Python for Data Science

Machine Learning 1

ENSAE-ENSAI
Formation continue
(Cepe)

What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

Linear Regression

Regularization

Validation

What is Machine Learning?

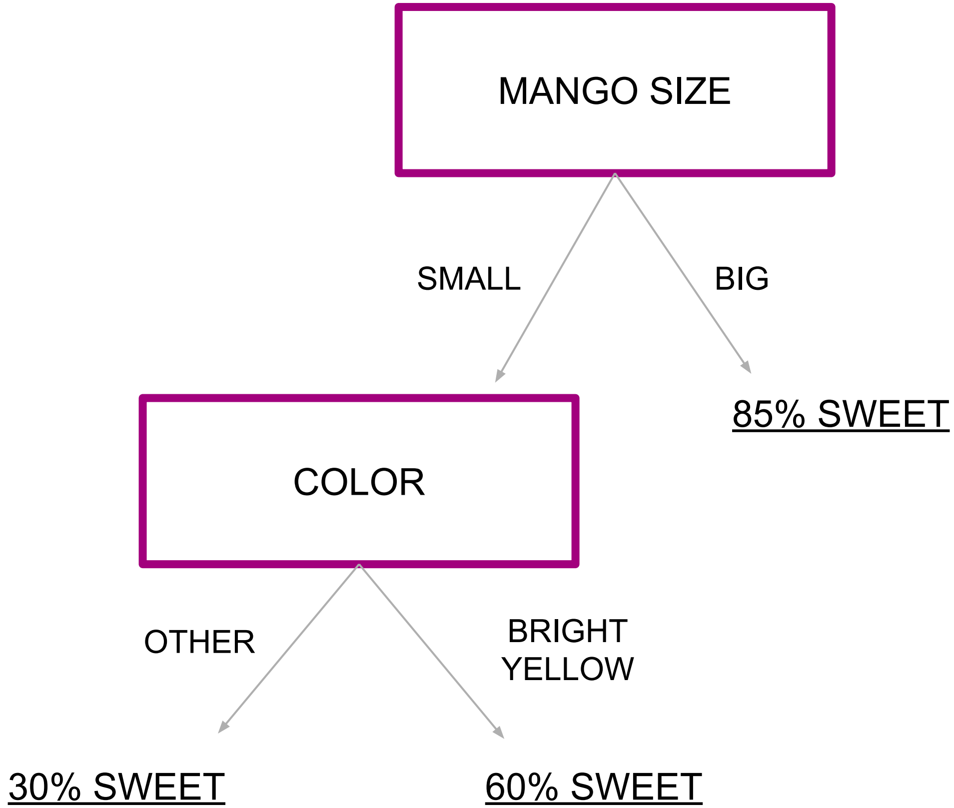
What is Machine Learning?

Mango Science

What is Machine Learning?

- ❑ We want to buy sweet mangoes (**target**)
- ❑ Your grandma always said that the bright yellows are the sweetest (**business rule**)
- ❑ You realize that only 60% of bright yellow mangoes you bought are sweet (**performance**)
- ❑ You learn that mangoes vary in size, you pick both small and big mangoes of all available colors (**sampling**)
- ❑ You observe that out of all the big mangoes, 85% were sweet. Based on your finding you create the following rule (**rule creation**)

What is Machine Learning?



What is Machine Learning?

- Your vendor has retired, you go to a new vendor and you find the big bright yellow mangoes to be a bit disappointing (**overfitting**)

- You decide to repeat your experience and come to a conclusion that the small red ones are the sweetest (**more learning**)

- Your best friend doesn’t care about sweet mangoes, he likes them juicy (**more targets**)

- Your get married, your spouse doesn’t like mangoes but she loves apples, she wants you to use all your knowledge about mangoes to pick the sweetest apples (**scoping**)

What is Machine Learning?

- ❑ You decide to do a PhD in Mango Science
- ❑ You learn that there are 400 different kinds of mangoes although you can buy in your country only 40 different kinds (**generalization**)
- ❑ You pick mangoes from different markets randomly (**training data**)
- ❑ You create a table to represent the basic data regarding the mangoes: color, size, country, shape, vendor (**features**)
- ❑ You notice that using some variables could give you more useful information: date (season, day of purchase), packaging, weather conditions, market type (**feature engineering**)
- ❑ You rate each mango by sweetness, juicyness, ripeness, sourness, … (**targets**)

What is Machine Learning?

- ❑ You use Python (or R or any other package) to build a classification model to find correlation between features and the output variables (**modelling**)
- ❑ Every time you go to the market you see how good is your prediction (**test**)
- ❑ You train a decision tree with scikit-learn and realize that if you have <u>too many rules</u> your model doesn’t work so well on new mangoes (**overfitting**).

<u>You need to remember that you can’t taste all mangoes on earth so your model must **generalize** for kinds that you never tried</u>

But wait …

Isn’t this just Statistics?

But wait ... Isn't this just Statistics?

