University of Bejaia
Faculty of Exact Sciences
Department of Operating Researchers
Academic Year: 2023/2024

First year Master SDAD (S2) Module: **Machine-Learning and Data mining**

TP 2: Regression (Supervised learning)

1/ Introduction

This second lab provides a general overview on how to build and evaluate a supervised learning algorithm such as a regression. We focus on linear regression (simple) and logistic regression.

2/ Learning Objectives

- Train and test data
- Making predictions
- Evaluating predictions

3/ Some vocabulary

Machine Learning is a vast and complex field. There are 4 important concepts that you use during all your machine learning projects, so you must be aware of that.

- *Dataset*: In Machine Learning, everything starts with a Dataset containing our data. In supervised learning, the Dataset contains the questions (y) and answers (x) to the problem that the machine must solve.
- *The model and its parameters*: starting from the Dataset, we create a model (a mathematical function). The coefficients of this function are the model parameters.
- *Cost Function*: When we test our model on the Dataset, it returns some errors. The sum of these errors is called the Cost Function.
- *Learning algorithm*: The central idea of Machine Learning is to let the machine find the parameters of the model that minimize the Cost Function.
- *Overfitting:* Overfitting occurs when the model fits the training data too closely, capturing noise or random fluctuations in the data. This leads to a model that performs very well on the training data but doesn't generalize well to new data.

  Example: Suppose we're building a linear regression model to predict house prices based on their size. If we have a dataset with outliers or unrepresentative data points, a complex machine learning model might try to fit to these points even if they don't truly represent the overall trend. This would result in overfitting, where the model is too complex relative to the size of the training data.
- *Underfitting:* Underfitting occurs when the model is too simple to capture the underlying structure of the data. This results in a model that fails to fit well even on the training data, leading to poor performance on both training and test data.

  Example: Let's revisit our house price prediction example. If we use a very simple linear regression model with just one feature (e.g., only the house size), it might underestimate the relationship between size and price, failing to account for other important factors like the number of bedrooms, location, etc. In this case, the model would be too simplistic to capture the complexity of the actual data.

  How to address these issues:

- To avoid overfitting, techniques such as regularization (e.g., L1 or L2) can be used to penalize overly complex models, or cross-validation can be employed to select optimal hyperparameters.
- To avoid underfitting, one can try more complex models or add relevant features to the data.

In summary, overfitting occurs when the model is too complex relative to the training data, while underfitting occurs when the model is too simple to capture the data's structure. Both issues need to be monitored and addressed to achieve high-performing and generalizable machine learning models (we'll see these concepts with more detail so far).

4/ Programming steps

To implement these 4 steps in Python, we need to take the following steps:

- Import all the necessary libraries.
- *Preparing the DataSet*. The machine receives data characterized by X variables (called features) and annotated with a y variable (called a label or target).
- *Select the model* (or estimator) the machine needs to learn, specifying the model's hyperparameters (for example, LinearRegression, …etc.).
- Train the model on data X and Y : **model.fit(X,Y)**
- Evaluate the model : **model.score(X,Y)**
- Use the model : **model.predict(x)**

5/ Software tools

You install anaconda (https://www.anaconda.com/), which is the best environment for machine learning codes and the sklearn library. If you use just Jupyter or spyder, you have to install the appropriate libraries.

Duration: 2 hours (+ 2 hours homework)

--- Preliminaries ---

Regression is a supervised learning problem where there is an input x an output y and the task is to learn the mapping from the input to the output. We have also seen that the approach in machine learning is that we assume a model, that is, a relation between x and y containing a set of parameters, say, _ in the following form:

$$y = g(x; \theta)$$

g(x; θ) is the regression function. The machine learning program optimizes the parameters θ such that the approximation error (called cost function) is minimized, that is, our estimates are as close as possible to the correct values given in the training set.

Several methods are proposed to optimize cost function such as variance, covariance, …etc. the best optimization method is called gradient descent.

Logistic regression is used when the dependent variable is binary (0/1, True/False, Yes/No) in nature. Even though the output is a binary variable, what is being sought is a probability function which may take any value from 0 to 1.

--- Examples ---

- Let see this simple example about linear regression with one variable as we called usually simple regression, we need to predict the house price using regression. The used dataset involves one features, one Target (price) and downloaded from Kaggle : https://www.kaggle.com/datasets/harlfoxem/housesalesprediction.

