Artificial intelligence

Book of exercises made by
Paulo Gouveia
Translated by Paulo Costa

1. For a first contact with Python language and VS Code editor, create a small program that starts by asking the user his name and age, and then tells him if he is adult or underage, treating him by his name.

2. Consider a set S consisting of n1 elements of a certain datatype and n2 elements of another datatype. Entropy is a measure that quantifies, in some way, the disorder (impurity) of a set of elements and, in case there are only 2 classes of elements, it's expressed as follows:

$$\text{Entropy}(S) = -p_1 \log_2(p_1) - p_2 \log_2(p_2),$$

   where p1=n1/(n1+n2) and p2=n2/(n1+n2) are the proportions of the first and second types of elements, respectively.
   Write in Python the entropy function, which begins by properly validating the input data of the problem, then calling it for sets with the following proportions: 0/1, 0.5/0.5, 0.1/0.9 e 0.9/0.1. Don't forget to properly comment the function.

3. Write a function in Python, identical of the previous problem, but which allows to calculate the entropy of a set containing elements of n distinct classes, which can be expressed mathematically as follows:

$$\text{Entropy}(S) = -\sum_{i=1}^{n} p_i \log_2(p_i),$$

   Being pi=ni/n the proportion of elements of the i-th class that are contained in the group.

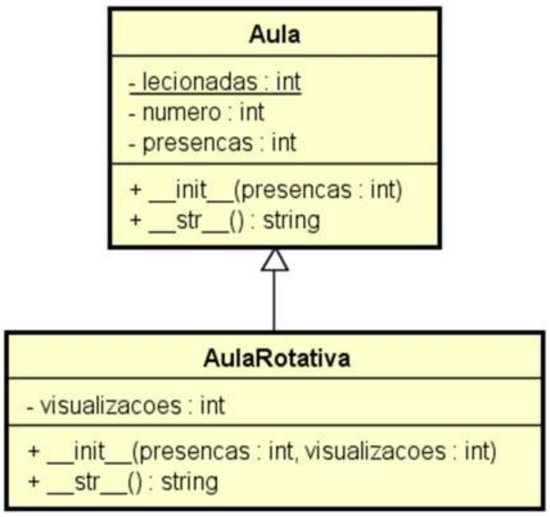
4. Read a text and present in alphabetical order the number of occurrences of each word contained in that text.

5. Define a function that checks whether given text is palindrome, ignore the case-sensitive of the text, and the punctuation, spaces, and other non-alphanumeric characters. For example, the text “Rise to vote, sir.” It's palindrome.

6. Define a class that represents a very simple collection of elements of any type, which supports, in addition to elementary insert and remove operations, querying the number of collected elements, through the **len()** function, as well as the possibility of sorting its elements, whenever a heterogeneous set is not concerned. In addition to allowing the insertion, of individual elements, the collection must be able to be created already with some elements. In order to simplify the solution, the class must store the elements in a Python list. Make the class itetable and test it properly.

7. Save instances of a class created in the previous exercise to the collection created and try to sort it. If this operation fails, refine the collection definition by properly handling this error. Later, make the instances of the class sortable.

8. Consider that you want to record the number of student attendances in classes of a certain course taken during the pandemic context that we faced in previous years. During that period, there was a need to implement a rotative system for the student attendance, with classes in two different typologies: one in normal model, where all students attended in person, and in a new typology, called rotation, in which some of the students attended in person and the rest attended online.



   a) Respecting the solution showing in the class diagram, build the classes for the problem, ensuring the automatic sequential numbering of the classes and in the case of rotative classes, both the number of physical attendances and the number of online students are recorded (Number of students attending class remotely).
   b) Add to the class(es) the possibility to reset the class numbering.
   c) Using “Lists by Understanding”, create, in a single line of code, a collection with the first classes, all face-to-face and with a decrease of 3 attendances per class, starting the 1st of them with 50 attendances and ending the last one with 20.
   d) Rebuild the previous collection, but now using generative expressions instead of understanding lists.
   e) Add three rotating classes to the previous collection and take advantage of polymorphism.

9. Build the NumPy array showing at right, without explicitly writing its values.

```
[[ 1  4  7 10 13 16 19 22 25 28]
 [ 2  5  8 11 14 17 20 23 26 29]
 [ 3  6  9 12 15 18 21 24 27 30]]
```

10. Suppose you want to assess the body mass index (BMI) of a group of people from their height and weight records. Knowing that each person's BMI is given by the ratio between their weight and the square of their height, implement all the following tasks using the features of the *NumPy package*.

   a) Build 2 arrays. The first one-dimensional, containing the names of a group of 10 people, who will choose randomly, and the other two-dimensional 2x10, with the weights and heights of these people, stored in the respective lines. In order not to enter all these values one by one, generate them randomly respecting the following constraints: all integer values; weights should be between 50 and 100 kg; and heights between 150 and 200 cm.
   b) Calculate, and save in a new array, the body mass index of each person.
   c) Show how people's weights are classified according to the following categories: “Low” if BMI<18.5; “High” if BMI>25; and “Ideal” if 18.5≤BMI≤25.
   d) Show all people's attributes in ascending order of BMI.
   e) Show standard deviation of weights and heights.
   f) Show people who are overweight.