- ❖ My answer: Yes and No

In theory:

1. Statistics is used to **analyse** the data
2. Machine Learning is used to make **predictions**

   ➢ When you do Machine Learning you need to understand Statistics

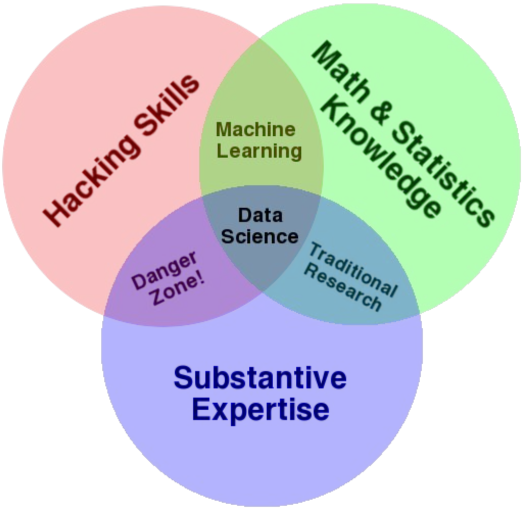
In practice:

1. When the data is **wide** (over 100 features) - it's ML
2. Variables are **correlated** - it's ML
3. Simple models are associated with Statistics (Linear Regression), While fancy methods are associated with Machine Learning (Random Forest)

Also, based on tools (i.e. R vs. Python debate), Statisticians have a reputation of being less good in software engineering.

What about Data Science?

What about Data Science?



Data Science is a more general term that is focusing on:

1. Machine Learning

2. Field expertise (e.g. I did a MSc in Mango Science)

3. Computer Literacy
   i. Data collection (API, scraping)
   ii. Databases (SQL in various flavours)
   iii. Deploying models on a server (Networking)
   iv. Data Visualization
   v. Not shy with Big Data (Hadoop, NoSQL)

   … Not Linus Torvalds but you don't need a "developer/babysitter" to watch you

4. You will define it

What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

Linear Regression

Regularization

Validation

Different Kinds of Machine Learning

Supervised Learning - Labeled data

- ❖ Classification - Predicting classes
  - ➢ Binary
  - ➢ Multiclass

- ❖ Regression - Predicting continouos values

Unsupervised Learning - Unlabeled data

- ❖ Clustering

Reinforcement Learning - Interactions between agent and environment

Different Kinds of Machine Learning

Classification Problem:

- Webmarketing Example: Campaign Ad
    - Marketing person for AIG
    - Sells car insurance
    - Decides to do an ad campaign to tell about a new offer
    - Each impression costs 0.1 cent
    - 0.1% of users click on the ad
    - 1% of visitors buy insurance
    - He needs 1 euro to get a visitor
    - He needs 100 euros to sell insurance for 1 person

- Objective 1: Increase CTR (Clickthrough rate)
- Objective 2: Increase conversion rate

Different Kinds of Machine Learning

Classification Problem:

- Webmarketing Example: Campaign Ad

  - Buy third-party data (LeMonde, Leboncoin, Blogs) about users
  - Launch your ad campaign
  - Discover the users that click and buy
  - Build a Machine Learning Model

  → Select users that have more probability to click and buy
  → If CTR was raised 0.1% to 1% we will cut cost per by 90%
  → If conversion rate was raised from 1% to 5% - costs go down by **98%**

Different Kinds of Machine Learning

Regression Problem:

- Retailer Example: Sales Prediction

  - You are the COO of Carrefour
  - You have more than 10,000 stores around the world
  - Big stores have on average 100 employees
  - Sales vary between stores, departments and time of the year
  - You need to organize your staff throughout the year

  → Too much staff - high labor cost
  → Not enough staff - blocks sales, bad reputation

Different Kinds of Machine Learning

Clustering Problem:

- ❖ E-Commerce Example: Fraud Detection

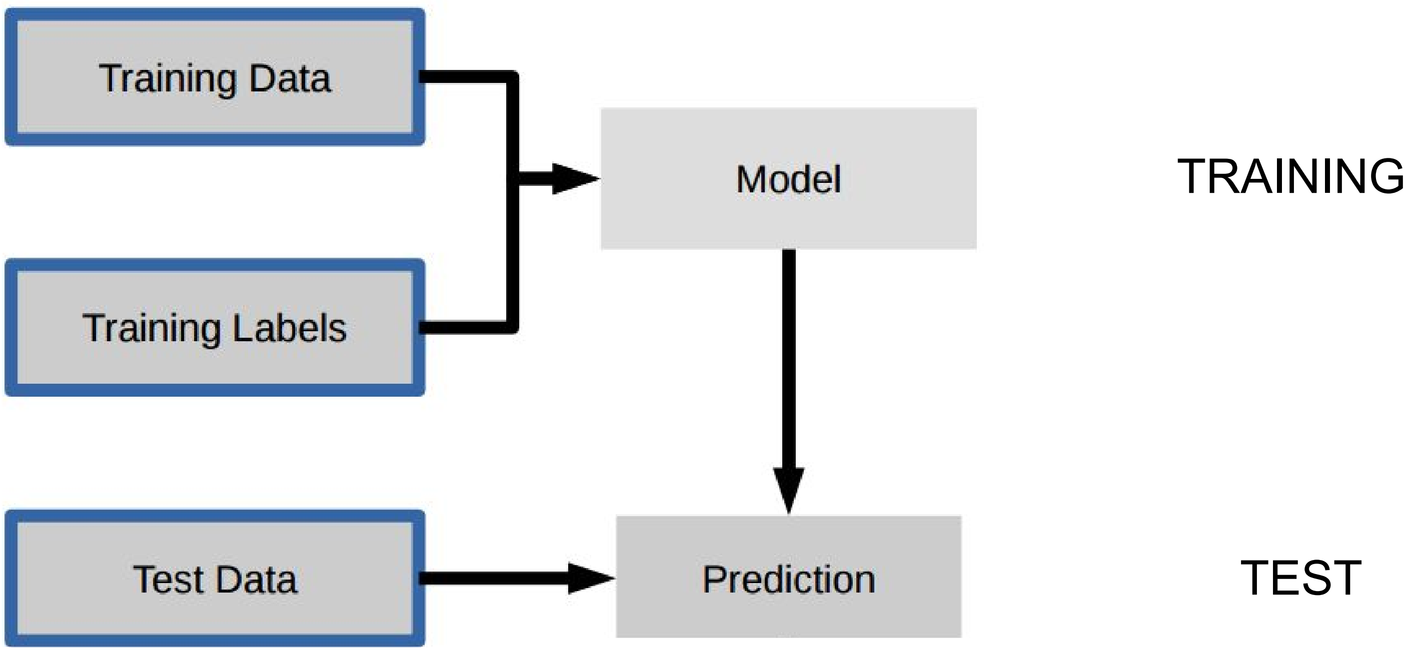
  - ❑ You are a manager at Visa
  - ❑ You work with dozens of analysts to find fraulant operations
  - ❑ Currently they are unable to go througly through all the records
  - ❑ You need to select **abnormal** operations
  - ❑ Many frauds are caught with classifications but scammers are smart and they are changing techniques constantly...

