PW3-Supervised Machine Learning : A basic supervised Learning algorithms

1 Summary

In this lesson, we focus on supervised learning problems: the aim is to develop algorithms able to learn predictive models. Based on labeled examples, these models will be able to predict the label of new objects. The aim of this PW is to develop the general concepts with Scikit-learn that enable us to formalize this type of problem. Here a definition of supervised learning concept:

Definition: Supervised learning is a machine learning technique where an algorithm learns from labeled data to make predictions or decisions. It is widely applied in various domains, such as classifying emails as spam or non-spam, predicting housing prices based on different factors, or detecting anomalies in datasets. Scikit-learn, a popular Python library, provides a comprehensive set of tools and algorithms for supervised learning tasks.

Our Objectives are

- Formalize a problem as a supervised learning problem;
- Choose a cost function;
- Look for its generalizability.

Problem formalization

Supervised learning problem can be formalized as follows: given n observations $\{x_1, x_2, ...x_n\}$, where each observation $x_i$is an element of the space of observations $\mathcal{X}$, and their labels $\{y1, y2, ...yn\}$, where each label yi belongs to the label space $\mathcal{Y}$. The aim of supervised learning is to find a function $f : \mathcal{X} \to \mathcal{Y}$ such that $f(x) \approx y$, for all pairs $(x, y) \in \mathcal{X} \times \mathcal{Y}$ having the same relationship as the observed pairs. The set of D = $(x_i, yi)i = 1, ..., n$ forms the training set. In this lesson, we will consider three special cases for Y:

- $Y = \mathbb{R}$: we deal with a regression problem;
- $Y = \{0, 1\}$: this is called a binary classification problem, and observations whose label 0 are called negative, while those with a label of 1 are called positive. In some cases, it is mathematically useful to use $Y = \{-1, 1\}$;

- $Y = \{1, 2, ...C\}, C > 2$: this is known as a multi-class classification problem.

In many situations, we assume we have $\mathcal{X} = \mathbb{R}^p$ where the observations are represented by p variables. In this case, the matrix $X \in \mathbb{R}^{n \times p}$ such that $X_{ij} = x_j^i$ is the j^{th} variable of the i^{th} observation is called the data matrix.

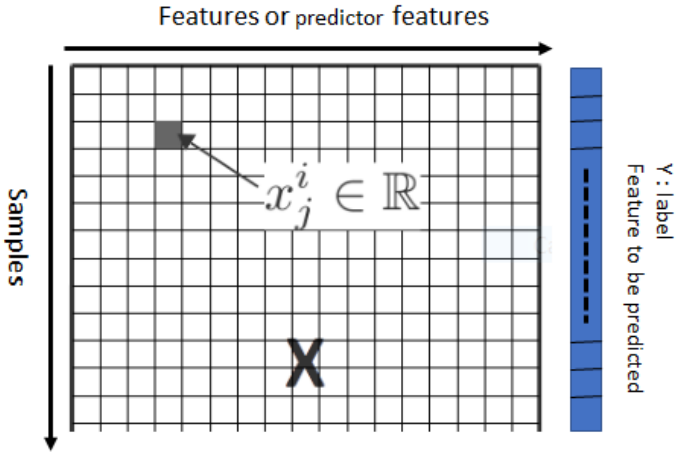


Figure 1: Supervised learning key elements

Decision

In the case of a classification problem, the predictive model can take the form of a function f with values in $\{0, 1\}$, or use a real-valued intermediate function g, which associates a higher score to observations. This score is, for example the probability that the observation belongs to the positive class. We then obtain f by thresholding g ; g is called the decision function.
(Decision function) In a binary classification problem, a decision function g, or a discriminant function $g : \mathcal{X} \to \mathbb{R}$ such that $f(x) = 0$ if and only if $g(x) \leq 0$ and $f(x) = 1$ if and only if $g(x) > 0$.
(Decision Space) In the case of binary classification, the function g split the observation space $\mathcal{X}$ into two decision Areas A_0 and A_1 such that:
$A_0 = \{x \in \mathcal{X} | g(x) \leq 0\}$ and $A_1 = \{xin\mathcal{X} | g(x) > 0\}$.

