Machine Learning: Introduction

Elsa Ferreira Gomes

2023/2024

Machine Learning

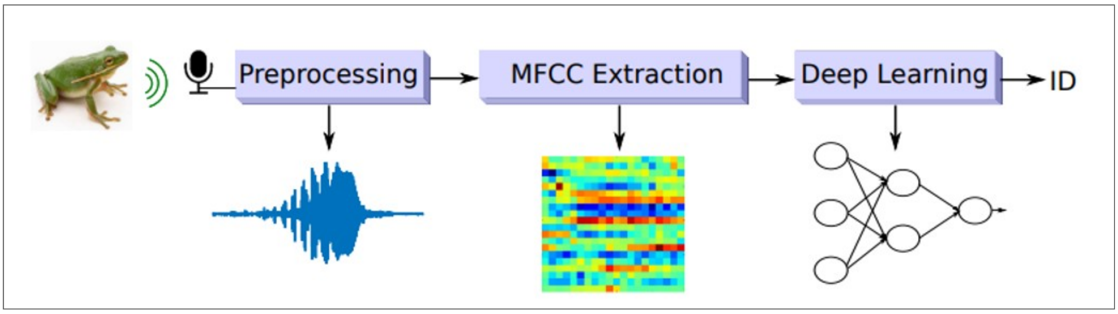
Problem examples

Classification using audio data: heart sounds and other signals

Machine Learning

Problem examples

Classification using audio data: heart sounds and other signals



Machine Learning

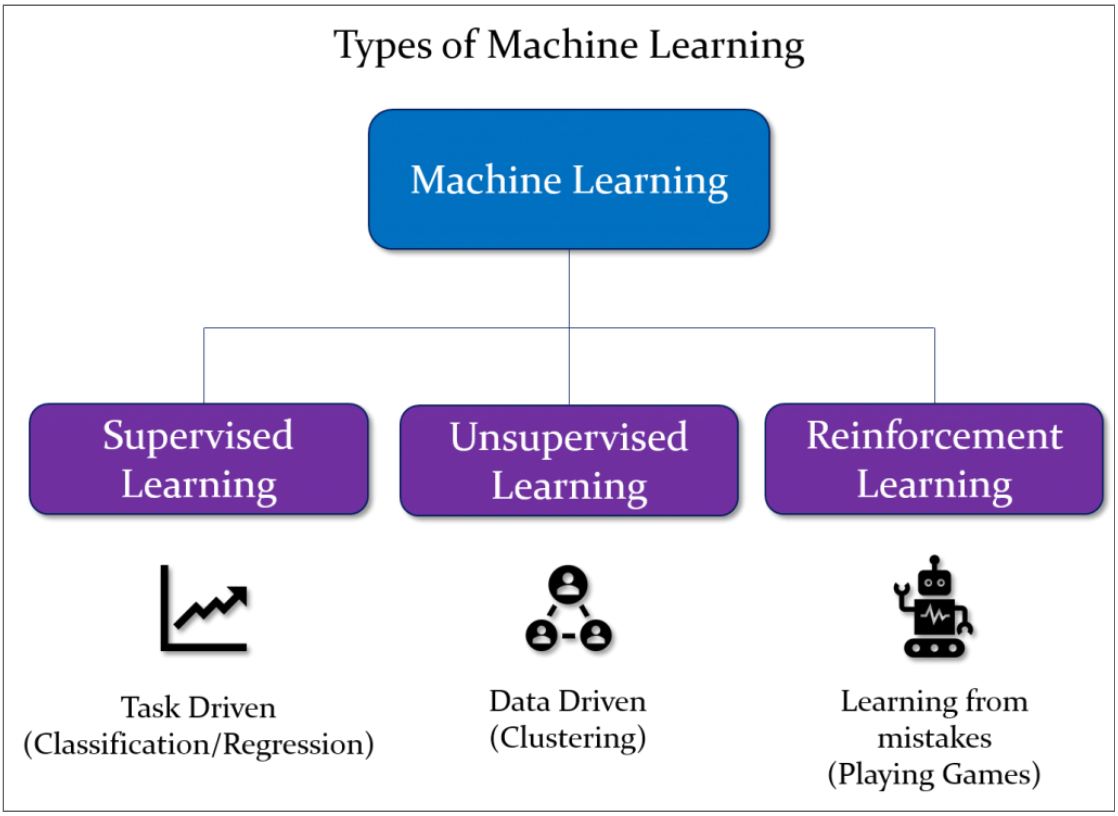
Problem examples:

- Spotify Popularity Prediction
- Predict House Prices
- Natural Language Processing with Disaster Tweets
- Email spam detection system
- Identify the risk factors for prostate cancer
- Identify numbers in a handwritten zip code
- Face recognition
- .....

Machine Learning

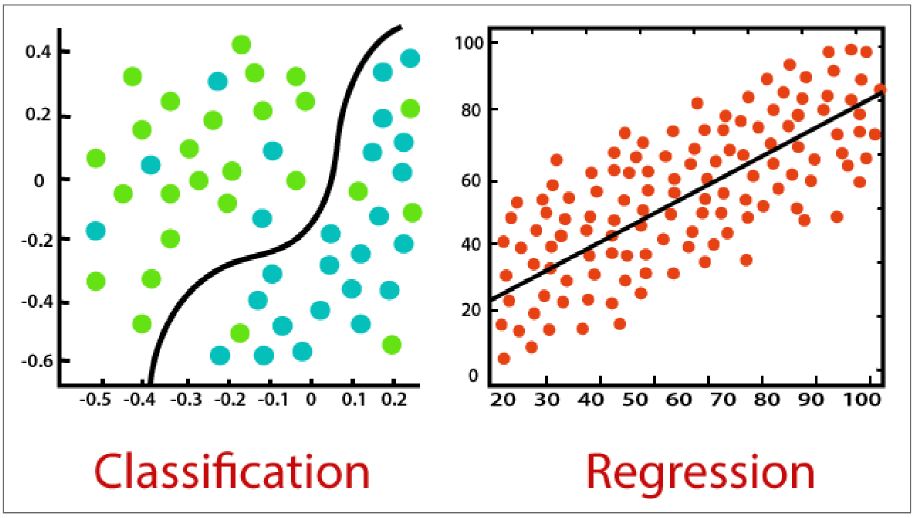
"A computer program is said to learn from **experience E** with respect to some class of **tasks T** and **performance measure P** if its performance at tasks in *T*, as measured by *P*, improves with experience *E*."
Tom Mitchell

Machine Learning



Supervised Learning Problem

- Outcome measurement Y (also called dependent variable, response, target).
- Vector of p predictor measurements X (or inputs, regressors, covariates, features, independent variables).
- In the **regression problem**, Y is quantitative (e.g price, blood pressure).
- In the **classification problem**, Y takes values in a finite, unordered set (e.g survived/died, digit 0-9, cancer class of tissue sample).
- The training data $(x_1, y_1), \ldots, (x_N, y_N)$ are observations (examples, instances) of these measurements.



Unsupervised Learning Problem

- No outcome variable
    - just a set of predictors (features) measured on a set of samples.
- Objective is more fuzzy
    - find groups of samples that behave similarly, find features that behave similarly, find linear combinations of features with the most variation.
- Difficult to know how well your are doing.
- Different from supervised learning,
    - but can be useful as a pre-processing step for supervised learning.

Supervised Learning Problem

Objectives

On the basis of the training data we would like to:

- Accurately predict **unseen test cases**.
- Understand which inputs (and how) affect the outcome.
- Assess the quality of our predictions and inferences.

Supervised Learning Problem

The aim

Our goal is to find a **useful approximation** $\hat{f}(x)$ of $f(x)$

- $f(x)$ is the function that generates the phenomenon it is **unknown**
- What is the best **approximation**?
- How do we measure the **goodness** of an approximation?
- How do we **find the best approximation** of $f(x)$ (or at least a very good one)?