Different Kinds of Machine Learning

Reinforcement Problem:

- Robotics Example: Roomba robotic vacuum cleaner
  - You are an engineer at iRobot
  - You work on a new, intelligent model
  - The robot should vacuum the floor selectively: find dirty rooms (kids rooms, kitchen, entrance)
  - Design is lightweight: the robot's protection can take 30,000 hits from the wall, more than that the robot becomes vulnerable
  - Robot can sense when it picked dirt and when it hit the wall
  - In addition, you have several sensors (IR, Ultrasonic, sound)
  - You need to use inputs from sensors to define the best cleaning strategy

Supervised Learning



What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

Linear Regression

Regularization

Validation

Variable Types

- Categorical - Categories without clear ordering
  - Example: Gender is sales records can contain the labels “Male”, “Female” or “Unknown”
  - Example: Country in a GDP prediction contains “France”, “Germany”, and 200 other countries

- Numerical - Continuous or discrete, Positive, Negative or zero
  - Example: Age in sales records

- Dates - need to be transformed to categoricals and numerical

- Ordinal - Categories with order
  - Example: Predicting T-shirt size: “Large”, “Medium”, “Small”

One-Hot Encoding (Dummification)

- Regression models deal well with numerical data
  - We need to convert categorical data to numerical values
  - We can do this using dummification (one-hot-encoding)
- Example:

```
import pandas as pd

df = pd.DataFrame([ ['green', 1, 10.1, 0], ['red', 2, 13.5, 1], ['blue', 3, 15.3, 0]])

df.columns = ['color', 'size', 'prize', 'class label']

df
```

	color	size	prize	class label
0	green	1	10.1	0
1	red	2	13.5	1
2	blue	3	15.3	0

```
pd.get_dummies(df)
```

	size	prize	class label	color_blue	color_green	color_red
0	1	10.1	0	0	1	0
1	2	13.5	1	0	0	1
2	3	15.3	0	1	0	0

One-Hot Encoding (Dummification)

- When using OHE make sure to encode **all** data (train and test)

- Missing values can be also useful sometimes, make sure to create a category for them

  `pd.get_dummies(data, dummy_na=True)`

  - When is it useful?

- If there are too many categories your table might be too wide
  - This is not good. Why?
  - You can eliminate categories in various methods, for example select only the top 10 categories. The rest label as “others”

- Be careful about numericals “in disguise” - e.g. index numbers

- There are other methods to serialize categories:
  - replace category with average of target

Binning

- Use continuous features to create new categorical variables
- Associate ranges of values with buckets

```
df
```

	regiment	company	name	preTestScore	postTestScore
0	Nighthawks	1st	Miller	4	25
1	Nighthawks	1st	Jacobson	24	94
2	Nighthawks	2nd	Ali	31	57
3	Nighthawks	2nd	Milner	2	62
4	Dragoons	1st	Cooze	3	70
5	Dragoons	1st	Jacon	4	25
6	Dragoons	2nd	Ryaner	24	94
7	Dragoons	2nd	Sone	31	57
8	Scouts	1st	Sloan	2	62
9	Scouts	1st	Piger	3	70
10	Scouts	2nd	Riani	2	62
11	Scouts	2nd	Ali	3	70

12 rows × 5 columns

```
bins = [0, 25, 50, 75, 100]
group_names = ['Low', 'Okay', 'Good', 'Great']
categories = pd.cut(df['postTestScore'], bins, labels=group_names)
df['categories'] = pd.cut(df['postTestScore'], bins, labels=group_names)
```

Binning

- Use continuous features to create new variables

```
df
```

	regiment	company	name	preTestScore	postTestScore	scoresBinned	categories
0	Nighthawks	1st	Miller	4	25	(0, 25]	Low
1	Nighthawks	1st	Jacobson	24	94	(75, 100]	Great
2	Nighthawks	2nd	Ali	31	57	(50, 75]	Good
3	Nighthawks	2nd	Milner	2	62	(50, 75]	Good
4	Dragoons	1st	Cooze	3	70	(50, 75]	Good
5	Dragoons	1st	Jacon	4	25	(0, 25]	Low
6	Dragoons	2nd	Ryaner	24	94	(75, 100]	Great
7	Dragoons	2nd	Sone	31	57	(50, 75]	Good
8	Scouts	1st	Sloan	2	62	(50, 75]	Good
9	Scouts	1st	Piger	3	70	(50, 75]	Good
10	Scouts	2nd	Riani	2	62	(50, 75]	Good
11	Scouts	2nd	Ali	3	70	(50, 75]	Good

12 rows × 7 columns

- Sometimes binning makes sense:
  - Seperate age<18 (minors) or age>60 (retired)
  - Seperate grades (failed vs. passed)
  - Seperate distances by walking vs. driving vs. flying

- Don't do this systematically everytime! Think, try, validate

Dealing with Missing Data

- Several possibilities to deal with it:

  - Remove Data - Only if missing data is **small** and **localized**
    - Remove entire row if you have many rows
    - Remove entire column if data is **systematiclly** missing in a column

  - Otherwise - Impute Data
    - Replace missing values with mean / median / mode
      - When should we use median instead of mean?
      - When should we use mode?
    - Replace data with 0 if variable is counting things
      - e.g. assume None/NaN as 0
    - Advanced: Use separation values - e.g. negative values

  In Pandas:

```
df.fillna(df.mean())
```

- Before doing anything fancy don’t forget to **look at the data carefully**!