Cost Function

Solving a supervised learning problem means finding a function $f \in F$ whose predictions are as close as possible to the true labels, over the whole space X . It is a Minimizing empirical risk. To formalize this, we use the notion of cost function.
(Cost function) A cost function $L : Y \times Y \to \mathbb{R}$, also called loss function or error function, is a function used to quantify the quality of a prediction: $L(y, f(x))$ is greater the closer the label $f(x)$ is to the true value y.

2 Exercises

Exercise 1 : Scale the data

You will need the dataset Wine quality, with Quality column has two classes good and bad. We have to classify our wine in these two classes based on the features:

```
 #   Column                Non-Null Count  Dtype
---  ------                --------------  -----
 0   fixed acidity         1599 non-null   float64
 1   volatile acidity      1599 non-null   float64
 2   citric acid           1599 non-null   float64
 3   residual sugar        1599 non-null   float64
 4   chlorides             1599 non-null   float64
 5   free sulfur dioxide   1599 non-null   float64
 6   total sulfur dioxide  1599 non-null   float64
 7   density               1599 non-null   float64
 8   pH                    1599 non-null   float64
 9   sulphates             1599 non-null   float64
 10  alcohol               1599 non-null   float64
 11  quality               1599 non-null   object
```

1. Read the Dataset, store the data into a variable ***'wine_data'*** and print the basic information and statistics. The date could be found from this location: Red-Wine
2. transform the wine_data into a dataframe using ***pandas.DataFrame.*** function and as parameters ***wine_data.values***, ***wine_data.columns***
3. check if there are missing values
4. Drop the quality column to set the rest of the data into a variable X using the function ***drop(data,axis)***
5. copy the column ***'quality'*** into a new variable Y
6. As you can see, the numeric data are with different scales and to avoid misinterpretation of ML results, we want to bring all features to a common scale. Many techniques exist. we commonly use two main techniques: standardization and normalization.

   Data Standardization or *z-score normalization*

   scales the data in a way that the average of each feature becomes 0, and the standard deviation becomes 1. This process is also known as ***z-score normalization***. It assumes that the data fills a Gaussian distribution. Standardization is robust to outliers because it uses the statistical state of data with mean and standard deviation, which are less affected by extreme values. It is commonly used when applying machine learning algorithms that assume normally distributed features or when you want to scale your data to unit variance.
   How to calculate : $x_{std} = \frac{x-mean}{std}$. With scikit learn, we have the class ***standardScaler*** to do this data transformation

   The second technique is min-max scaling and many people call it the normalization.

min-max data transformation

min-max Normalization scales each feature individually where the values fall within a specified range, usually [0, 1]. This process does not assume any particular distribution of the data. The resulting normalization is sensitive to outliers because it uses the minimum and maximum values to perform the scaling. This method is commonly used when you want to bring all features to a common scale, especially when using algorithms that rely on distances or gradients, such as k-nearest neighbors (KNN) or gradient descent-based methods.
How to calculate : $x_{min-max} = \frac{x-min}{max-min}$. With scikit learn, we have the class ***MinMaxScaler*** to do this data transformation

7. Create the training and test datasets : x_train, x_test, y_train, y_test with 0.3 for test and random_state=20.
8. Use the correct scale and apply it to the train and test data set.
9. You have to apply a linear regression model to predict the quality. Try to formalize the problem
10. with scikit learn, use LinearRegression() class after the import as following : from sklearn.linear_model import LinearRegression
11. fit your model
12. predict on your test dataset
13. calculate the mean square error on the test dataset using : from sklearn.metrics import mean_squared_error, r2_score

Exercise 2 : Prediction pipeline and evaluation

We can remark, the lack of the linear model to predict the right quality based on the input features : Mean Squared Error mse: 0.6215899256984494 is very high and the R-squared R^2: 0.25761478584858577 is low. With a prediction based on a linear regression model, we needed two parameters to evaluate the model performance.

