ML Supervised Learning

Pierre Jeannin - Pierrick Gouerec

PART 1:

The script in Part1/part1.py is used to generate an artificial dataset with eight columns, each filled with different types of random data. The data is then saved to a CSV file named artificial_dataset.csv.

Here's a brief explanation of the imported libraries:
- numpy: This library is used for numerical computations in Python. In this script, it's used to generate random numbers following different distributions (normal, integer) and to perform operations on these numbers.
- pandas: This library is used for data manipulation and analysis. In this script, it's used to create a DataFrame from the generated data and to save this DataFrame to a CSV file.

PART 2:

The script in Part2/part2.py is used to create a dissimilarity matrix for the given dataset (saved as 'dataset.csv'). As the dataset provides informations about people, we choose to give more weight to the city and to the age. We created a function custom_metric to calculate the dissimilarity matrix based on the weight of the parameters.

Here's a brief explanation of the imported libraries:

- numpy: This library is used for numerical computations in Python. In this script, it's used to calculate our metric.
- pandas: This library is used for data manipulation and analysis. In this script, it's used to load the DataFrame from the provided CSV file.

PART 3:

The script in Part3/part3.py is used to predict the winner of a NBA game. To achieve this we used the LogisticRegression model from the sklearn library and the Support Vector Classifier model from the same library. We tried different parameters and the best results were with the following :

Logistic Regression :
- solver: lib linear
- C: 0.001
- class_weight: balanced

Support Vector Classifier :
- kernel: linear

For this part we compared the parameter with manual tests. We modified each parameters and ran the script several times to find the best parameters.

We also used the accuracy_score function from sklearn to calculate the accuracy of each model.

We used numpy to load the dataset from the NPX files.

PART 4:

The script in Part4/part4-try4-final.py is used to predict the amount of electricity product by a wind farm.

For this problem, we listed several models from three libraries, sklearn, XGBoost and LightGBM :
- Ridge
- Lasso
- MLP Regressor
- SVR
- AdaBoostRefressor
- XGBoost
- LightGBM

We listed all parameters and possible values to iterate through them and find the best configuration. We also use the StandardScaler class from sklearn to preprocess the data.

After that we ranked the models based on their R2 score and here are the results :

1. Lasso: 0.86
2. Ridge: 0.66
3. SVR: 0.62
4. LightGBM: 0.54
5. XGBoost: 0.39
6. AdaBoostRegressor: 0.38
7. MLPRegressor: -0.36

We managed to have a R2 score above 0.85 with the Lasso model.

We left the other tries for this part in the other python files in the Part4 directory.

PART 5 :

Dataset link : https://www.kaggle.com/datasets/arunjangir245/boston-housing-dataset?resource=download

I want to build a model that estimate the price of house in Boston. To do this I found a dataset that contains various informations aout houses such as the proportion of lare residential lots, the average number of room per dwelling … There are 506 lines in this dataset, so I will split it in two to have a train dataset and a test dataset.

Here is an explanation of some values of the dataset :

- crim: This represents the per capita crime rate by town. It gives an idea about the safety of the neighborhood.
- zn: This is the proportion of residential land zoned for lots over 25,000 sq.ft. It provides information about the spaciousness of the area and the potential for large homes.
- ndus: This is the proportion of non-retail business acres per town. It indicates the extent of commercial activity in the area.
- Chas: This is a binary variable indicating if the property is near the Charles River (1 for yes, 0 for no). Being near a river could be a desirable feature for some homeowners.
- nox: This represents the concentration of nitrogen oxides in the air, which can be a measure of air pollution in the area.
- m: This is the average number of rooms per dwelling, which can give an idea about the size of the houses in the area.
- age: This is the proportion of owner-occupied units built before 1940. It can provide information about the age and potentially the style of the houses.
- dis: This represents the weighted distances to Boston employment centers. It can be an important factor for people who work in these centers and want to minimize their commute.
- ad: This is an index of accessibility to radial highways. Easy access to highways can be a positive factor for people who commute by car.
- tax: This is the property tax rate per $10,000. It gives an idea about the ongoing costs of owning a property in the area.

In the a python script a made a analyze_data function that calculate the mean and the median for each feature of the dataset. Here are the results :

Feature	Mean	Median
crim	3,61	0,26
zn	11,36	0,00
indus	11,14	9,69
nox	0,55	0,54
rm	6,28	6,21

Feature	Mean	Median
age	68,57	77,50
dis	3,80	3,21
rad	3,80	3,21
tax	408,24	330,00
ptratio	18,46	19,05
b	356,67	391,44
lstat	12,65	11,36
medv	22,53	21,20

In a preprocess_data function I fill ed the NaN values with the mean of the column, and I used the fit_transform function of StandardScaler to have a mean of 0 and a standard deviation of 1. It can help the model converge faster and can also prevent certain features from dominating others due to differences in scale.

In the process of identifying the best model for the given dataset, several models were evaluated. These models include Linear Regression, Decision Tree Regressor, Random Forest Regressor, and Support Vector Regression (SVM). Each model was initialized with a set of parameters that were fine-tuned to optimize the model's performance. For instance, the fit_intercept parameter was adjusted for the Linear Regression model, the criterion and splitter parameters for the Decision Tree, the n_estimators and criterion parameters for the Random Forest, and the C and kernel parameters for the SVM. The performance of each model was then evaluated using the R-squared (R2) score, which measures the proportion of the variance in the dependent variable that can be predicted from the independent variables. The model with the highest R2 score was considered the best model for the dataset.

Here are the result obtained with the different models :

Model: random_forest
Best Score (R2): 0.6446378601357596
Best Parameters: {'criterion': 'absolute_error', 'n_estimators': 200}

Model: svm
Best Score (R2): 0.4748884106198049
Best Parameters: {'C': 10, 'kernel': 'rbf'}

Model: decision_tree
Best Score (R2): 0.38190473762633237
Best Parameters: {'criterion': 'poisson', 'splitter': 'random'}

Model: linear_regression
Best Score (R2): 0.351065723748072
Best Parameters: {'fit_intercept': True}

Here's a brief analysis of the results:

1. Random Forest: This model performed the best among all the models with an R2 score of approximately 0.645. The best parameters for this model were found to be 'absolute_error' for the criterion and 200 for the number of estimators (n_estimators). This means that a Random

Forest model with 200 trees and using absolute error as the quality of a split performed the best in predicting the target variable.

2. Support Vector Machine (SVM): This model had the second-best performance with an R2 score of approximately 0.475. The best parameters for this model were a C value of 10 and 'rbf' for the kernel. This suggests that an SVM model with a regularization parameter of 10 and using the Radial basis function (rbf) kernel performed the second best.

3. Decision Tree: This model had the third-best performance with an R2 score of approximately 0.382. The best parameters for this model were 'poisson' for the criterion and 'random' for the splitter. This means that a Decision Tree model using the Poisson criterion for quality of a split and random splitter performed third best.

4. Linear Regression: This model had the least performance among all the models with an R2 score of approximately 0.351. The best parameter for this model was 'True' for fit_intercept, which suggests that a Linear Regression model with the y-intercept included in the model performed the least well.

In conclusion, based on the R2 scores, the Random Forest model seems to be the best model for this particular dataset and problem, followed by SVM, Decision Tree, and Linear Regression

The results indicate that predicting house prices is a complex task due to the numerous parameters involved. Despite this, our models, especially the Random Forest model, have shown that it's possible to achieve reasonable results even without specific domain knowledge. However, understanding each feature's significance could potentially improve the model's accuracy.