First, we import the useful libraries.

```
import numpy as np
import pandas as pd
import seaborn as sb
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn import linear_model
from sklearn.metrics import classification_report
%matplotlib inline
```

Read the dataset, we use here the kc_house_data.csv downloaded from : https://www.kaggle.com/datasets/harlfoxem/housesalesprediction

```
df = pd.read_csv("kc_house_data.csv")
df.head(15)
```

Now, we explore our data set to find missed values. We Determine the features that affect (theoretically) house prices.

```
df.info()
df.isnull().sum()
df = df.drop(['id','date', 'lat', 'long','zipcode'], axis =1)
df.head()
plt.figure(figsize=(48, 6))
sb.stripplot(x="yr_built", y="bedrooms", data=df);
plt.figure(figsize=(20, 8))
sb.set_context("notebook", font_scale=1.5, rc={"lines.linewidth": 2.5})
sb.stripplot(x="bedrooms", y="price", data=df);
plt.figure(figsize=(48, 8))
sb.barplot(x="bedrooms", y="price", hue="grade", data=df);
sb.countplot(x='bedrooms',data=df, palette='hls')
```

Next, we try to build our model using linear regression, we split our data set to 70% for training and 30% for testing.

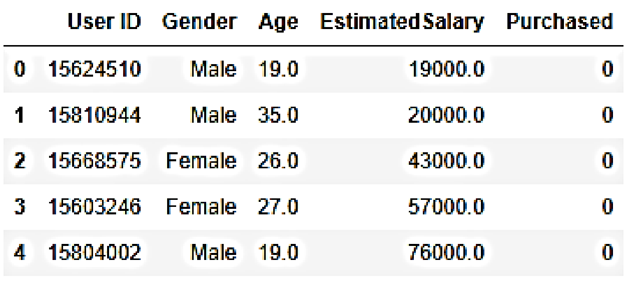
```
columns = ['bedrooms', 'bathrooms', 'sqft_living', 'sqft_lot', 'floors', 'waterfront', 'view', 'condition',
'grade','sqft_above', 'sqft_basement', 'yr_built','yr_renovated', 'sqft_living15', 'sqft_lot15']
labels = df['price'].values
features = df[list(columns)].values
X_train, X_test, y_train, y_test = train_test_split(features, labels, test_size=0.30)
regr = linear_model.LinearRegression()
regr.fit(X_train, y_train)
```

We evaluate our model.

```
Accuracy = regr.score(X_train, y_train)
print ("Accuracy in the training data: ", Accuracy*100, "%")
accuracy = regr.score(X_test, y_test)
print ("Accuracy in the test data", accuracy*100, "%")
```

- Now, let us see another example about logistic regression. We use a dataset provided by the teacher to predict id a client purchased an object or no, that means we study the influence of some features on the result.

Here an excerpt about the dataset :



	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19.0	19000.0	0
1	15810944	Male	35.0	20000.0	0
2	15668575	Female	26.0	43000.0	0
3	15603246	Female	27.0	57000.0	0
4	15804002	Male	19.0	76000.0	0

```
# import libraries
%matplotlib
import numpy as np
import matplotlib.pyplot as plt
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
from matplotlib.colors import ListedColormap
from mpl_toolkits.mplot3d import Axes3D
import seaborn as sns

# Import the dataset
dataset = pd.read_csv('clients.csv')

# Data visualisation
dataset.head()

# Dataset visualisation
plt.scatter(dataset.EstimatedSalary, dataset.Purchased)

# Delete User ID
dataset.drop(['User ID'],axis='columns',inplace=True)

# Data visualisation
dataset.head()

# Transformation of gender variable
dataset.Gender = dataset.Gender.map({'Male': 1, 'Female': 2})
dataset.head()
ax = plt.axes(projection='3d')
ax.scatter(dataset.Gender,dataset.Age,dataset.EstimatedSalary, c=dataset.Purchased)

# Achat percent
count_sub = len(dataset[dataset['Purchased']==1])
count_no_sub = len(dataset[dataset['Purchased']==0])
pct_of_no_sub = count_no_sub/(count_no_sub+count_sub)
print("Pourcentage absence d'achat", pct_of_no_sub*100)

# genre Influence on achat
table= pd.crosstab(dataset.Gender,dataset.Purchased)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Genre / Achat')
plt.xlabel('Genre')
plt.ylabel('Pourcentage de client')
dataset.drop(['Gender'],axis='columns',inplace=True)
dataset.head()
table= pd.crosstab(dataset.Age,dataset.Purchased)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Age / Achat')
plt.xlabel('Age')
plt.ylabel('Pourcentage de client')
plt.savefig('Age-Achat')
table= pd.crosstab(dataset.EstimatedSalary,dataset.Purchased)
table.div(table.sum(1).astype(float), axis=0).plot(kind='bar', stacked=True)
plt.title('Salaire / Achat')
plt.xlabel('Salaire')
plt.ylabel('Pourcentage de client')

# define Y dependent variable and X independent variable
X = dataset.iloc[:, [0, 1]].values
y = dataset.iloc[:, -1].values

# points Visualisation
plt.scatter(X[:,0],X[:,1], c=y)

# Split the dataset
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.25, random_state = 0)

# Feature Scaling
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
X_test

# Build the model
classifier = LogisticRegression(random_state = 0, solver='liblinear')
classifier.fit(X_train, y_train)
# Faire de nouvelles prédictions
y_pred = classifier.predict(X_test)
classifier.score(X_test,y_test)

# confusion matrix
cm = confusion_matrix(y_test, y_pred)
print(cm)
#print(classification_report(y_test, y_pred))

# Results Visualisation
X_set, y_set = X_train, y_train
X1, X2 = np.meshgrid(np.arange(start = X_set[:, 0].min() - 1, stop = X_set[:, 0].max() + 1, step = 0.01),
                     np.arange(start = X_set[:, 1].min() - 1, stop = X_set[:, 1].max() + 1, step = 0.01))
plt.contourf(X1, X2, classifier.predict(np.array([X1.ravel(), X2.ravel()]).T).reshape(X1.shape),
             alpha = 0.4, cmap = ListedColormap(('red', 'green')))
plt.xlim(X1.min(), X1.max())
plt.ylim(X2.min(), X2.max())
for i, j in enumerate(np.unique(y_set)):
    plt.scatter(X_set[y_set == j, 0], X_set[y_set == j, 1],
                c = ListedColormap(('red', 'green'))(i), label = j)
plt.title('Résultats du Training set')
plt.xlabel('Age')
plt.ylabel('Salaire Estimé')
plt.legend()
plt.show()
x_predict = sc.transform([[35,15000]])
classifier.predict(x_predict)
```