Mean Square Error (mse)

The mean square error is calculetd as the average of the squared difference between each pair $(ypredict_i, ytest_i)$ for $i = 1..len(ytest)$. The formula of $mse = \frac{len(ytest)^1}{\sum(ypredict_i, ytest_i)^2}$ for $i = 1, .., len(ytest)$.

However to evaluate the goodness of fit of a regression model we need the (R^2).

R-squared (R^2)

R^2 is a statistical measure. It quantifies the proportion of the variance in the dependent variable (the predicted variable) which is explained by the independent variables (the features or input predictors) in the regression model. In short, R-squared tells you how well the model fits the data. R^2 is calculated as $R^2 = 1 - \frac{\sum(ypredict_i, ytest_i)^2}{\sum(ytest_i meanTest)^2}$.
R^2 is ranged between 0 and 1. It measures the proportion of the variance in the dependent variable that is predictable from the independent variables. For example, and R^2 value of 0.7 means that 70% of the variance in the dependent variable is explained by the independent variables in the model. But in our case, we only obtained 0.2 (20%), which means that our model is unable to explain wine quality on the basis of input characteristics.

We would compare the same prediction by changing different parameters as: random_state, the transformer, etc. To do this in a right way, we have to implement pipelines.

1. write a python function named ***LR_Pipeline*** with the signature

```
    def LR_Pipeline(scaler):
        #put your code
        return my_pipeline

```

allowing you to create your pipeline with the following steps : 1) scale data, 2) predict. Your function need the scaler as a parameter to be able to call MinMaxScaler() as standardScaler()

2. call your function with the standardscaler and compare the results with a random-state equal to 20.

Comment your results!

evaluation	R^2	mse
scaler=min-max, test_size=0.1	?	?
scaler=standard, test_size=0.1	?	?
scaler=min-max, test_size=0.2	?	?
scaler=standard-max, test_size=0.2	?	?

3. Plot the quality prediction VS. the real quality in the case of the best configuration of your predictor model.

```
# Visualize the predicted vs. actual values
plt.scatter(y_test, y_pred)
plt.xlabel("Real Wine Quality")
plt.ylabel("Predicted Wine Quality")
plt.title("Real vs. Predicted Wine Quality")
plt.show()
```

4. If we note $\Theta = (\theta_1, .., \theta_1 1)$ and B the estimated parameters of our predictor model, try to display them from your pipeline and write the exact formula of your predictor.

```
  #Access to parameters
    predictor.coef_
  # Access the intercept (bias) term
    intercept = predictor.intercept_
```

Exercise 3: Logistic regression

We try to explain our wine quality prediction to a classification problem aiming to distinguish between good and bad wines. Let suppose we have two binary classes defined as follow: if the wine quality is greater than 5, the label will be 1 and 0 else.

1. transform the target output feature y to be $\{0, 1\}$

```
    (wine['quality'] >= 5).astype(int)

```

2. define an other function for your pipeline taking as parameters the scaler and the predictor model.

```
    def Reg_Pipeline(scaler, model):
    #put your code here
    return my_pipeline

```

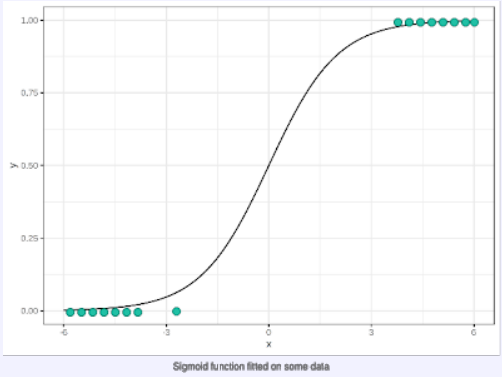
3. Fit and predict your pipeline with a standardscaler and LogisticRegression model.

