Supervised Learning:
Introduction to Supervised Learning

AAA-Python Edition

Plan

- 1- Supervised Learning
- 2- Classification
- 3- Regression
- 4- Features
- 5- Linear Regression
- 6- Polynomial Regression

Machine Learning

Machine learning

Build

Mathematical Model   with   **Tunable parameters**

Fit

Observed Data: described by a **set of** features

Predict

New Data

[By Amina Delali]

Supervised Learning

Supervised learning

Build

| Mathematical Model | with | Tunable parameters |

Fit

Labeled Observed Data

Fitting == **minimize** the **cost** function
(**difference** between **True** and **Predicted** labels)

Predict the Labels

Predicting == **Apply** the **obtained mathematical** model on the **New Data**

[By Amina Delali]

1- Supervised Learning

Unsupervised Learning

Unsupervised learning

Build

Mathematical Model   with   Tunable parameters

Fit

Not Labeled Observed Data

Predict the Labels

Predicting = **Extract** information about **New Data**

[By Amina Delali]

Classification

Supervised learning: Classification

Build

Mathematical Model   with   Tunable parameters

Fit

Labeled Observed Data: The Labels are **categories**

Predict

Predict the **known categories** for the New Data

6

[By Amina Delali]

- Scikit-learn is a python library used for data mining and data analysis

- It is build on :"Numpy", "Scipy", and "Matplotlib" libraries.

- It will be used for both classification and regression by importing the corresponding modules.

- It will also be used for feature extraction using :"sklearn.feature_extraction" module

- The full documentation about scikit-learn, is available at its homepage: **http://scikit-learn.org/stable/index.html**

2- Classification

school of a

[By Amina Delali]

Classification Problems

Classifying images

Fit

Observed Data: **x-ray** images. Labels == **yes** or **no.**
Yes == Presence of Tumor, No == Absence of Tumor

Predict if there is a tumor or not

New Data: Not labeled x-ray images.

Classifying text

Observed Data: **words** in sentences. Labels == **verb**, **subject**, or **adjective**

Predict if the category of the word

New Data: Not labeled words

[By Amina Delali]

3- Regression

school of a

Regression

Supervised learning: Regression

Build

Mathematical Model with **Tunable parameters**

Fit

Labeled Observed Data: The Labels are **continuous** quantities

Predict

Predict the **New quantities** for the New Data

[By Amina Delali]

Regression Problems

House pricing

Fit

Observed Data: **houses** described by a set of **characteristics**.
Labels == **prices** of the houses

Predict houses price

New Data: a set of houses described by the same set of characteristics (with different values), but they don't have the price indicated.

Photometric redshift

Observed Data: **galaxies** described by their **brightness** at several **wavelengths**. Labels == **distances** of the galaxies

Predict the distance

New galaxies without the distance information

10

[By Amina Delali]

Introduction

Machine Learning

Fit

Observed Data: the data must be described by a set of **characteristics: numerical values**

Predict if there is a tumor or not

The new Data: must be described by the same set but with different values

These characteristics, aren't always in a numerical format. They can be:

Categories

Text

Images

They have to be transformed in a numerical format

4- Features

school of a

Categorical features

Categories → **One-hot-encoding**

The Feature 3 is transformed into 3 other features each one representing the presence of a category belonging to feature 3

```python
from pandas import DataFrame as DF, Series as S
```

```python
# we assume that the features are contained in dataframe dataF
# the feature: "Feat3" is a categorical feature (not numeric)

dataF = DF([{"Feat1": 545,"Feat2":3, "Feat3": "Cat1" },{"Feat1": 362,"Feat2":2, "Feat3": "Cat3" }
  {"Feat1": 1005,"Feat2":5, "Feat3": "Cat2" }],index=["house1","house2","house3"])
dataF
```

```python
# The DictVectorizer from sklearn will be used for the one-hot-encoding
from sklearn.feature_extraction import DictVectorizer
# convert the dataframe to a dictionary values
myDict = dataF.to_dict('records')
# create an instance of a DictVectorizer
vec = DictVectorizer(sparse=False, dtype=int)
# transform the category data
trDict=vec.fit_transform(myDict)
# convert the dictionary to a dataframe (just for visualization purpose)
DF.from_dict(trDict)
```

myDict

```
[{'Feat1': 545, 'Feat2': 3, 'Feat3': 'Cat1'},
 {'Feat1': 362, 'Feat2': 2, 'Feat3': 'Cat3'},
 {'Feat1': 1005, 'Feat2': 5, 'Feat3': 'Cat2'}]
```