11. Create a NumPy 3x10 array, with actual random values, evenly distributed between 10 and 20 (this one, exclusive). Then automatically identify the nearest value of 15 on each of the lines.

12. Using the Pandas package and, in particular, its DataFrame structure, remember the fateful numbers of COVID-19 in the first year it struck the world. Start by importing the *dataset* with the daily report, on November 8, 2020[1], of COVID-19 in the world, made available by Johns Hopkins University, to a Pandas DataFrame, [2] through the following link:
`https://raw.githubusercontent.com/CSSEGISandData/COVID-19/master/csse_covid_19_data/csse_covid_19_daily_reports/11-08-2020.csv`
Then analyze this dataset, performing the operations described in the following paragraphs.
    a) Start with a quick inspection of the *dataset*, looking at the columns that make it up (quantity, names, and types of values) and the number of records.
    b) Try to find a method from the DataFrame itself that gives us all the previous information right away. Then conclude your short inspection of the *dataset* with a preview of its first and last 5 lines.
    c) Create a new *dataset* with the number of confirmed, recovered, active, and fatality cases by country, and sort it in descending order of confirmed cases.
    d) From the previous table, check where Portugal is in the *ranking* of countries with the most confirmed cases, and place our country on the world stage in terms of location and dispersion measures. Finally, view all the rows in the table with a single *output*.
    e) Print, in descending order, the 10 countries with the most fatalities.
    f) Determine the Portuguese and global COVID-19 case fatality rates.
    g) Count the increase that the main indicators suffered in the previous 24 hours, by country: number of confirmed cases, recovered, active and fatalities. To do this, start by loading the previous day's dataset and checking that there is a complete agreement between the lines of the two *datasets*.

13. With the help of the *Matplotlib and Seaborn packages, continue to analyze the* datasets *from the previous exercise, but now from a more visual perspective, creating the graphs requested in the following paragraphs.*
    a) Start by transferring the DataFrames from the previous exercise to a new *notebook/file*.
    b) Graphically visualize the degree of correlation between the numerical variables of the daily COVID-19 report that we imported from the previous year.
    c) Create a scatter plot that relates confirmed cases to the number of fatalities, using either the *Matplotlib or Seaborn packages.* Tell us, then, if from the graph, it will be possible to take any conclusions regarding the variability of the lethality rate between countries?
    d) Overlay on a bar chart the active cases, recoveries, and fatalities of the 10 countries with the most confirmed cases, and draw your conclusions.

[1] It was on March 11, 2020 that the WHO characterized COVID-19 as a pandemic. But as early as January 30, 2020, the WHO had declared that the outbreak of the new coronavirus constituted a Public Health Emergency of International Concern.
[2] COVID-19 Data Repository by the Center for Systems Science and Engineering (CSSE) at Johns Hopkins University (url: https://github.com/CSSEGISandData/COVID-19).

14. With the Titanic dataset, available in the Seaborn package, produce the following graphics.
    a) A bar chart showing the average age of the men, women and children who boarded the Titanic.
    b) A graph showing the dispersion of ages in each group of people: men, women and children.
    c) A graph showing the survival rate of men, women and children.
    d) A graph showing the survival rate of men, women and children, by cabin class.
    e) A chart showing the average fare paid by men, women and children, per cabin class.

15. Based on the Iris dataset, also available in the Seaborn package, graphically show that the length and width of the petals of an iris plant can be used to determine, with a correct high, the type of plant concerned: silky, versicolor or virginica. Then check that there is no other pair of attributes that together allow for a better separation of the three plant types.

16. As we may know, in a simple linear regression problem we try to establish a linear relationship between an independent variable (explanatory variable) and a dependent variable (the target variable). But in this exercise we will take our learning one step further, creating a multiple linear regression model, which relates one dependent variable to several independent variables.
    a) From IPB.virtual, download the housing_dataset.csv and housing_descricao.pdf files to a local folder, containing a classic *dataset* and its description, respectively. It's a *very popular dataset* among the machine learning community, with prices and other housing data in Boston. Then start by importing the respective dataset to your desktop and analyze it. Change the name of the last column to 'y' to identify your target variable.
    b) Even before making any decisions about the model that we are going to develop, put aside 20% of the data in the *dataset*, so that in the end we can test our model with data that it has never seen.
    c) This *dataset* is perfectly suited to regression problems, using the median house price as the target variable. Not all of the remaining 12 variables may have the necessary explanatory power to make a positive contribution to the training model. Then, for predictor variables, choose only the two that have the highest correlation with the variable to be predicted.
    d) Using Scikit-learn, finally create the multiple linear regression model, training it, and test it with the subset of data we have set aside for this purpose and using R^2 as a performance metric.
    e) Now use the developed model to predict the price of a house with the characteristics of the first copy of the test set. Compare it to the actual price.
    f) Check to see if the performance of the model can be improved by adding new explanatory variables.
    g) Save the training and test subsets to disk for future use.