Machine Learning

Example: Predicting Used Car Prices

- We have observations (experience)
- We assume there is a function that outputs the position given the (car) age

$$price = f(car.age)$$

In [1]:
```python
import matplotlib.pyplot as plt
        import numpy as np
from sklearn.linear_model import LinearRegression
import pandas as pd

# create a dataframe from scratch using a dictionary
used_cars = pd.DataFrame({
    'car_age': [4, 4, 5,  5,  7,  7, 8, 9, 10, 11, 12],
    'price': [6300, 5800, 5700, 4500, 4500, 4200, 4100, 3100, 2100, 2500, 2200]
})

used_cars
```

Out[1]:

	car_age	price
0	4	6300
1	4	5800
2	5	5700
3	5	4500
4	7	4500
5	7	4200
6	8	4100
7	9	3100
8	10	2100
9	11	2500
10	12	2200

Python libraries

Pandas

Pandas is used to analyze data.

Numpy

NumPy is the fundamental package for scientific computing in Python.used for working with arrays (domain of linear algebra, fourier transform, and matrices)

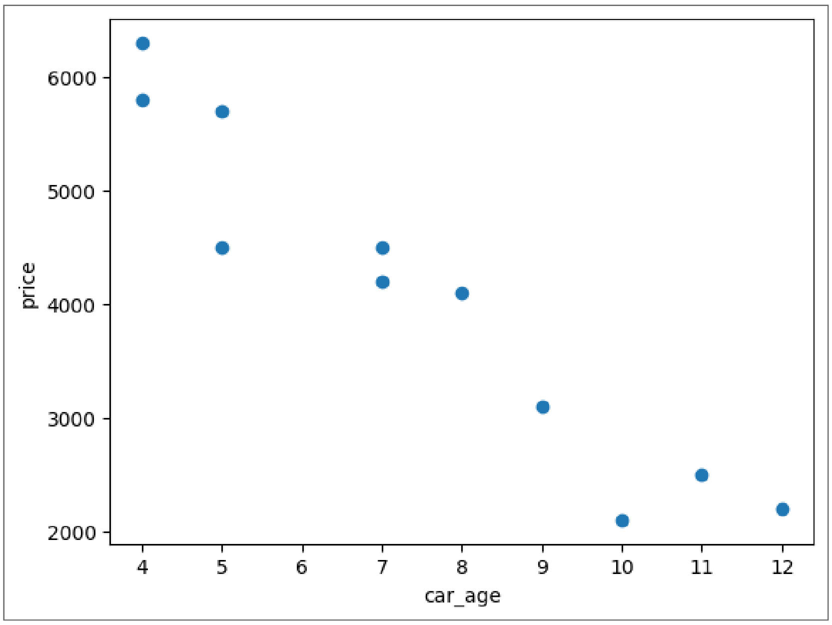
Sklearn

Machine Learning in Python (is simple and efficient tools for predictive data analysis)

Matplotlib

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python

```
In [2]: ax=plt.axes()
        ax.scatter(used_cars.car_age,used_cars.price)
ax.set(xlabel='car_age')
ax.set(ylabel='price'); # this semicolon supresses the output of this command. Try withou
```



Machine Learning

The goal

- Find a **useful approximation** $\hat{f}(x)$ of $f(x)$
- We need to know how we:
  - **find** the best **approximation**
  - **measure** the **goodness**
  - **find the best approximation**

Linear model

Find a linear function

- We can use **linear functions**

$$\hat{y} = \hat{f}(x) = \beta_0 + \beta_1 x$$

- Now we have to look for one linear function that suits us
  - All we have, all we know, is our **data**
  - There are **many** different ways to do that
  - **One** is the **least squares** method

We can use a linear model to aproximate the function

- Linear Regression

```
In [3]: x=np.array(used_cars.car_age)
        X=x[:, np.newaxis] # newaxis gives X the shape of a matrix. Try without
y=np.array(used_cars.price)

model = LinearRegression(fit_intercept=True)
model.fit(X,y)

# Pretty print model
beta1=model.coef_[0]
beta0=model.intercept_

print('price =',beta0,'+',beta1,'*car_age')
```

```
price = 7836.258660508083 + -502.4249422632795 *car_age
```

Approximating the function

- Here, the approximated function $\hat{f}(x)$ is:

$$price = \hat{f}(car.age) = 7836.25 - 502.42 \times car.age$$

Hint: The coefficient estimates for Ordinary Least Squares rely on the independence of the features.

