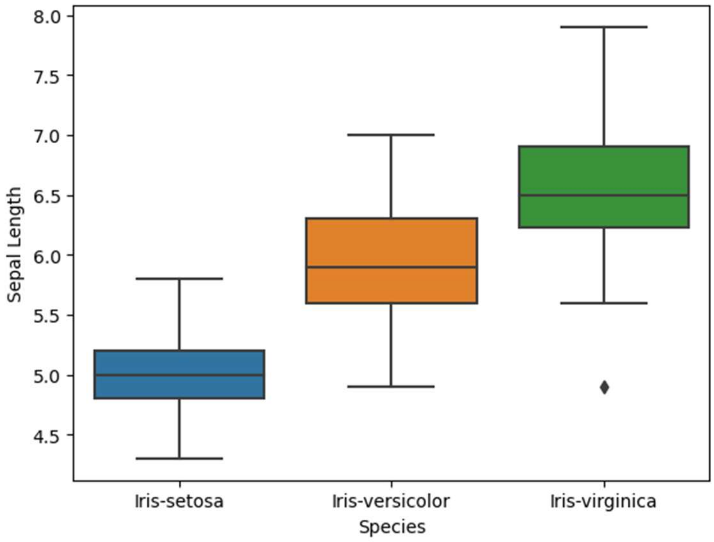
https://github.com/IrishRugbyman/QuentinLambolez_IrisProject

I made a Python script that performs a k-NN classification on the IRIS flower dataset. I first load the dataset, perform some data analysis (including graphs), pre-process the data, train a k-NN classifier using a pipeline, and evaluate the.

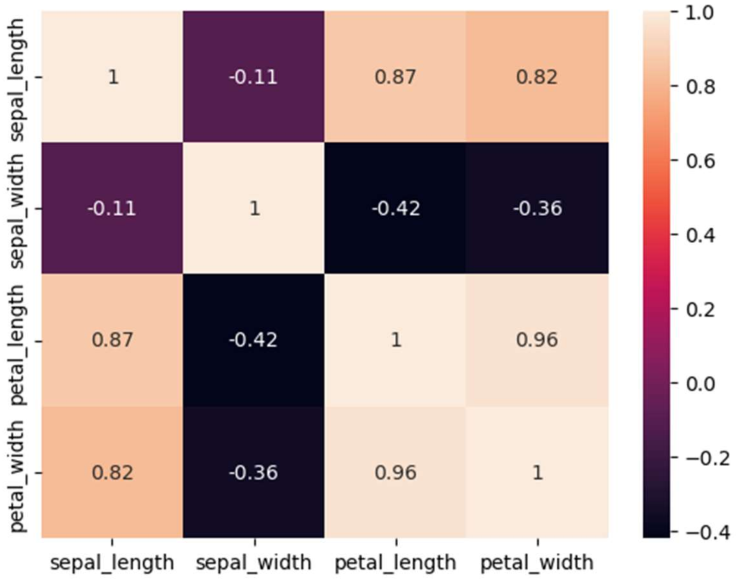
Data Analysis

It's necessary to perform some data analysis to get a better understanding of the data. In this script, I used the `seaborn` library to create visualizations of the dataset.

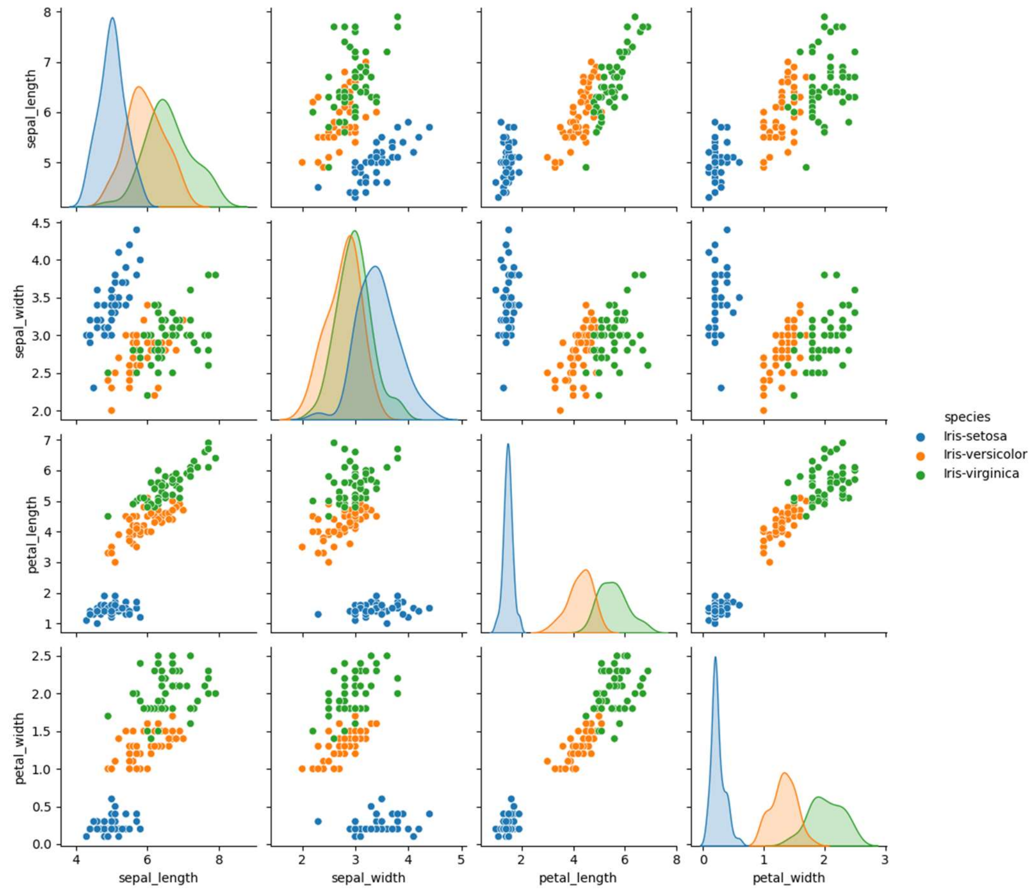
First, I created a box plot to compare the sepal length across the three different species of flowers. The *"sns.boxplot()"* function from the seaborn library is used for this.
We can see that the sepal length for the Iris Setosa species is much smaller than the other two species.