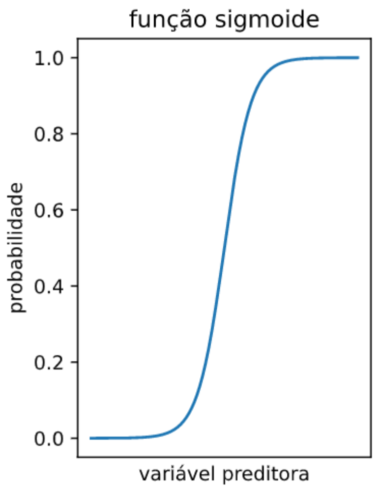
17. Linear regression is not always the best solution to capture the relationship between variables. Sometimes this relationship can be modeled more effectively, for example, by the curved line represented by a polynomial function. In this exercise we will try to increase the ability to predict housing prices in Boston with this new form of modeling: polynomial regression.

a) Using Scikit-learn, create, educate, and test an order 2 polynomial regression model, with two independent variables (use the two most correlated with the dependent variable) – being x_1 and x_2 the independent variables, their relation to the dependent can be represented by the quadratic function $y = a + bx_1 + cx_2 + dx_1x_2 + ex_1^2 + fx_2^2$.
b) Use the developed polynomial regression model to predict the price of a house with the characteristics of the first example of the test set. Compare it to the actual value.
c) See if you can further increase the predictive ability of the model, either by using more explanatory variables or by increasing the polynomial order of the model.
d) Final question just for reflection. You've probably even managed to come up with a model that has excellent predictability. But can it be expected to retain that ability when used with other test sets? In fact, if we think about it, the final version of the model we arrived at ended up being influenced by the test data itself, as it was based on successive tests that we made options to improve it. That is why it would be very useful to have another test set, completely new, that would allow us to make the final evaluation of the final model. This would then be the real test set. The other dataset that was used in the successive intermediate tests during the refinement of the model is called the "validation set". Therefore, whenever we want to refine the model under development in several iterations, we should start by partitioning the initial *dataset* into three subsets: training data, validation data, and test data.

18. Logistic regression is another supervised ML algorithm. However, despite being called "regression", it is a classification algorithm. Instead of predicting the value of a numerical variable, it estimates the probability that the value of a categorical variable will take on a specific class. This probability is estimated by the sigmoid function,

$$y = \frac{1}{1 + \mathrm{e}^{-(\beta_0 + \beta x)}}$$

which converts the value of the predictor variable to a numeric value between 0 and 1. It is during the training of the model that the coefficients β_0 and β are automatically adjusted to the data.
In this exercise we will create a simple logistic regression model for a *binary classification* dataset.



a) With the load_breast_cancer() function of the Scikit-learn datasets package, upload to your desktop a classic *dataset* used for binary classification, containing breast cancer registries in the state of Wisconsin, with 30 predictive attributes and respective diagnosis (benign/malignant). Start by inspecting this *dataset* and convert it to DataFrame format.
b) Graphically show that the first two attributes of the *dataset* help diagnose disease severity. Build a scatter plot with these two predictive variables, marking malignant cases in red and benign cases in blue.
c) Reserve 30% of the data for the test phase, opting for a properly balanced partition.
d) Using Scikit-learn, create the logistic regression model that allows us to classify the type of disease (benign/malignant) based on the 30 available predictors, training it, and calculate its success rate.

e) The accuracy rate is not, by itself, sufficient to evaluate the true performance of a model, particularly in *datasets* with class imbalance. Observe better the model's capability by using metrics that allow a more consistent (more reliable) assessment of its true performance, such as the confounding matrix, f1 measure, ROC curve and AUC value.

f) Finally, use the model developed to predict the severity of the disease from the first example of the test set, based on the 30 indicators. What is the probability of this diagnosis being made?