How does the model approximate the true function

Calculate the residuals

In [4]:
```
ypred=model.predict(X)
residuals=y-ypred
residuals
```

Out[4]:

```
array([ 473.44110855,  -26.55889145,  375.86605081, -824.13394919,
        180.71593533, -119.28406467,  283.1408776 , -214.43418014,
       -712.00923788,  190.41570439,  392.84064665])
```

In [5]:
```
# Residual Sum of Squares

sum(residuals**2)
```

Out[5]:

```
1915900.692840648
```

```python
In [6]: quartiles = np.percentile(residuals, [25, 50, 75])
        maxr=max(residuals)
minr=min(residuals)
# Printing with format (%). We define number of digits to print in floats (.3f)
print('Residuals summary:')
print('min = %.3f' % minr)
for i in range(3):
    print('Q%i = %.3f' % (i+1, quartiles[i]))
print('max = %.3f' % maxr)
```

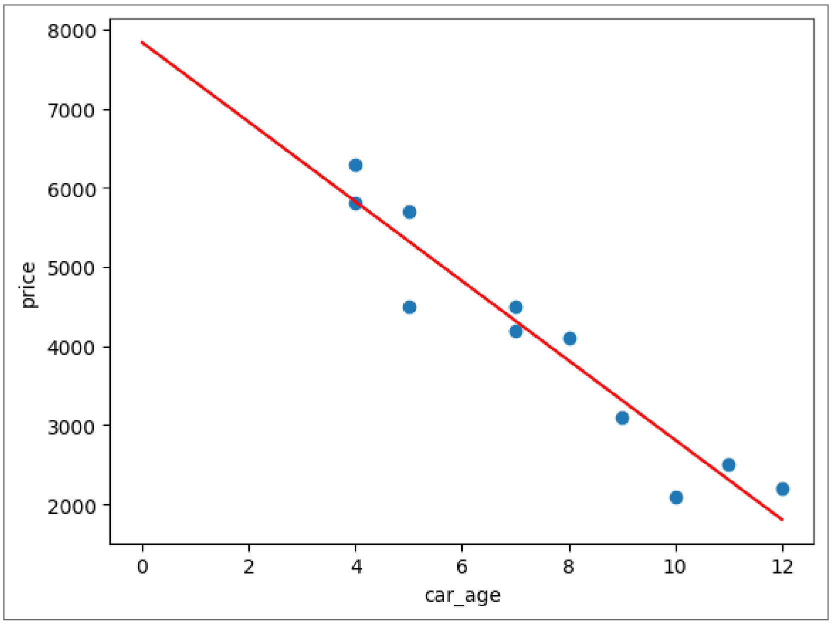
```
Residuals summary:
min = -824.134
Q1 = -166.859
Q2 = 180.716
Q3 = 329.503
max = 473.441
```

In [7]:
```
# plot the regression line against the points

xfit = np.array([0,max(x)])

# newaxis gives xfit the shape of a matrix
yfit = model.predict(xfit[:,np.newaxis])

ax=plt.axes()
ax.scatter(x,y)
ax.set(xlabel='car_age')
ax.set(ylabel='price');
ax.plot(xfit, yfit,color='red');
```



In [8]:
```
# Did we find a useful approximation?
# Let's try some test cases

xtest=np.array([0.1,1,5,10,100])
Xtest=xtest[:,np.newaxis] # Xtest is a matrix now
ypred = model.predict(Xtest)
ypred
```

Out[8]:

```
array([  7786.01616628,   7333.83371824,   5324.13394919,   2812.0092378
8,
       -42406.23556582])
```

In [9]:
```
# The output is a float, so we should round to an integer
list(map(lambda predpos: int(np.round(predpos,0)),ypred))
```

Out[9]:

```
[7786, 7334, 5324, 2812, -42406]
```

In [10]:
```
# How good is the model on average?
# We will measure R2 of the model

print('R2 of the linear model is %.3f' % model.score(X,y))
```

```
R2 of the linear model is 0.912
```