Preprocessing Notebook

`Titanic Data Set`

What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

Linear Regression

Regularization

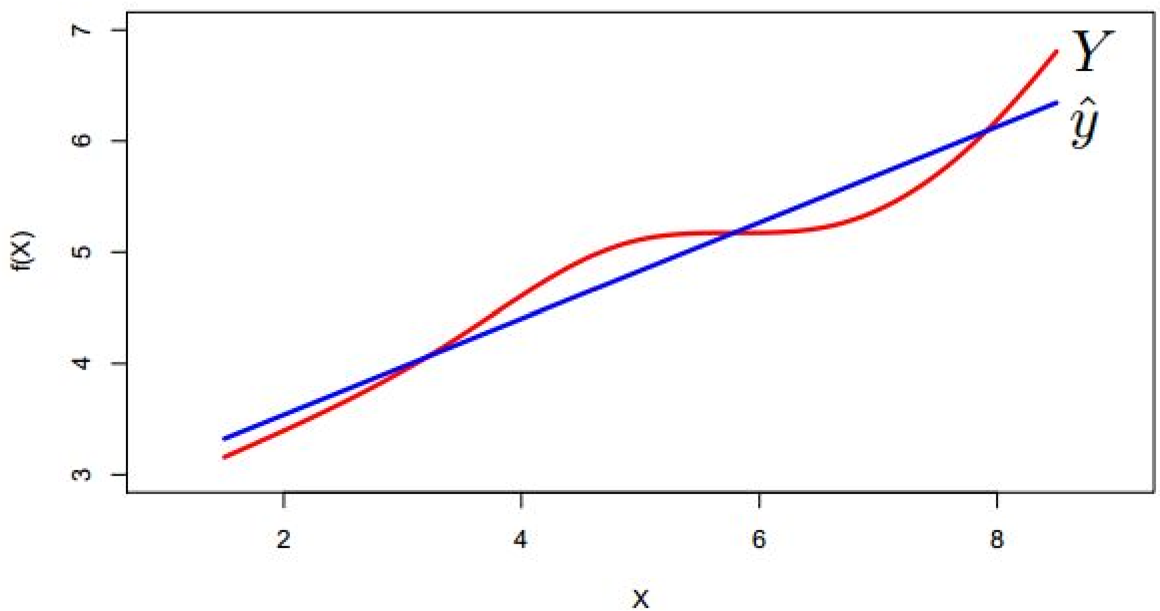
Validation

Linear Regression

(let's refresh our memory ...)

- Assuming that the relationship can be described by: $$Y = \beta_0 + \beta_1 X + \epsilon$$
- Where β_1 is the slope and β_0 is the intercept
- Error ε follows a gaussian distribution

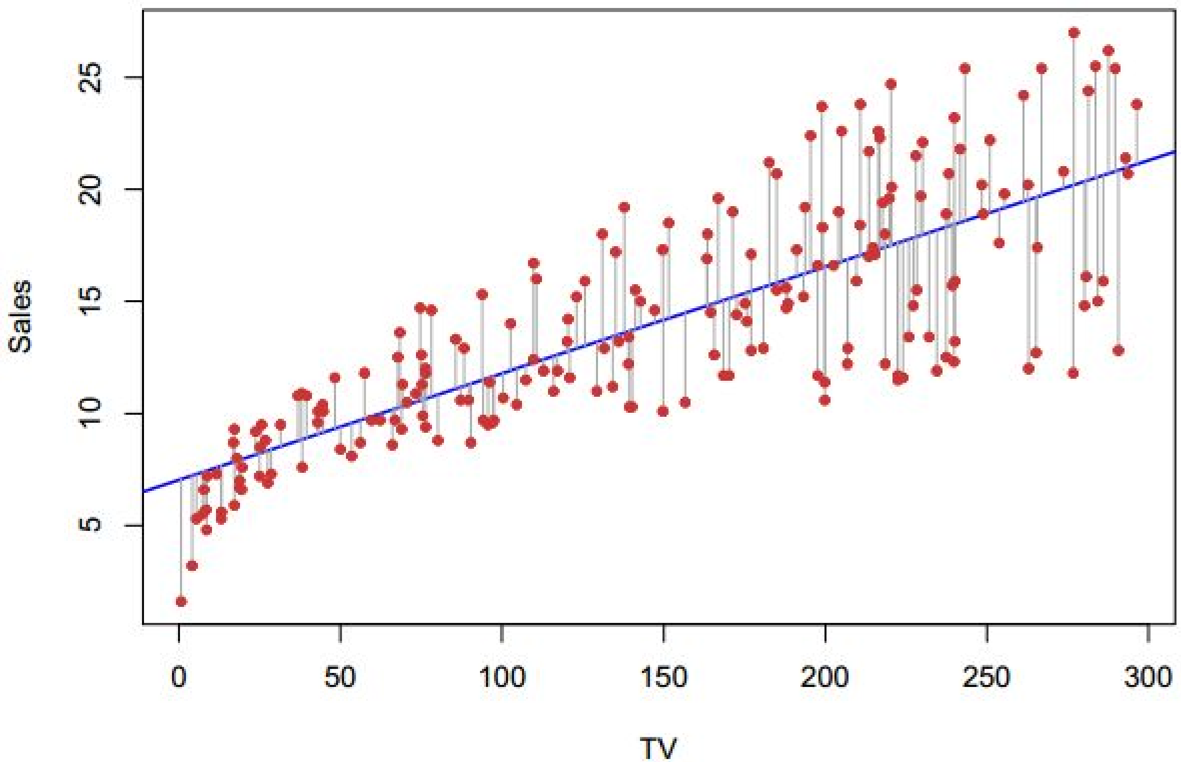
➡ We can estimate the coefficiencies β_1, β_0: $$\hat{y} = \hat{\beta}_0 + \hat{\beta}_1 x$$