-- **Activities** --

Activity 1

The aim of this activity is to predict the relationship between the price of a pizza and its size using the linear regression. As previously specified, and as we have chosen a linear regression, this means that we are making the hypothesis that there is a linear relationship between the price of a pizza and its size.

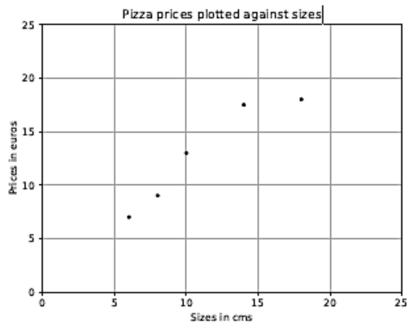
- Dataset : We will first generate a set of points for the training set:

Size(Feature)	Price(Target)
6	7
8	9
10	13
14	17.5
18	18

We then need to declare a numpy array containing the data (pizza sizes) and declare an array containing the corresponding prices , for example :

```
X = np.array([[6], [8], [10], [14], [18]])
y = [7, 9, 13, 17.5, 18]
```

- Now display the data in a graph to see if there is a relationship between the size and price of a pizza. To do this, we'll use matplotlib. We can see that there is indeed a relationship: the price of a pizza increases with its size, which is consistent with our expertise on the subject.



We are going to use the sklearn library and in particular the linear regression model:

```
from sklearn.linear_model import LinearRegression
```

- Now use the functions fit for training and predict to predict the model's response to an example.
- We now need to evaluate our model. To do this we need to define a loss function (also called a cost function). The difference between the actual price of the pizzas and the price predicted by our model is called the residual error or learning error. When we evaluate our model on an independent test basis, the resulting error is called the prediction error or test error.

We can create the best possible model by minimizing the sum of the residual error. This error is called the residual sum of squares (RSS).

The cost function associated with this error is defined as follows : $RSS = \sum_{i=1}^{n}(y_i - f(x_i))^2,$

With Yi the observed value (real value) and f(xi) the predicted value.

Calculate the error obtained in python (Residual sum of squares : 8.75)

Activity 2

The second activity involves applying simple linear regression on the dataset below using the least-squares method. The student would build his own model using Python without using the Python machine learning instructions.

Size (x)	Price (y)
2104	4.5
1416	3.5
1534	3.2

852	1.6

Our objective is to find the equation y= ax+b that fit with the cloud points.

- Calculate a and b?
- Calculate the correlation indicator?
- Indicate whether the equation found is accurate ?

Do the same work using the gradient descent. Compare your prediction results with the python machine learning code.

Activity 3

In the first part of this exercise, we'll build a logistic regression model to predict whether a student gets admitted to a university. Suppose that you are the administrator of a university department, and you want to determine each applicant's chance of admission based on their results on two exams. You have historical data from previous applicants that you can use as a training set for logistic regression. For each training example, you have the applicant's scores on two exams and the admissions decision. To accomplish this, we're going to build a classification model that estimates the probability of admission based on the exam scores.