	Feat1	Feat2	Feat3
house1	545	3	Cat1
house2	362	2	Cat3
house3	1005	5	Cat2

	0	1	2	3	4
house1	545	3	1	0	0
house2	362	2	0	0	1
house3	1005	5	0	1	0

[By Amina Delali]

Word count

Text

TF – IDF:
term frequency–inverse document frequency

Word count

```
# a multilines string
text= '''this is a simple text
we will count and we will see this'''
# converting each line to a list element
textT = text.split("\n")
# import CountVectorizer for word count encoding
from sklearn.feature_extraction.text import CountVectorizer
# create an instance of CountVectorizer
vec = CountVectorizer()
# create the encoding
wc = vec.fit_transform(textT)
# create the corresponding dataframe
DF(wc.toarray(),columns=vec.get_feature_names())
```

Each line will represent a row. And each word will represent a feature. Features values in each row will be the count of the corresponding word in the corresponding line

```
['this is a simple text', 'we will count and we will see this']
```

	and	count	is	see	simple	text	this	we	will
0	0	0	1	0	1	1	1	0	0
1	1	1	0	1	0	0	1	2	2

Appears **0** times in line **0**
Appears **2** times in line **1**

4- Features

13

[By Amina Delali]

Text features

Word count

Text

TF – IDF:
term frequency–inverse document frequency

TF – IDF

```python
# import TfidfVectorizer for  frequency-inverse document frequency encoding
from sklearn.feature_extraction.text import TfidfVectorizer
# create an instance of CountVectorizer
vec = TfidfVectorizer(norm=None)
# create the encoding
wc = vec.fit_transform(textT)
# create the corresponding dataframe
DF(wc.toarray(), columns=vec.get_feature_names())
```

We didn't normalize the results

$$idf(t)=\ln\left(\frac{1+n_d}{1+df(d,t)}\right)+1$$

$$tf-idf(t,d)=tf(t,d)*idf(t)$$

$n_d=total\ number\ of\ documents$

$tf(t,d)=number\ of\ \times the\ term\ t\ occurs\in document\ t$

$df(d,t)=number\ of\ documents\ that\ contain\ the\ term\ t$

== 1 *
(ln[(1+2)/1+1]+1)
== 1.405465

['this is a simple text', 'we will count and we will see this']

	and	count	is	see	simple	text	this	we	will
0	0.000000	0.000000	1.405465	0.000000	1.405465	1.405465	1.0	0.00000	0.00000
1	1.405465	1.405465	0.000000	1.405465	0.000000	0.000000	1.0	2.81093	2.81093

Use pixel values as features values:
number of features ==
number of pixels

Image

```python
# importing digits dataset sample
from sklearn.datasets import load_digits
# loding digits dataset sample
digits = load_digits()
# the dataset has pixel values for 1797 images,
# each image is coded in 8*8 pixel values
# indicating grayscale values
print(digits.images.shape)
# for example, the 35th image represent a "6"
digits.images[34]
```

```python
1  # ploting the 35th image ( the digit 6)
2  %matplotlib inline
3
4  import matplotlib.pyplot as plt
5  plt.imshow(digits.images[34])
```

`digits.data[34]`

```
(1797, 8, 8)
array([[ 0.,   0.,   0.,   5.,  14.,   2.,   0.,   0.],
       [ 0.,   0.,   1.,  13.,  11.,   0.,   0.,   0.],
       [ 0.,   0.,   5.,  16.,   2.,   0.,   0.,   0.],
       [ 0.,   0.,   6.,  15.,   5.,   0.,   0.,   0.],
       [ 0.,   1.,  15.,  16.,  15.,  11.,   1.,   0.],
       [ 0.,   2.,  13.,  14.,   1.,  12.,   9.,   0.],
       [ 0.,   0.,   4.,  16.,   7.,  13.,   9.,   0.],
       [ 0.,   0.,   0.,   5.,  16.,  15.,   3.,   0.]]])
```
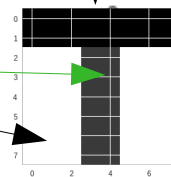
Is just a reshape of images
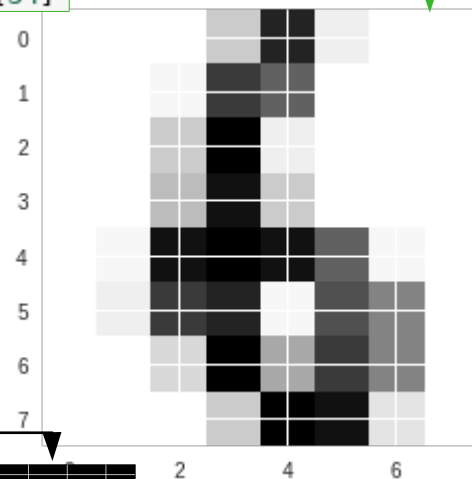
```python
4  myList = ([16]*8)*2+ ([0]*3+[13]*2+[0]*3)* 6
5  myT= np.array(myList).reshape(8,8)
6  plt.imshow(myT)
```

