-potability-prediction-best-accuracy-69-5/notebook