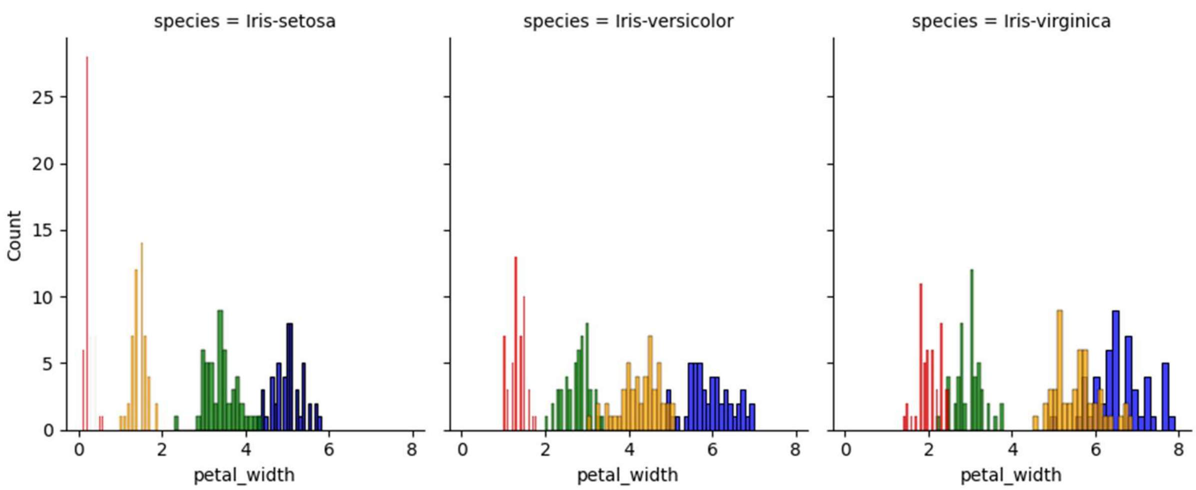
Next, I created a heatmap to visualize the correlation between the features. The *sns.heatmap()* function is used again.
It's clear that the petal length and petal width are highly correlated with each other, while the sepal length and sepal width are less correlated.



Then, I made a pairplot to visualize the relationships between the features and the target variable. The *sns.pairplot()* function is used once again.
We learn that the petal length and petal width are the most important features for distinguishing between the three different species of flowers.



Finally, I created a FacetGrid plot to visualize the distribution of each feature for each species of flower. I first created a FacetGrid using the *sns.FacetGrid()* function, passing in the dataset and the 'species' column as the variable.
Then, I used the *g.map()* function to apply a histogram plot using the *sns.histplot()* function to each column. The resulting plot allows us to easily compare the distribution of each feature for each species of flower.



Pre-processing

After performing some data analysis, the next step is to pre-process the data:

- Splitting the data into features and target variables.
- Encoding the target variable.
- Splitting the data into training and test sets.

The *iloc()* function from the pandas library is used to split the data into features (X) and target (y).
Then, the *LabelEncoder* class from the sklearn.preprocessing library is used to encode the target variable.
And, lastly, the *train_test_split()* function from the sklearn.model_selection library is used to split the data into training and test sets.

Training the k-NN classifier

After pre-processing the data, the next step is to train the k-NN classifier. In this code, I used a pipeline to connect the steps of getting the data ready and the actual classification process.

The k-NN classifier is trained using the training set. The *GridSearchCV* class is used to perform a grid search over the hyperparameters of the k-NN classifier. The hyperparameters include the number of neighbors, the weight, and the distance.

Evaluating the classifier

After training the k-NN classifier, the next step is to evaluate its performance on the test set. In this script, the *classification_report()* function and the *accuracy_score()* function to evaluate the classifier.

```
                 precision    recall  f1-score   support

    Iris-setosa       1.00      1.00      1.00        10
Iris-versicolor       1.00      1.00      1.00         9
 Iris-virginica       1.00      1.00      1.00        11

       accuracy                           1.00        30
      macro avg       1.00      1.00      1.00        30
   weighted avg       1.00      1.00      1.00        30

Accuracy: 1.0000
```

We can see that the precision, the recall, the f1-score are all equal to 1. This may come from the fact that the dataset does not have a lot of data so the model does not have enough data to work with.