[By Amina Delali]

4- Features

school of a

15

5- Linear Regression

school of a

Simple Linear Regression : 1 feature

Build

Mathematical Model== a Line, modelized by:
$Y = a x + b$

with

a: the slope
b: the intercept

Fit

Labeled Observed Data: described by 1 feature: **x**
The labels are **y** values. Fitting == find **a** and **b** that minimize the difference between the real labels: y and the estimated ones.

Predict

Predict the **New y** for the New not labeled x values

16

[By Amina Delali]

Example

5- Linear Regression

school of a

```python
1  # importing form linear_model module, LinearRegression
2  from sklearn.linear_model import LinearRegression
3  # create an instance of the model
4  model = LinearRegression()
5  # fiting the model using x and y values
6  model.fit(np.array(x).reshape(-1,1), np.array(y).reshape(-1,1))
7  # values of a (coef) and b(intercept) after fitting
8  print("a == the value in  ", model.coef_)
9  print("b == the value in  ", model.intercept_)
10 # new values of x : we create this values to generate the
11 # line representing the model
12 newX = np.linspace(1,20,1000)
13 # predict the labels for newX
14 newy = model.predict(np.array(newX).reshape(-1,1))
15 # plotting the old and the new values
16 plt.scatter(x, y)
17 plt.plot(newX, newy, color="g");
```

The model to train
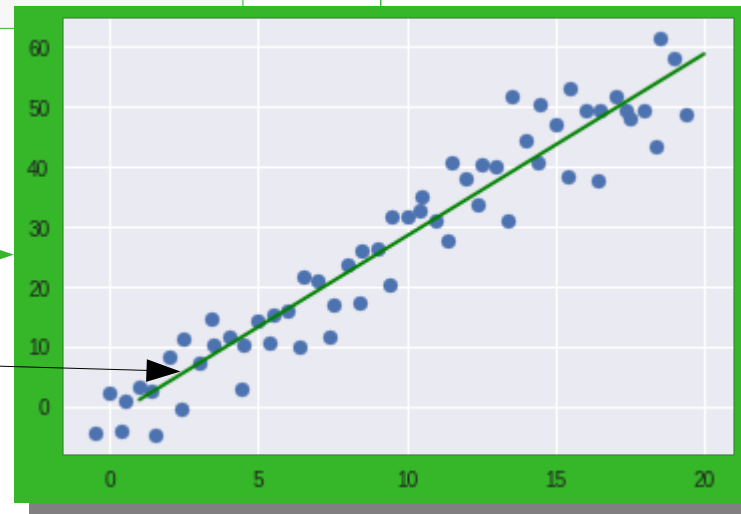
Initialization of a linear regressor instance

Train the model

Parameters of the model

Predict y for new values: actually this new x values are used to draw the model

```
a == the value in    [[3.029415]]
b == the value in    [-1.97286737]
```

The line representing the model

[By Amina Delali]

6-Polynomial Regression

Build

Mathematical Model== a Linear curve, modelized by:

$Y = a_1x + a_2x^2 + ... + a_nx^n + b$

with

The parameters: a_1, a_2, …., a_n, b
N: the degree of the polynomial model

Fit

Labeled Observed Data: described by 1 feature: **x**
From that **x**, new **polynomial features** are **generated:** $x^2, x^3, ..., x^n$.

Predict

The new x values must be **transformed** first into polynomial feature, before applying the model.

[By Amina Delali]

Example

```python
1  from sklearn.preprocessing import PolynomialFeatures
2  from sklearn.pipeline import make_pipeline
3  xArr= np.array(x).reshape(-1,1)
4  y2Arr= np.array(y2).reshape(-1,1)
5  # the order of the polynomial features
6  order =11
```