Linear Regression

(let's refresh our memory ...)

- We want to fit a linear model that will have minimal distance to the data points we obtained (Sales / Budget) :



Linear Regression

(let's refresh our memory ...)

- To do this, we create a loss function and try to minimize it using **Least Squares**:
- Using residual sum of squares that describes the goodness of the fit:

$$\mathrm{RSS} = e_1^2 + e_2^2 + \cdots + e_n^2$$

$$\mathrm{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

- We want to find β_0 β_1 that will minimize RSS.
- It's simple to see that RSS is minimal at:

$$\hat{\beta}_1 = \frac{\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})}{\sum_{i=1}^{n}(x_i - \bar{x})^2}$$

$$\hat{\beta}_0 = \bar{y} - \hat{\beta}_1 \bar{x},$$

where:

$$\bar{y} \equiv \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\bar{x} \equiv \frac{1}{n}\sum_{i=1}^{n} x_i$$

Scikit-learn

- Open-source machine learning library for Python
- Written in Python and C / C++
- Requires NumPy, SciPy to be installed
- Includes contributions from INRIA and Google
- One-stop shop for your ML needs

Linear Regression in Scikit-Learn

```python
from sklearn.linear_model import LinearRegression

regr = LinearRegression()

# Train the model using the training set
regr.fit(X_train, y_train)

# Predict target for the testing set
y_hat = regr.predict(X_test)
```

Linear Regression Notebook

`Boston housing dataset`

What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

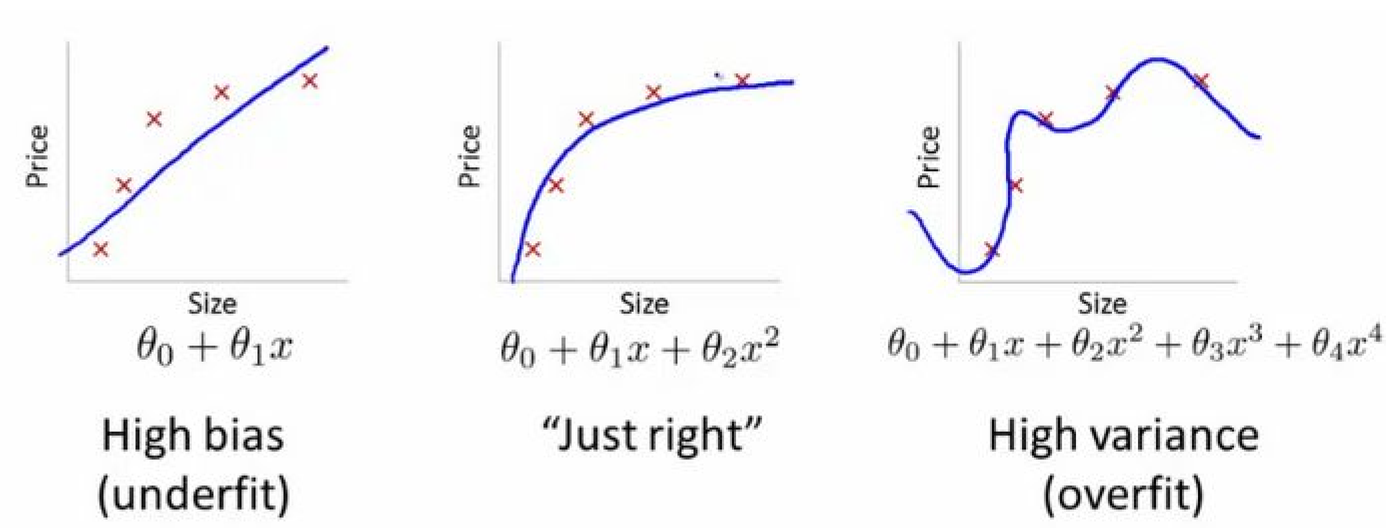
Linear Regression

Regularization

Validation

Regularization

- A good predictive model is a model that can **generalizex**
- Performance is measured as error for **unseen observations**
- This is why we must Regularize:
  - Simplify our model (less parameters)
  - Reduce variables
  - This is also called **Bias-Variance Tradeoff**



Regularized Linear Regression

- Instead of using minimizing the cost function:

$$\mathrm{RSS} = (y_1 - \hat{\beta}_0 - \hat{\beta}_1 x_1)^2 + (y_2 - \hat{\beta}_0 - \hat{\beta}_1 x_2)^2 + \ldots + (y_n - \hat{\beta}_0 - \hat{\beta}_1 x_n)^2$$

- Two common methods for regularization:

  - <u>Lasso Regression</u>
    - Loss Function = RSS + Γ ⨉ $||\beta||_1$

  - <u>Ridge Regression</u>
    - Loss Function = RSS + Γ ⨉ $||\beta||_2$

Γ is a coefficient that we need to select

$||\beta||_1$ - l1 norm $\|x\|_1 = \sum_i |x_i|$

$||\beta||_2$ - l2 norm $\|x\|_2 = \sqrt{\sum_i x_i^2}$

equancy

Regularized Linear Regression

Both implemented in sklearn:

```
>>> from sklearn.linear_model import Ridge
>>> clf = Ridge(alpha=1.0)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
```

```
>>> from sklearn.linear_model import Lasso
>>> clf = Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
```

Regularized Linear Regression

Both implemented in sklearn:

```
>>> from sklearn.linear_model import Ridge
>>> clf = Ridge(alpha=1.0)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
```

```
>>> from sklearn.linear_model import Lasso
>>> clf = Lasso(alpha=0.1)
>>> clf.fit([[0,0], [1, 1], [2, 2]], [0, 1, 2])
```

What is Machine Learning?

Different Kinds of Machine Learning

Preprocessing

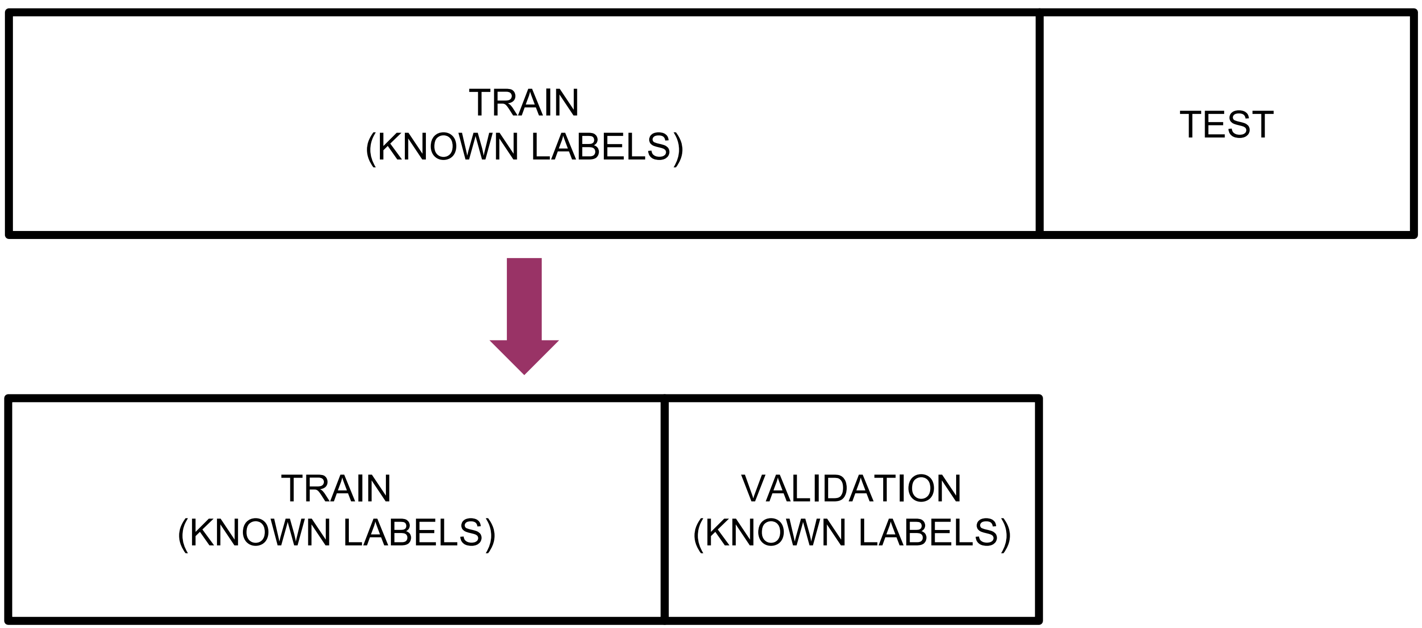
Linear Regression

Regularization

Validation

Validation

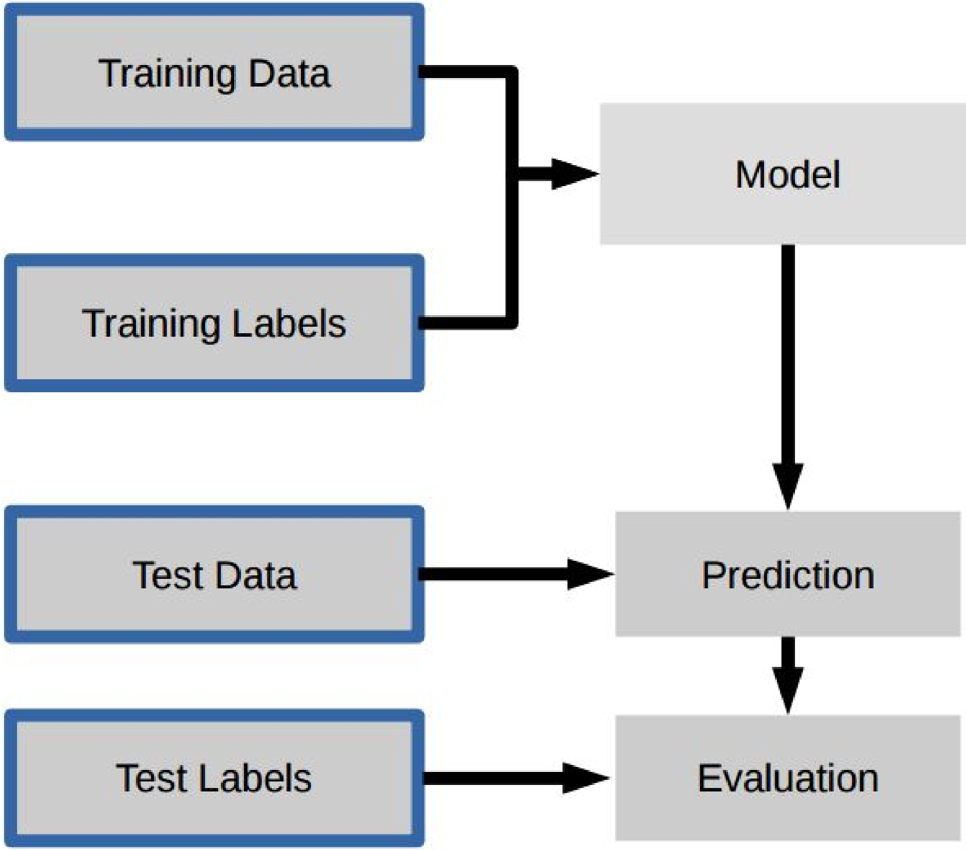
- Fixed Validation:
  - **Split** train set into 2 sets
  - Splitting must be random