Logistic Regression

The logistic regression is a predictive analysis of statistical method:

- It is used to predict a binary outcome (1 or 0, Yes or No, True or False) given a set of independent variables. Logistic regression is a classification algorithm used to assign observations to a discrete set of classes. For this data set, it is better to using binary logistic regression with modified "quality".
- Logistic regression is a statistical model used to study the relationships between a set of qualitative variables Xi and a qualitative variable Y. It is a generalized linear model using a logistic function as the link function.
- A logistic regression model can also predict the probability of an event occurring (value of 1) or not (value of 0), based on the optimization of regression coefficients. This result always varies between 0 and 1. When the predicted value is above a threshold, the event is likely to occur, whereas when this value is below the same threshold, it is not.
- The aim of logistic regression is to find a probability function P such that we can calculate : $y = \{1 \textbf{ if } P(X) \geq threshold, 0 \textbf{ if } P(X) < threshold\}$

The function P that fulfils these conditions is the sigmoid function, defined on $\mathbb{R}$ with values in [0,1]. It is written as follows: $\frac{1}{1+e^{-x}}$. The sigmoid function transforms $P(x)$ into a value between 0 and 1. When x is large (positive or negative), the sigmoid approaches 1 or 0, respectively. When x is 0, the sigmoid evaluates to 0.5.



Sigmoid function fitted on some data

Formulation of the logistic regression: $L(x) = \frac{1}{1+e^{-\sum \theta_i x_i}}$. Then the supervised classification is an optimisation problem looking for the best θ_i allowing the sigmoid to fit our outcome y

4. Now we try to evaluate the classification performance.

```
    # Evaluate the model's performance
from sklearn.metrics import accuracy_score, classification_report, confusion_matrix

accuracy = accuracy_score(y_test, pred)
confusion = confusion_matrix(y_test, pred)
classification_rep = classification_report(y_test,pred)

print("Accuracy:", accuracy)
print("Confusion Matrix:\n", confusion)
print("Classification Report:\n", classification_rep)
```

Evaluation metrics

Let note 0 as a negatif prediction and 1 as a positive one. We can obtain 4 measures: true positive (TP :true 1), true negative (TN: true 0), false positive (FP: false 1) and flase negative (FN : false 0). Let detail now each of them.
$TP = (y\_predict = 1$ **when** $y\_true = 1) = NB\_1\_predict/NB\_1\_true$
$TN = (y\_predict = 0 when y\_true = 0) = NB\_0\_predict/NB\_0\_true$

$FP = (y\_predict = 1 when y\_true = 0) = NB\_1\_Falsepredict/NB\_0\_true$
$FN = (y\_predict = 0 when y\_true = 1) = NB\_0\_predict/NB\_1\_true$

Precision $(P) = \frac{TP}{TP+FP}$
Recall $(R) = \frac{TP}{TP+FN}$
F-score = $2.\frac{PR}{P+R}$
The global accuracy is calculated as: $\frac{TP+TN}{len(y\_test)}$
Accuracy values range from 0 to 1, where:
An accuracy of 1 (100%) indicates that the model's predictions are entirely correct, and there are no prediction errors.
An accuracy of 0 (0%) indicates that the model's predictions are entirely incorrect, and none of the predictions are correct.
While accuracy is a widely used metric, it has some limitations, especially in situations with imbalanced datasets. In imbalanced datasets, where one class significantly outnumbers the other, a high accuracy score can be misleading. For example, if 95% of the instances belong to class A and only 5% belong to class B, a model that predicts all instances as class A will have a high accuracy of 95%. However, it fails to correctly predict any instances of class B, which may be the more critical class.
In such cases, other metrics like precision, recall, F1-score, or area under the ROC curve (AUC-ROC) may provide a more comprehensive assessment of a classification model's performance, as they take into account true positives, false positives, true negatives, and false negatives.
In summary, accuracy measures the overall correctness of a classification model's predictions, and it is an essential metric to consider. However, it should be used in conjunction with other metrics, especially when dealing with imbalanced datasets or when different costs are associated with false positives and false negatives