19. As Random Forests (RF) are a learning method based on committees (also called *ensembles* or experts), which attempts to create a strong classifier from multiple decision trees. Like most ML methods, RFs can be used for either classification or regression. In order to exemplify their use in a simple way, let's apply them to our already well-known *iris* dataset. Although it is focused on classification, we will in this exercise understand how this same set of data can be used for regression – the student can then easily develop an analogous model of RFs for classification.

    a) As we know, the Scikit-learn iris dataset consists of 4 predictors (width and length of sepal and petal) and a categorical target variable (plant species). To have a regression problem, assume that it is the length of the petal that we want to predict and that its species is one of the predictor variables. Then start by creating the subsets that will be used for training and testing the respective model.

    b) Create a first, non-fine-tuned version of the RF model that allows us to predict the length of the petal of an iris flower, training it and test it by calculating its coefficient of determination (R^2).

    c) Try optimizing the model by using cross-validation in fine-tuning some of your hyperparameters.

    d) Graphically visualize the relationship between model predictions and actual values.

    e) Represent in a bar graph the importance of each of the explanatory variables in predicting petal length.

20. Repeat the study from the previous exercise, but supporting your prediction model on a single decision tree. Compare the results obtained with the previous ones.

21. Re-repeat the previous prediction study now using the KNN algorithm (the K nearest neighbors).

22. In exercise 16, we created a simple linear regression model to predict the price of a house in Boston, probably achieved a coefficient of determination R^2 which did not exceed the 80%. In this exercise we will try to further increase the accuracy of that prediction by supporting our model on a support vector machine (SVM), a more sophisticated ML algorithm.

    a) Using the training and test subsets that we save in exercise 16, start by quickly creating an SVM regression model with his base configuration, No need to worry about fine-tuning or data normalization issues. Training and test the model created. Surprised by the result?... Proceed to the next step.

    b) With the describe() method of DataFrames we may easily verify that the range of variation of the predictive variables is quite distante, with some of them assuming amplitudes several hundred times higher than those of others. Since SVMs are quite sensitive to the scale of predictive variables, refine the model by properly normalizing the data. Notice the difference in performance.

c) Try to further improve the performance of the model by tuning it properly.

d) Predict the price of a house in Boston with the characteristics of the first copy of the test set. Compare it to the actual price. Also predict the price of the last copy.

23. Now use artificial neural networks to try to further increase the ability to predict housing prices in Boston, replicating the procedure followed in the previous year. Compare the results.

24. In exercise 19in order to be able to use *the iris dataset* to induce a regression model, instead of classification, we have to swapped the roles of the variables 'species' and 'petal length'. As a result, it now has, among its predictive variables, one of a categorical type: the 'species'. Perhaps you will now realize, with the knowledge you have acquired in meantime, that the solution found at the time contains a slight inaccuracy. In fact, since the 3 species (categories) were represented by integer codes (0, 1 and 2), the values of this variable ended up being misinterpreted by the model as numerical quantities. Correct this imperfection by properly encoding the 'species' variable.

25. Using the iris dataset in a more conventional way, develop several multi-class classification models that can identify the plant species based on the dimensions of its petals and sepals. To build each of these models, use a different ML algorithm from the ones you studied (logistic regression, *random forest*, SVM and artificial neural networks). Compare the performance achieved by the different methods.

26. Suppose a clothing company decided to produce a new model of shorts, and asked you with the difficult task of finding the set of sizes to produce that best fit your target audience. The size of this type of piece is defined by either the waist circumference or the length of the upper leg of the standard person. Suppose then that you are provided with *a* ***dataset*** containing these measurements for a significant sample of the population concerned. In order to be able to identify, from this *dataset, the* various sizes of the model to be produced, you will need to subdivide the population of that sample into several subsets, as homogeneous as possible. Then, you will only have to propose a specific size of shorts for each of these groups of people. It is precisely this type of problem that ***clustering*** algorithms try to address, relying on unsupervised learning techniques. The aim of this exercise is to use K-Means, one of the *simplest and most widely used* clustering algorithms. Notice that all the ML models you've developed so far have been based on supervised learning techniques.

    a) Start by importing from https://data.world/rhoyt/body-measurements The *dataset* BMX_G.csv with the necessary information. In this *dataset* there are 26 standard measurements of 9338 individuals, such as height, weight, BMI and other more specific measurements, including the two that you will need (bmxleg and bmxwaist columns), and which we will now identify simply by leg (upper leg length) and waist (waist circumference). After proceeding with the selection and cleaning of the data, graphically show the dispersion of these two attributes.

    b) Using the K-Means algorithm, apply the ***clustering*** technique to the data, in order to find the 3 sizes of shorts that best fit the set of people considered.

c) Show again the dispersion of the two measurements, leg and waist, now also representing their association with the *clusters* and their centroids.

d) Evaluate the quality of *the clustering* you have achieved by calculating his Silhouette Coefficient. It is a performance measure that measures both the cohesion within each *cluster* and the separation between them. It varies between -1 and 1. Values close to 0 indicate overlapping clusters*, and close to -1 instances allocated to* the wrong clusters.

e) Now try to find the best set of sizes by optimizing the k parameter.