In Python:

from sklearn.cross_validation import train_test_split

Supervised Learning



TRAINING

VALIDATION

Evaluation

Setting Metrics to a model is **key**.

MSE:

- Used as a cost function for linear regression
- Aggressively punishes big errors
- Symmetrical

RMSE:

- Like MSE but same scale as target

MAE:

- Easiest interpretation, good for reporting

MAPE:

- Useful when target has a large variation
- Expressed in percentage

Mean squared error	$\text{MSE} = \frac{1}{n}\sum_{t=1}^{n} e_t^2$
Root mean squared error	$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{t=1}^{n} e_t^2}$
Mean absolute error	$\text{MAE} = \frac{1}{n}\sum_{t=1}^{n} \lvert e_t \rvert$
Mean absolute percentage error	$\text{MAPE} = \frac{100\%}{n}\sum_{t=1}^{n} \left\lvert \frac{e_t}{y_t} \right\rvert$

Evaluation

Setting Metrics to a model is **key.**

- Evaluation should be driven by a **business objectives** (i.e. real-life)
  - Our predictions will **always** have errors !
    - Adequat evaluation is basis to **comparison** and continuous **improvement**

- Important questions to ask yourself:
  - What happenes when we predict too high / too low ? **Symmetricity**
  - When is it important to know if you over-predict or under-predict?

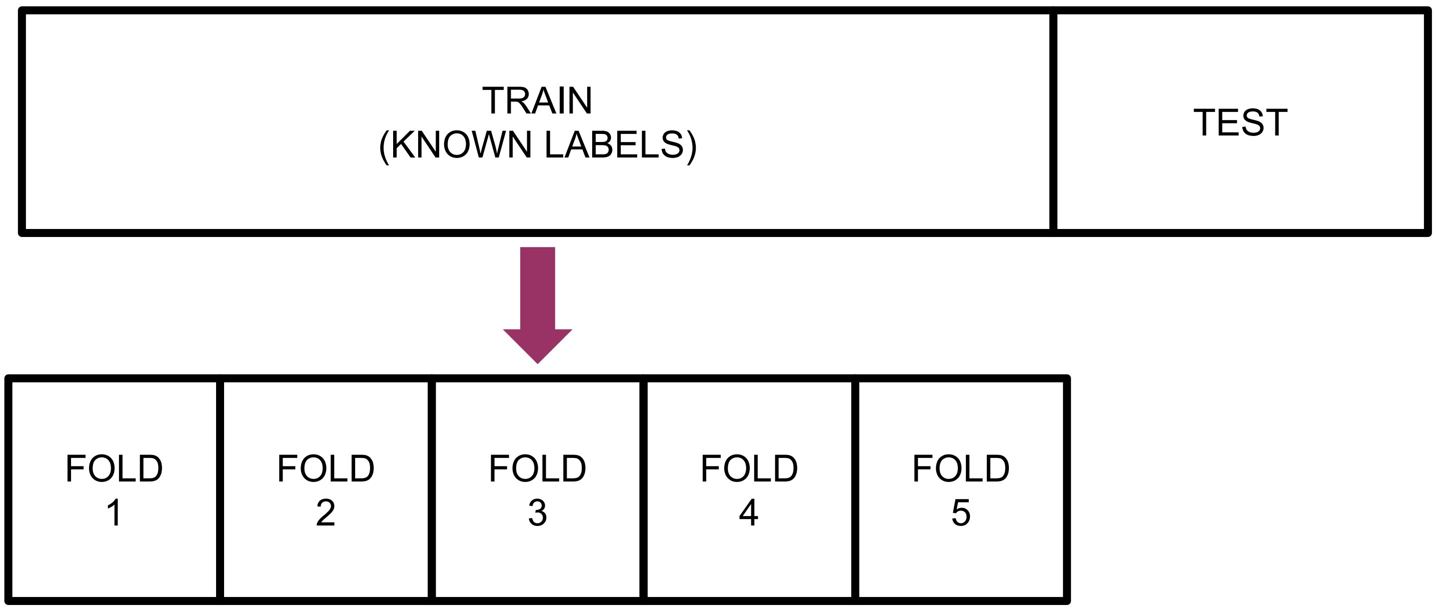
- What happenes when **extreme** errors shouldn't have additional cost ?
  - Predict sales for a store
  - A model that gives low errors for 51 weeks but very high error for one week could be useful, maybe the **Square Error** assumption should be relaxed ?
    - If in the end we want to have a yearly evaluation we should use **Average Errors** instead.