5. Check the imbalaced datasets problem by comparing the rate of trained 0 samples VS. 1 samples.

Exercise 4: Naive Bayes classification

In this exercise, we are looking for the improvement of Naïve Bayes to classify the wine quality. Naive Bayes is a simple algorithm aiming to estimate the probabilistic of a random event using the following concepts:

Naive Bayes

- Bayes' Theorem: Naive Bayes classifiers are based on Bayes' theorem, which calculates the probability of a particular event based on prior knowledge of conditions that might be related to the event. In the context of classification, it calculates the probability of a particular class label given a set of features.
- Conditional Independence: it assumes all features are conditionally independent of each other given the class label. While this assumption is often unrealistic in practice, Naive Bayes can still perform well, especially in cases with limited data and small number of features.
- Probability Estimation: Naive Bayes calculates the conditional probabilities of each feature given each class label and the prior probability of each class label. It uses these probabilities to estimate the probability of a specific class label given the observed features.
- Classification Rule: The classification rule in Naive Bayes selects the class label with the highest probability given the observed features. This is known as the Maximum A Posteriori (MAP) classification rule

During the training step, Naive Bayes estimates the probabilities and parameters needed for classification. It calculates the prior probabilities of class labels and conditional probabilities of features given each class. During the classification step, Naive Bayes uses the Bayes' theorem to calculate the posterior probabilities of class labels given the observed features. The class with the highest posterior probability is selected as the predicted class label.

1. Import the adequate class from Scikit learn to use the Naive Bayes, with a Gaussian distribution **GaussianNB**.
2. Use your pipeline with this new model.
3. Fit the model to the training set
4. Predict the classes of your test dataset
5. calculate the confusion matrix, the precision, recall, F-score and accuracy.
6. What is your conclusion?

Exercise 5: performance comparison between different models

We used three different models to predict wine quality. First, linear regression is used to predict the quality value, not the label. Secondly, two similar models, logistic regression and naive Bayes, are used to predict binary classes of wine quality. We therefore need to decide which model might be appropriate for our problem in the event that these models are comparable.

1. How we can compare Naive Bayes and Logistic Regression
2. use the AUC-ROC to decide which model might be validated to this classification problem.

Exercise 6: Do it from scratch

Now you have a new dataset describing the Breast cancer (Source Kaggle). Features are computed from a digitized image of a fine needle aspirate (FNA) of a breast mass. They describe characteristics of the cell nuclei present in the image. The 3-dimensional space is that described in: "K. P. Bennett and O. L. Mangasarian: "Robust Linear Programming Discrimination of Two Linearly Inseparable Sets", Optimization Methods and Software 1, 1992, 23-34".

This database is also available through the UW CS ftp server: ¿ ftp ftp.cs.wisc.edu ¿ cd math-prog/cpo-dataset/machine-learn/WDBC/

Also can be found on UCI Machine Learning Repository: `https://archive.ics.uci.edu/ml/datasets/Breast+Cancer+Wisconsin+%28Diagnostic%29`

The Attribute Information are:

1. ID number
2. Diagnosis (M = malignant, B = benign)

Ten real-valued features are computed for each cell nucleus:

1. radius (mean of distances from center to points on the perimeter)
2. texture (standard deviation of gray-scale values)
3. perimeter
4. area
5. smoothness (local variation in radius lengths)
6. compactness $\frac{(perimeter^2}{area-1.0})$
7. concavity (severity of concave portions of the contour)
8. concave points (number of concave portions of the contour)
9. symmetry
10. fractal dimension ("coastline approximation" - 1)

The mean, standard error and "worst" or largest (mean of the three largest values) of these features were computed for each image, resulting in 30 features. For instance, field 3 is Mean Radius, field 13 is Radius SE, field 23 is Worst Radius.

All feature values are recorded with four significant digits.

Missing attribute values: none. But you have to check. Class distribution: 357 benign, 212 malignant

To do: **Make a comparative study between Logistic regression and Naive Bayes.**