The model is the same, we will just generate new features: **polynomial features**

```python
polyObj= PolynomialFeatures(order,include_bias=False)
myNewFeatures=polyObj.fit_transform(xArr)
# instanciate a linear regrossor model
myModel = LinearRegression()
# fitinig the model
myModel.fit(myNewFeatures, y2Arr)
# the calculated parameters
for i in range(1,order+1):
    print("a"+ str(i)+" == "+ str(myModel.coef_[0,i-1]))
print ("b == ",myModel.intercept_)

# new values of x : to visualize the model
newX = np.linspace(1,20,1000)
newXArr= np.array(newX).reshape(-1,1)
# corresponding polynomial features for the new values
newPlolyX =polyObj.fit_transform(newXArr)
# predict the labels for newX
newy2 = myModel.predict(newPlolyX)
# plotting the old values and the visualization of the model
plt.scatter(x, y2)
plt.plot(newX, newy2, color="g");
```

```
a1  == -0.057443431666221394
a2  == -0.10775669970224205
a3  == -0.15702923169271918
a4  == -0.09393489084681118
a5  == 0.12136012879617761
a6  == -0.04165097865144431
a7  == 0.0072554716298266568
a8  == -0.0007421930578640621
a9  == 4.655613420523336e-05
a10 == -1.7661168388830744e-06
a11 == 3.727072246228341e-08
a12 == -3.3641874008782935e-10
b  ==  [0.81724983]
```
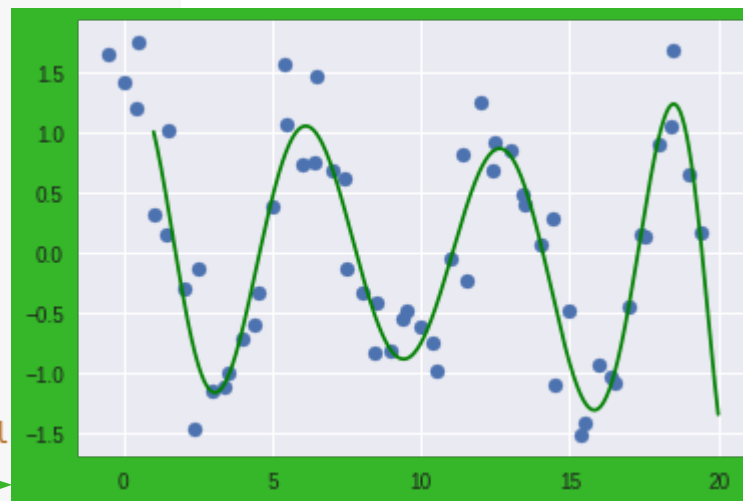
[By Amina Delali]

Example with a pipeline

6-Polynomial Regression

```python
from sklearn.pipeline import make_pipeline
# create the pipeline
myPipeline= make_pipeline(polyObj,myModel)
myPipeline.fit(xArr, y2Arr)
# the calculated parameters
for i in range(1,order+1):
    print("a"+ str(i)+" == "+ str(myModel.coef_[0,i-1]))
print ("b == ",myModel.intercept_)

# predict the labels for newX
newy2 = myPipeline.predict(newXArr)
# plotting the old values and the model visualization
plt.scatter(x, y2)
plt.plot(newX, newy2, color="g");
```

This pipeline will:
1- generate polynomial features from the data.
2- apply the regression model to the new data

The newX values are generated in a way that the model can be plotted in a linear form :
it describes the model
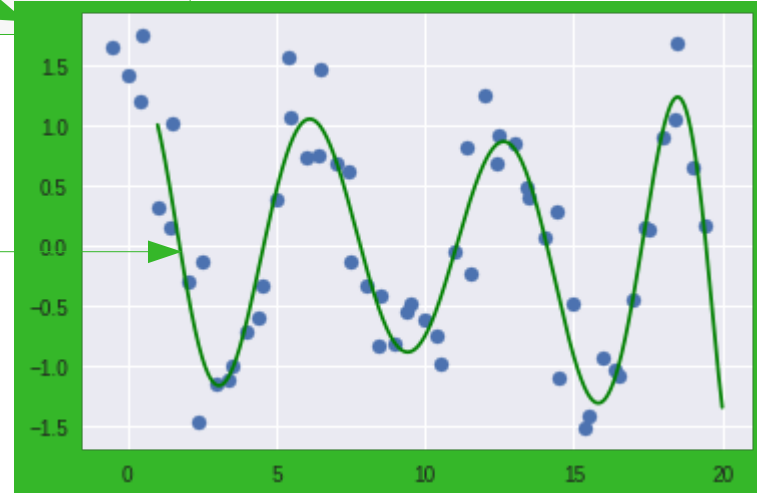
[By Amina Delali]

References

- Joshi Prateek. Artificial intelligence with Python. Packt Publishing, 2017.

- Jake VanderPlas. Python data science handbook: essential tools for working with data. O'Reilly Media, Inc, 2017.

Thank you!

FOR ALL YOUR TIME