Linear Regression Notebook

`Boston housing dataset`

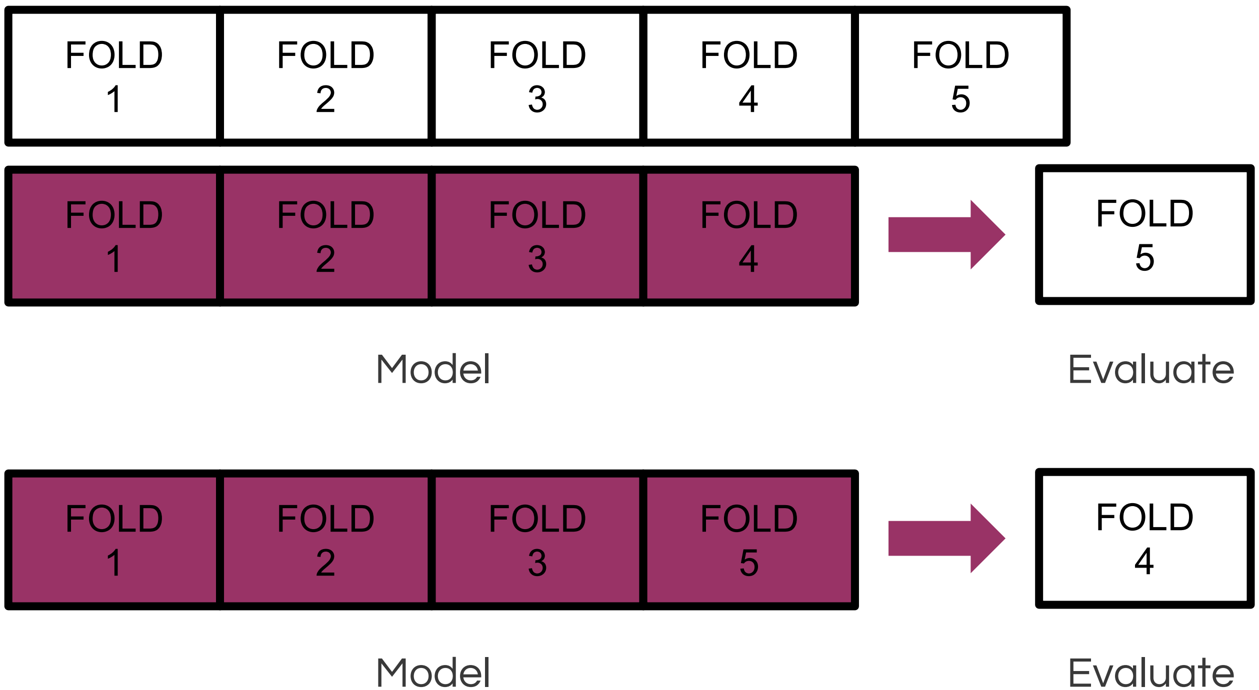
Validation

- Cross-Validation (k-fold):
  - Split train set into k **stratified** sets randomly (example: 5-fold)
  - Train 5 models and predict data



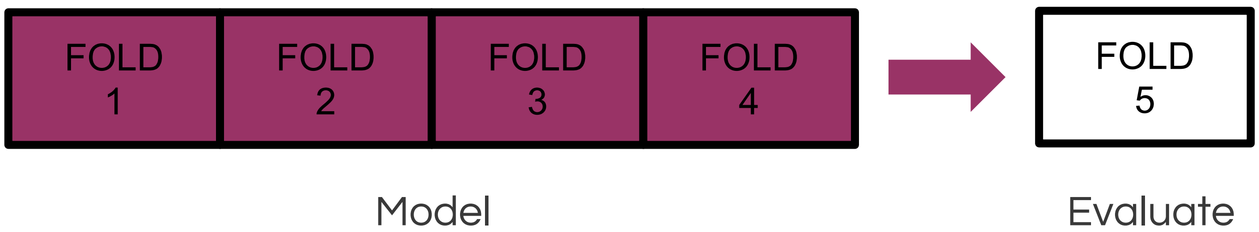
Validation

❖ Cross-Validation (k-fold):



Validation

- ❖ Repeat 5 times
- ❖ Collect 5 scores
- ❖ Use average



Advantages:

- ❖ Uses more data
- ❖ Evaluating on the entire dataset

Disadvantage:

- ❖ Slow (need to fit and predict 5 times)
- ❖ Sometimes, we don’t want the evaluation to be random

Model Tuning

- Using the validation method we can now compare between models

- What should we compare:
  - Different models (LR, Lasso, Ridge, ...)
  - Feature sets
  - Hyperparameters for the models (e.g. Γ for Lasso)
  - Imputation methods (e.g. mean vs . median)
  - ...

- This is where Machine Learning is **Art** rather than science

- We can't always try **all** the possibilities

- We need to design a **work plan** to make useful tests

Grid-Search

- To make many tests we need to use **automatic** methods
- We need to test different combinations of variations and pick the best
- We need to estimate time and IT resources required for our tests
- ***GridSearchCV*** is you friend:

```
>>> from sklearn import grid_search, datasets
>>> from sklearn.linear_model import Ridge
>>> clf = Ridge()
>>> iris = datasets.load_iris()
>>> parameters = {'alpha':[0, 1.0, 10.0]}
>>> gs = grid_search.GridSearchCV(clf, parameters)
>>> gs.fit(iris.data, iris.target)
```

Linear Regression Notebook

`Boston housing dataset`

Thanks a lot!

kkarp@equancy.com